TRUST Reference Manual V1.8.4

Support team: trust@cea.fr

Link to: **TRUST Generic Guide**

December 9, 2021

Contents

1 Syntax to define a mathematical function

In a mathematical function, used for example in field definition, it's possible to use the predifined function (an object parser is used to evaluate the functions) :

ABS	: absolute value function
COS	: cosine function
SIN	: sine function
TAN	: tangent function
ATAN	: arctangent function
EXP	: exponential function
LN	: natural logarithm function
SQRT	: square root function
INT	: integer function
ERF	: error function
RND(x)	: random function (values between 0 and x)
COSH	: hyperbolic cosine function
SINH	: hyperbolic sine function
TANH	: hyperbolic tangent function
ACOS	: inverse cosine function
ASIN	: inverse sine function
ATANH	: inverse hyperbolic tangent function
NOT(x)	: NOT x (returns 1 if x is false, 0 otherwise)
SGN(x)	: SGN x (returns 1 if x is positive, -1 if negative, 0 if zero)
x_AND_y	: boolean logical operation AND (returns 1 if both x and y are true, else 0)
x_OR_y	: boolean logical operation OR (returns 1 if x or y is true, else 0)
x_GT_y	: greater than (returns 1 if x>y, else 0)
x_GE_y	: greater than or equal to (returns 1 if x>=y, else 0)
x_LT_y	: less than (returns 1 if x<y, else 0)
x_LE_y	: less than or equal to (returns 1 if x<=y, else 0)
x_MIN_y	: returns the smallest of x and y
x_MAX_y	: returns the largest of x and y
x_MOD_y	: modular division of x per y
x_EQ_y	: equal to (returns 1 if x==y, else 0)
x_NEQ_y	: not equal to (returns 1 if x!=y, else 0)

You can also use the following operations:

+	: addition
-	: subtraction
/	: division

* : multiplication
% : modulo
$: max
ˆ : power
< : less than
> : greater than
[: less than or equal to
] : greater than or equal to

You can also use the following constants:
Pi : pi value (3,1415...)

The variables which can be used are:
x,y,z : coordinates
t : time

Examples:
Champ_front_fonc_txyz 2 cos(y+xˆ2) t+ln(y)
Champ_fonc_xyz dom 2 tanh(4*y)*(0.95+0.1*rnd(1)) 0.

Possible errors:
Error 1:
Champ_fonc_txyz 1 cos(10*t)*(1<x<2)*(1<y<2)
Previous line is wrong. It should be written as:
Champ_fonc_txyz 1 cos(10*t)*(1<x)*(x<2)*(1<y)*(y<2)

Error 2:
Champ_front_fonc_xyz 1 20*(x<-2)+10*(y]-5)+3*(z>0)
Previous line is wrong because negative values are not written between parentheses. It should be written as:
Champ_front_fonc_xyz 1 20*(x<(-2))+10*(y](-5))+3*(z>0)

2 Existing & predefined fields names

Here is a list of post-processable fields, but it is not the only ones.

Physical values	**Keyword for field_name**	**Unit**
Velocity	**Vitesse** or **Velocity**	$m.s^{-1}$
Velocity residual	**Vitesse_residu**	$m.s^{-2}$
Kinetic energy per elements $(0.5\rho\|\|u_i\|\|^2)$	**Energie_cinetique_elem**	$kg.m^{-1}.s^{-2}$
Total kinetic energy $\left(\frac{\sum_{i=1}^{nb\_elem} 0.5\rho\|\|u_i\|\|^2 vol_i}{\sum_{i=1}^{nb\_elem} vol_i}\right)$	**Energie_cinetique_totale**	$kg.m^{-1}.s^{-2}$
Vorticity	**Vorticite**	s^{-1}
Pressure in incompressible flow $(P/\rho + gz)$ For Front Tracking probleme $(P + \rho gz)$	**Pression** [1]	$Pa.m^3.kg^{-1}$ or Pa

... continued on next page ...

[1]The post-processed pressure is the pressure divided by the fluid's density $(P/\rho + gz)$ on incompressible laminar calculation. For turbulent, pressure is $P/\rho + gz + 2/3 * k$ cause the turbulent kinetic energy is in the pressure gradient.

Physical values	**Keyword for field_name**	**Unit**
Pressure in incompressible flow (P+ρgz)	**Pression_pa** or **Pressure**	Pa
Pressure in compressible flow	**Pression**	Pa
Hydrostatic pressure (ρgz)	**Pression_hydrostatique**	Pa
Totale pressure (when quasi compressible model is used)=Pth+P	**Pression_tot**	Pa
Pressure gradient $(\nabla(P/\rho + gz))$	**Gradient_pression**	$m.s^{-2}$
Velocity gradient	**gradient_vitesse**	s^{-1}
Temperature	**Temperature**	^{o}C or K
Temperature residual	**Temperature_residu**	^{o}C.s^{-1} or K.s^{-1}
Phase temperature of a two phases flow	**Temperature_EquationName**	^{o}C or K
Mass transfer rate between two phases	**Temperature_mpoint**	$kg.m^{-2}.s^{-1}$
Temperature variance	**Variance_Temperature**	K^2
Temperature dissipation rate	**Taux_Dissipation_Temperature**	$K^2.s^{-1}$
Temperature gradient	**Gradient_temperature**	$K.m^{-1}$
Heat exchange coefficient	**H_echange_Tref** [2]	$W.m^{-2}.K^{-1}$
Turbulent heat flux	**Flux_Chaleur_Turbulente**	$m.K.s^{-1}$
Turbulent viscosity	**Viscosite_turbulente**	$m^2.s^{-1}$
Turbulent dynamic viscosity (when quasi compressible model is used)	**Viscosite_dynamique_turbulente**	$kg.m.s^{-1}$
Turbulent kinetic energy	**K**	$m^2.s^{-2}$
Turbulent dissipation rate	**Eps**	$m^3.s^{-1}$
Turbulent quantities K and Epsilon	**K_Eps**	($m^2.s^{-2}$,$m^3.s^{-1}$)
Residuals of turbulent quantities K and Epsilon residuals	**K_Eps_residu**	($m^2.s^{-3}$,$m^3.s^{-2}$)
Constituent concentration	**Concentration**	
Constituent concentration residual	**Concentration_residu**	
Component velocity along X	**VitesseX**	$m.s^{-1}$
Component velocity along Y	**VitesseY**	$m.s^{-1}$
Component velocity along Z	**VitesseZ**	$m.s^{-1}$
Mass balance on each cell	**Divergence_U**	$m^3.s^{-1}$
Irradiancy	**Irradiance**	$W.m^{-2}$
Q-criteria	**Critere_Q**	s^{-1}
Distance to the wall $Y^+ = yU/\nu$ (only computed on boundaries of wall type)	**Y_plus**	dimensionless
Friction velocity	**U_star**	$m.s^{-1}$
Void fraction	**alpha**	dimensionless
Cell volumes	**Volume_maille**	m^3
Chemical potential	**Potentiel_Chimique_Generalise**	
Source term in non Galinean referential	**Acceleration_terme_source**	$m.s^{-2}$

... continued on next page ...

[2]Tref indicates the value of a reference temperature and must be specified by the user. For example, H_echange_293 is the keyword to use for Tref=293K.

Physical values	**Keyword for field_name**	**Unit**
Stability time steps	**Pas_de_temps**	S
Listing of boundary fluxes	**Flux_bords**	cf each *.out file
Volumetric porosity	**Porosite_volumique**	dimensionless
Distance to the wall	**Distance_Paroi** [3]	m
Volumic thermal power	**Puissance_volumique**	$W.m^{-3}$
Local shear strain rate defined as $\sqrt{(2SijSij)}$	**Taux_cisaillement**	s^{-1}
Cell Courant number (VDF only)	**Courant_maille**	dimensionless
Cell Reynolds number (VDF only)	**Reynolds_maille**	dimensionless
Viscous force	**viscous_force**	$kg.m^2.s^{-1}$
Pressure force	**pressure_force**	$kg.m^2.s^{-1}$
Total force	**total_force**	$kg.m^2.s^{-1}$
Viscous force along X	**viscous_force_x**	$kg.m^2.s^{-1}$
Viscous force along Y	**viscous_force_y**	$kg.m^2.s^{-1}$
Viscous force along Z	**viscous_force_z**	$kg.m^2.s^{-1}$
Pressure force along X	**pressure_force_x**	$kg.m^2.s^{-1}$
Pressure force along Y	**pressure_force_y**	$kg.m^2.s^{-1}$
Pressure force along Z	**pressure_force_z**	$kg.m^2.s^{-1}$
Total force along X	**total_force_x**	$kg.m^2.s^{-1}$
Total force along Y	**total_force_y**	$kg.m^2.s^{-1}$
Total force along Z	**total_force_z**	$kg.m^2.s^{-1}$

3 interprete

Description: Basic class for interpreting a data file. Interpretors allow some operations to be carried out on objects.

See also: objet_u (34) read (3.81) associate (3.12) discretize (3.31) mailler (3.60) maillerparallel (3.62) ecrire_fichier_bin (3.121) ecrire (3.120) read_file (3.82) lire_tgrid (3.84) solve (3.99) execute_parallel (3.37) end (3.50) dimension (3.28) bidim_axi (3.14) axi (3.13) transformer (3.111) rotation (3.96) dilate (3.27) criteres_convergence (3.22) testeur (3.104) test_solveur (3.103) postraiter_domaine (3.77) modif_bord_to_raccord (3.63) remove_elem (3.90) regroupebord (3.89) supprime_bord (3.100) calculer_moments (3.18) imprimer_flux (3.52) decouper_bord_coincident (3.26) raffiner_anisotrope (3.79) raffiner_isotrope (3.80) trianguler (3.112) tetraedriser (3.106) orientefacesbord (3.67) reorienter_tetraedres (3.93) reorienter_triangles (3.94) verifiercoin (3.118) porosites (3.74) porosites_champ (3.76) discretiser_domaine (3.30) { (3.24) } (3.51) export (3.38) debog (3.23) pilote_icoco (3.72) moyenne_volumique (3.64) lire_ideas (3.59) system (3.102) redresser_hexaedres_vdf (3.87) analyse_angle (3.11) remove_invalid_internal_boundaries (3.92) reordonner (3.95) precisiongeom (3.78) nettoiepasnoeuds (3.65) scatter (3.97) distance_paroi (3.32) extruder (3.46) extract_2d_from_3d (3.39) extruder_en20 (3.48) extrudeparoi (3.45) decoupebord (3.25) extraire_plan (3.42) extraire_domaine (3.41) extraire_surface (3.43) integrer_champ_med (3.54) orienter_simplexes (3.86) verifier_simplexes (3.117) verifier_qualite_raffinements (3.115) testeur_medcoupling (3.105) bloc_phases (3.16) phases (3.71) option_vdf (3.66) bloc_b (3.15) espece (3.36) Op_Conv_EF_Stab_PolyMAC_Face (3.6) Option_CoviMAC (3.7) Op_Conv_EF_Stab_CoviMAC_Face (3.5) Op_Conv_EF_Stab_CoviMAC_Elem (3.4) ecrire_med (3.122) read_med (3.9) lata_to_other (3.58) lata_to_med (3.56) ecrire_champ_med (3.33) Merge_MED (3.2) ecriturelecturespecial (3.35) Raffiner_isotrope_parallele (3.8) extrudebord (3.44) corriger_frontiere_periodique (3.20) refine_mesh (3.88) polyedriser (3.73) interprete_geometrique_base (3.55) partition_multi (3.70) partition (3.68) Deactivate_SIGINT_Catch (3.1) disable_TU (3.29) MultipleFiles (3.3)

[3] distance_paroi is a field which can be used only if the mixing length model (see 2.15.1.2) is used in the data file.

Usage:
interprete

3.1 Deactivate_sigint_catch

Description: Flag to disable the detection of the signal SIGINT.

See also: interprete (3)

Usage:
Deactivate_SIGINT_Catch

3.2 Merge_med

Description: This keyword allows to merge multiple MED files produced during a parallel computation into a single MED file.

See also: interprete (3)

Usage:
Merge_MED med_files_base_name time_iterations
where

- **med_files_base_name** *str*: Base name of multiple med files that should appear as base_name-_xxxxx.med, where xxxxx denotes the MPI rank number. If you specify NOM_DU_CAS, it will automatically take the basename from your datafile's name.
- **time_iterations** *str into ['all_times', 'last_time']*: Identifies whether to merge all time iterations present in the MED files or only the last one.

3.3 Multiplefiles

Description: Change MPI rank limit for multiple files during I/O

See also: interprete (3)

Usage:
MultipleFiles type
where

- **type** *int*: New MPI rank limit

3.4 Op_conv_ef_stab_covimac_elem

Description: Class Op_Conv_EF_Stab_CoviMAC_Elem

See also: interprete (3)

Usage:
Op_Conv_EF_Stab_CoviMAC_Elem {

[**alpha** *float*]

}
where

- **alpha** *float*: parametre ajustant la stabilisation de 0 (schema centre) a 1 (schema amont)

3.5 Op_conv_ef_stab_covimac_face

Description: Class Op_Conv_EF_Stab_CoviMAC_Face

See also: interprete (3)

Usage:
Op_Conv_EF_Stab_CoviMAC_Face {

[**alpha** *float*]

}
where

- **alpha** *float*: parametre ajustant la stabilisation de 0 (schema centre) a 1 (schema amont)

3.6 Op_conv_ef_stab_polymac_face

Description: Class Op_Conv_EF_Stab_PolyMAC_Face_PolyMAC

See also: interprete (3)

Usage:
Op_Conv_EF_Stab_PolyMAC_Face {

[**alpha** *float*]

}
where

- **alpha** *float*: parametre ajustant la stabilisation de 0 (schema centre) a 1 (schema amont)

3.7 Option_covimac

Description: Class of CoviMAC options.

See also: interprete (3)

Usage:
Option_CoviMAC {

[**interp_ve1** *int*]

}
where

- **interp_ve1** *int*: Flag to enable a first order velocity face-to-element interpolation (the default value is 0 which means a second order interpolation)

3.8 Raffiner_isotrope_parallele

Description: Refine parallel mesh in parallel

See also: interprete (3)

Usage:
Raffiner_isotrope_parallele {

> **name_of_initial_zones** *str*
> **name_of_new_zones** *str*
> [**ascii**]
> [**single_hdf**]

}
where

- **name_of_initial_zones** *str*: name of initial Zones
- **name_of_new_zones** *str*: name of new Zones
- **ascii** : writing Zones in ascii format
- **single_hdf** : writing Zones in hdf format

3.9 Read_med

Synonymous: **lire_med**

Description: Keyword to read MED mesh files where domain_name corresponds to the domain name, filename.med corresponds to the file (written in format MED) containing the mesh named mesh_name.
Note about naming boundaries: When reading filename.med, TRUST will detect boundaries between domain (Raccord) when the name of the boundary begins by type_raccord_. For example, a boundary named type_raccord_wall in filename.med will be considered by TRUST as a boundary named wall between two domains.
NB: To read several domains from a mesh issued from a MED file, use Read_Med to read the mesh then use Create_domain_from_sous_zone keyword.
NB: If the MED file contains one or several subzone defined as a group of volumes, then Read_MED will read it and will create two files domain_name_ssz.geo and domain_name_ssz_par.geo defining the subzones for sequential and/or parallel calculations. These subzones will be read in sequential in the datafile by including (after Read_Med keyword) something like:
Read_Med ....
Read_file domain_name_ssz.geo ;
During the parallel calculation, you will include something:
Scatter { ... }
Read_file domain_name_ssz_par.geo ;

See also: interprete (3) lire_medfile (3.10)

Usage:
read_med [**vef**] [**convertAllToPoly**] [**family_names_from_group_names**] [**short_family_names**] **nom_dom** **nom_dom_med** **file**
where

- **vef** *str into ['vef']*: Option vef is obsolete and is kept for backward compatibility.
- **convertAllToPoly** *str into ['convertAllToPoly']*: Option to convert mesh with mixed cells into polyhedras/polygons cells

- **family_names_from_group_names** *str into ['family_names_from_group_names']*: The option family-_names_from_group_names uses the group names instead of the family names to detect the boundaries into a MED mesh (useful when trying to read a MED mesh file from Gmsh tool which can now read and write MED meshes).
- **short_family_names** *str into ['short_family_names']*: The option short_family_names is useful to suppress FAM_-*_ from the boundary names of the MED meshes.
- **nom_dom** *str*: corresponds to the domain name
- **nom_dom_med** *str*: name of the mesh in med file
- **file** *str*: corresponds to the file (written in format MED) containing the mesh

3.10 Lire_medfile

Description: Obsolete keyword to read a mesh with MED file API

See also: read_med (3.9)

Usage:
lire_medfile [**vef**] [**convertAllToPoly**] [**family_names_from_group_names**] [**short_family_names**] **nom_dom** **nom_dom_med** **file**
where

- **vef** *str into ['vef']*: Option vef is obsolete and is kept for backward compatibility.
- **convertAllToPoly** *str into ['convertAllToPoly']*: Option to convert mesh with mixed cells into polyhedras/polygons cells
- **family_names_from_group_names** *str into ['family_names_from_group_names']*: The option family-_names_from_group_names uses the group names instead of the family names to detect the boundaries into a MED mesh (useful when trying to read a MED mesh file from Gmsh tool which can now read and write MED meshes).
- **short_family_names** *str into ['short_family_names']*: The option short_family_names is useful to suppress FAM_-*_ from the boundary names of the MED meshes.
- **nom_dom** *str*: corresponds to the domain name
- **nom_dom_med** *str*: name of the mesh in med file
- **file** *str*: corresponds to the file (written in format MED) containing the mesh

3.11 Analyse_angle

Description: Keyword Analyse_angle prints the histogram of the largest angle of each mesh elements of the domain named name_domain. nb_histo is the histogram number of bins. It is called by default during the domain discretization with nb_histo set to 18. Useful to check the number of elements with angles above 90 degrees.

See also: interprete (3)

Usage:
analyse_angle **domain_name** **nb_histo**
where

- **domain_name** *str*: Name of domain to resequence.
- **nb_histo** *int*

3.12 Associate

Synonymous: **associer**

Description: This interpretor allows one object to be associated with another. The order of the two objects in this instruction is not important. The object objet_2 is associated to objet_1 if this makes sense; if not either objet_1 is associated to objet_2 or the program exits with error because it cannot execute the Associate (Associer) instruction. For example, to calculate water flow in a pipe, a Pb_Hydraulique type object needs to be defined. But also a Domaine type object to represent the pipe, a Scheme_euler_explicit type object for time discretization, a discretization type object (VDF or VEF) and a Fluide_Incompressible type object which will contain the water properties. These objects must then all be associated with the problem.

See also: interprete (3)

Usage:
associate objet_1 objet_2
where

- **objet_1** *str*: Objet_1
- **objet_2** *str*: Objet_2

3.13 Axi

Description: This keyword allows a 3D calculation to be executed using cylindrical coordinates (R,θ,Z). If this instruction is not included, calculations are carried out using Cartesian coordinates.

See also: interprete (3)

Usage:
axi

3.14 Bidim_axi

Description: Keyword allowing a 2D calculation to be executed using axisymetric coordinates (R, Z). If this instruction is not included, calculations are carried out using Cartesian coordinates.

See also: interprete (3)

Usage:
bidim_axi

3.15 Bloc_b

Description: not set

See also: interprete (3)

Usage:
bloc_b {

[**rayon_bulle** *float*]
[**coeff_derive** *float*]

}
where

- **rayon_bulle** *float*: Radius of the bubbles (useful for the correlation and it is required)
- **coeff_derive** *float*: Drift coefficient (useful for the correlation and it is required)

3.16 Bloc_phases

Description: not_set

See also: interprete (3)

Usage:
bloc_phases {

[**liquide** *bloc_lecture*]
[**gaz** *bloc_lecture*]

}
where

- **liquide** *bloc_lecture* (3.17): definition of the liquid phase
- **gaz** *bloc_lecture* (3.17): definition of the gazeous phase

3.17 Bloc_lecture

Description: to read between two braces

See also: objet_lecture (33)

Usage:
bloc_lecture
where

- **bloc_lecture** *str*

3.18 Calculer_moments

Description: Calculates and prints the torque (moment of force) exerted by the fluid on each boundary in output files (.out) of the domain nom_dom.

See also: interprete (3)

Usage:
calculer_moments nom_dom mot
where

- **nom_dom** *str*: Name of domain.
- **mot** *lecture_bloc_moment_base* (3.19): Keyword.

3.19 Lecture_bloc_moment_base

Description: Auxiliary class to compute and print the moments.

See also: objet_lecture (33) calcul (3.19.1) centre_de_gravite (3.19.2)

Usage:

3.19.1 Calcul

Description: The centre of gravity will be calculated.

See also: (3.19)

Usage:
calcul

3.19.2 Centre_de_gravite

Description: To specify the centre of gravity.

See also: (3.19)

Usage:
centre_de_gravite point
where

- **point** *un_point* (3.19.3): A centre of gravity.

3.19.3 Un_point

Description: A point.

See also: objet_lecture (33)

Usage:
pos
where

- **pos** *x1 x2 (x3)*: Point coordinates.

3.20 Corriger_frontiere_periodique

Description: The Corriger_frontiere_periodique keyword is mandatory to first define the periodic boundaries, to reorder the faces and eventually fix unaligned nodes of these boundaries. Faces on one side of the periodic domain are put first, then the faces on the opposite side, in the same order. It must be run in sequential before mesh splitting.

See also: interprete (3)

Usage:
corriger_frontiere_periodique {

> **domaine** *str*
> **bord** *str*
> [**direction** *n x1 x2 ... xn*]
> [**fichier_post** *str*]

}
where

- **domaine** *str*: Name of domain.
- **bord** *str*: the name of the boundary (which must contain two opposite sides of the domain)

- **direction** *n x1 x2 ... xn*: defines the periodicity direction vector (a vector that points from one node on one side to the opposite node on the other side). This vector must be given if the automatic algorithm fails, that is:
  - when the node coordinates are not perfectly periodic
  - when the periodic direction is not aligned with the normal vector of the boundary faces
- **fichier_post** *str*: .

3.21 Create_domain_from_sous_zone

Description: This keyword fills the domain domaine_final with the subzone par_sous_zone from the domain domaine_init. It is very useful when meshing several mediums with Gmsh. Each medium will be defined as a subzone into Gmsh. A MED mesh file will be saved from Gmsh and read with Lire_Med keyword by the TRUST data file. And with this keyword, a domain will be created for each medium in the TRUST data file.

See also: interprete_geometrique_base (3.55)

Usage:
create_domain_from_sous_zone {

> **domaine_final** *str*
> **par_sous_zone** *str*
> **domaine_init** *str*

}
where

- **domaine_final** *str*: new domain in which faces are stored
- **par_sous_zone** *str*: a sub-area allowing to choose the elements
- **domaine_init** *str*: initial domain

3.22 Criteres_convergence

Description: convergence criteria

See also: interprete (3)

Usage:
aco [**inco**] [**val**] **acof**
where

- **aco** *str into ['{']: Opening curly bracket.*
- **inco** *str: Unknown (i.e: alpha, temperature, velocity and pressure)*
- **val** *float: Convergence threshold*
- **acof** *str into ['}']* : Closing curly bracket.

3.23 Debog

Description: Class to debug some differences between two TRUST versions on a same data file.
If you want to compare the results of the same code in sequential and parallel calculation, first run (mode=0) in sequential mode (the files fichier1 and fichier2 will be written first) then the second run in parallel calculation (mode=1).
During the first run (mode=0), it prints into the file DEBOG, values at different points of the code thanks to

the C++ instruction call. see for example in Noyau/Resoudre.cpp file the instruction: Debog::verifier(msg,value); Where msg is a string and value may be a double, an integer or an array.
During the second run (mode=1), it prints into a file Err_Debog.dbg the same messages than in the DEBOG file and checks if the differences between results from both codes are less than a given value (error). If not, it prints Ok else show the differences and the lines where it occured.

See also: interprete (3)

Usage:
debog pb fichier1 fichier2 seuil mode
where

- **pb** *str*: Name of the problem to debug.
- **fichier1** *str*: Name of the file where domain will be written in sequential calculation.
- **fichier2** *str*: Name of the file where faces will be written in sequential calculation.
- **seuil** *float*: Minimal value (by default 1.e-20) for the differences between the two codes.
- **mode** *int*: By default -1 (nothing is written in the different files), you will set 0 for the sequential run, and 1 for the parallel run.

3.24 {

Description: Block's beginning.

See also: interprete (3)

Usage:
{

3.25 Decoupebord

Synonymous: **decoupebord_pour_rayonnement**

Description: To subdivide the external boundary of a domain into several parts (may be useful for better accuracy when using radiation model in transparent medium). To specify the boundaries of the fine_domain_name domain to be splitted. These boundaries will be cut according the coarse mesh defined by either the keyword domaine_grossier (each boundary face of the coarse mesh coarse_domain_name will be used to group boundary faces of the fine mesh to define a new boundary), either by the keyword nb_parts_naif (each boundary of the fine mesh is splitted into a partition with nx*ny*nz elements), either by a geometric condition given by a formulae with the keyword condition_geometrique. If used, the coarse_domain_name domain should have the same boundaries name of the fine_domain_name domain.
A mesh file (ASCII format, except if binaire option is specified) named by default newgeom (or specified by the nom_fichier_sortie keyword) will be created and will contain the fine_domain_name domain with the splitted boundaries named boundary_name

See also: interprete (3)

Usage:
decoupebord {

> **domaine** *str*
> [**domaine_grossier** *str*]
> [**nb_parts_naif** *n n1 n2 ... nn*]
> [**nb_parts_geom** *n n1 n2 ... nn*]
> **bords_a_decouper** *n word1 word2 ... wordn*

[**nom_fichier_sortie** *str*]
[**condition_geometrique** *n word1 word2 ... wordn*]
[**binaire** *int*]

}
where

- **domaine** *str*
- **domaine_grossier** *str*
- **nb_parts_naif** *n n1 n2 ... nn*
- **nb_parts_geom** *n n1 n2 ... nn*
- **bords_a_decouper** *n word1 word2 ... wordn*
- **nom_fichier_sortie** *str*
- **condition_geometrique** *n word1 word2 ... wordn*
- **binaire** *int*

3.26 Decouper_bord_coincident

Description: In case of non-coincident meshes and a paroi_contact condition, run is stopped and two external files are automatically generated in VEF (connectivity_failed_boundary_name and connectivity-_failed_pb_name.med). In 2D, the keyword Decouper_bord_coincident associated to the connectivity-_failed_boundary_name file allows to generate a new coincident mesh.

See also: interprete (3)

Usage:
decouper_bord_coincident domain_name bord
where

- **domain_name** *str*: Name of domain.
- **bord** *str*: connectivity_failed_boundary_name

3.27 Dilate

Description: Keyword to multiply the whole coordinates of the geometry.

See also: interprete (3)

Usage:
dilate domain_name alpha
where

- **domain_name** *str*: Name of domain.
- **alpha** *float*: Value of dilatation coefficient.

3.28 Dimension

Description: Keyword allowing calculation dimensions to be set (2D or 3D), where dim is an integer set to 2 or 3. This instruction is mandatory.

See also: interprete (3)

Usage:

dimension dim
where

- **dim** *int into [2, 3]*: Number of dimensions.

3.29 Disable_tu

Description: Flag to disable the writing of the .TU files

See also: interprete (3)

Usage:
disable_TU

3.30 Discretiser_domaine

Description: Useful to discretize the domain domain_name (faces will be created) without defining a problem.

See also: interprete (3)

Usage:
discretiser_domaine domain_name
where

- **domain_name** *str*: Name of the domain.

3.31 Discretize

Synonymous: **discretiser**

Description: Keyword to discretise a problem problem_name according to the discretization dis.
IMPORTANT: A number of objects must be already associated (a domain, time scheme, central object) prior to invoking the Discretize (Discretiser) keyword. The physical properties of this central object must also have been read.

See also: interprete (3)

Usage:
discretize problem_name dis
where

- **problem_name** *str*: Name of problem.
- **dis** *str*: Name of the discretization object.

3.32 Distance_paroi

Description: Class to generate external file Wall_length.xyz devoted for instance, for mixing length modelling. In this file, are saved the coordinates of each element (center of gravity) of dom domain and minimum distance between this point and boundaries (specified bords) that user specifies in data file (typically, those associated to walls). A field Distance_paroi is available to post process the distance to the wall.

See also: interprete (3)

Usage:
distance_paroi dom bords format
where

- **dom** *str*: Name of domain.
- **bords** *n word1 word2 ... wordn*: Boundaries.
- **format** *str into ['binaire', 'formatte']*: Value for format may be binaire (a binary file Wall_length.xyz is written) or formatte (moreover, a formatted file Wall_length_formatted.xyz is written).

3.33 Ecrire_champ_med

Description: Keyword to write a field to MED format into a file. Useful with Homard.

See also: interprete (3)

Usage:
ecrire_champ_med nom_dom nom_chp file
where

- **nom_dom** *str*: domain name
- **nom_chp** *str*: field name
- **file** *str*: file name

3.34 Ecrire_fichier_formatte

Description: Keyword to write the object of name name_obj to a file filename in ASCII format.

See also: ecrire_fichier_bin (3.121)

Usage:
ecrire_fichier_formatte name_obj filename
where

- **name_obj** *str*: Name of the object to be written.
- **filename** *str*: Name of the file.

3.35 Ecriturelecturespecial

Description: Class to write or not to write a .xyz file on the disk at the end of the calculation.

See also: interprete (3)

Usage:
ecriturelecturespecial type
where

- **type** *str*: If set to 0, no xyz file is created. If set to EFichierBin, it uses prior 1.7.0 way of reading xyz files (now LecFicDiffuseBin). If set to EcrFicPartageBin, it uses prior 1.7.0 way of writing xyz files (now EcrFicPartageMPIIO).

3.36 Espece

Description: not_set

See also: interprete (3)

Usage:
espece {

> **mu** *champ_base*
> **cp** *champ_base*
> **masse_molaire** *float*

}
where

- **mu** *champ_base* (14.1): Species dynamic viscosity value (kg.m-1.s-1).
- **cp** *champ_base* (14.1): Species specific heat value (J.kg-1.K-1).
- **masse_molaire** *float*: Species molar mass.

3.37 Execute_parallel

Description: This keyword allows to run several computations in parallel on processors allocated to TRUST. The set of processors is split in N subsets and each subset will read and execute a different data file. Error messages usualy written to stderr and stdout are redirected to .log files (journaling must be activated).

See also: interprete (3)

Usage:
execute_parallel {

> **liste_cas** *n word1 word2 ... wordn*
> [**nb_procs** *n n1 n2 ... nn*]

}
where

- **liste_cas** *n word1 word2 ... wordn*: N datafile1 ... datafileN. datafileX the name of a TRUST data file without the .data extension.
- **nb_procs** *n n1 n2 ... nn*: nb_procs is the number of processors needed to run each data file. If not given, TRUST assumes that computations are sequential.

3.38 Export

Description: Class to make the object have a global range, if not its range will apply to the block only (the associated object will be destroyed on exiting the block).

See also: interprete (3)

Usage:
export

3.39 Extract_2d_from_3d

Description: Keyword to extract a 2D mesh by selecting a boundary of the 3D mesh. To generate a 2D axisymmetric mesh prefer Extract_2Daxi_from_3D keyword.

See also: interprete (3) extract_2daxi_from_3d (3.40)

Usage:
extract_2d_from_3d dom3D bord dom2D
where

- **dom3D** *str*: Domain name of the 3D mesh
- **bord** *str*: Boundary name. This boundary becomes the new 2D mesh and all the boundaries, in 3D, attached to the selected boundary, give their name to the new boundaries, in 2D.
- **dom2D** *str*: Domain name of the new 2D mesh

3.40 Extract_2daxi_from_3d

Description: Keyword to extract a 2D axisymetric mesh by selecting a boundary of the 3D mesh.

See also: extract_2d_from_3d (3.39)

Usage:
extract_2daxi_from_3d dom3D bord dom2D
where

- **dom3D** *str*: Domain name of the 3D mesh
- **bord** *str*: Boundary name. This boundary becomes the new 2D mesh and all the boundaries, in 3D, attached to the selected boundary, give their name to the new boundaries, in 2D.
- **dom2D** *str*: Domain name of the new 2D mesh

3.41 Extraire_domaine

Description: Keyword to create a new domain built with the domain elements of the pb_name problem verifying the two conditions given by Condition_elements. The problem pb_name should have been discretized.

Keyword Discretize should have already been used to read the object.
See also: interprete (3)

Usage:
extraire_domaine {

> **domaine** *str*
> **probleme** *str*
> [**condition_elements** *str*]
> [**sous_zone** *str*]

}
where

- **domaine** *str*: Domain in which faces are saved
- **probleme** *str*: Problem from which faces should be extracted
- **condition_elements** *str*
- **sous_zone** *str*

3.42 Extraire_plan

Description: This keyword extracts a plane mesh named domain_name (this domain should have been declared before) from the mesh of the pb_name problem. The plane can be either a triangle (defined by the keywords Origine, Point1, Point2 and Triangle), either a regular quadrangle (with keywords Origine, Point1 and Point2), or either a generalized quadrangle (with keywords Origine, Point1, Point2, Point3). The keyword Epaisseur specifies the thickness of volume around the plane which contains the faces of the extracted mesh. The keyword via_extraire_surface will create a plan and use Extraire_surface algorithm. Inverse_condition_element keyword then will be used in the case where the plane is a boundary not well oriented, and avec_certains_bords_pour_extraire_surface is the option related to the Extraire_surface option named avec_certains_bords.

Keyword Discretize should have already been used to read the object.
See also: interprete (3)

Usage:
extraire_plan {

> **domaine** *str*
> **probleme** *str*
> **epaisseur** *float*
> **origine** *n x1 x2 ... xn*
> **point1** *n x1 x2 ... xn*
> **point2** *n x1 x2 ... xn*
> [**point3** *n x1 x2 ... xn*]
> [**triangle**]
> [**via_extraire_surface**]
> [**inverse_condition_element**]
> [**avec_certains_bords_pour_extraire_surface** *n word1 word2 ... wordn*]

}
where

- **domaine** *str*: domain_namme
- **probleme** *str*: pb_name
- **epaisseur** *float*
- **origine** *n x1 x2 ... xn*
- **point1** *n x1 x2 ... xn*
- **point2** *n x1 x2 ... xn*
- **point3** *n x1 x2 ... xn*
- **triangle**
- **via_extraire_surface**
- **inverse_condition_element**
- **avec_certains_bords_pour_extraire_surface** *n word1 word2 ... wordn*

3.43 Extraire_surface

Description: This keyword extracts a surface mesh named domain_name (this domain should have been declared before) from the mesh of the pb_name problem. The surface mesh is defined by one or two conditions. The first condition is about elements with Condition_elements. For example: Condition_elements x*x+y*y+z*z<1
Will define a surface mesh with external faces of the mesh elements inside the sphere of radius 1 located at (0,0,0). The second condition Condition_faces is useful to give a restriction.
By default, the faces from the boundaries are not added to the surface mesh excepted if option avec_les_bords is given (all the boundaries are added), or if the option avec_certains_bords is used to add only

some boundaries.

Keyword Discretize should have already been used to read the object.
See also: interprete (3)

Usage:
extraire_surface {

domaine *str*
probleme *str*
[**condition_elements** *str*]
[**condition_faces** *str*]
[**avec_les_bords**]
[**avec_certains_bords** *n word1 word2 ... wordn*]

}
where

- **domaine** *str*: Domain in which faces are saved
- **probleme** *str*: Problem from which faces should be extracted
- **condition_elements** *str*
- **condition_faces** *str*
- **avec_les_bords**
- **avec_certains_bords** *n word1 word2 ... wordn*

3.44 Extrudebord

Description: Class to generate an extruded mesh from a boundary of a tetrahedral or an hexahedral mesh.
Warning: If the initial domain is a tetrahedral mesh, the boundary will be moved in the XY plane then extrusion will be applied (you should maybe use the Transformer keyword on the final domain to have the domain you really want). You can use the keyword Ecrire_Fichier_Meshtv to generate a meshtv file to visualize your initial and final meshes.
This keyword can be used for example to create a periodic box extracted from a boundary of a tetrahedral or a hexaedral mesh. This periodic box may be used then to engender turbulent inlet flow condition for the main domain.
Note that ExtrudeBord in VEF generates 3 or 14 tetrahedra from extruded prisms.

See also: interprete (3)

Usage:
extrudebord {

domaine_init *str*
direction *x1 x2 (x3)*
nb_tranches *int*
domaine_final *str*
nom_bord *str*
[**hexa_old**]
[**trois_tetra**]
[**vingt_tetra**]
[**sans_passer_par_le2d** *int*]

}
where

- **domaine_init** *str*: Initial domain with hexaedras or tetrahedras.
- **direction** *x1 x2 (x3)*: Directions for the extrusion.
- **nb_tranches** *int*: Number of elements in the extrusion direction.
- **domaine_final** *str*: Extruded domain.
- **nom_bord** *str*: Name of the boundary of the initial domain where extrusion will be applied.
- **hexa_old** : Old algorithm for boundary extrusion from a hexahedral mesh.
- **trois_tetra** : To extrude in 3 tetrahedras instead of 14 tetrahedras.
- **vingt_tetra** : To extrude in 20 tetrahedras instead of 14 tetrahedras.
- **sans_passer_par_le2d** *int*: Only for non-regression

3.45 Extrudeparoi

Description: Keyword dedicated in 3D (VEF) to create prismatic layer at wall. Each prism is cut into 3 tetraedra.

See also: interprete (3)

Usage:
extrudeparoi {

> **domaine** *str*
> **nom_bord** *str*
> [**epaisseur** *n x1 x2 ... xn*]
> [**critere_absolu** *int*]
> [**projection_normale_bord**]

}
where

- **domaine** *str*: Name of the domain.
- **nom_bord** *str*: Name of the (no-slip) boundary for creation of prismatic layers.
- **epaisseur** *n x1 x2 ... xn*: n r1 r2 .... rn : (relative or absolute) width for each layer.
- **critere_absolu** *int*: relative (0, the default) or absolute (1) width for each layer.
- **projection_normale_bord** : keyword to project layers on the same plane that contiguous boundaries. defaut values are : epaisseur_relative 1 0.5 projection_normale_bord 1

3.46 Extruder

Description: Class to create a 3D tetrahedral/hexahedral mesh (a prism is cut in 14) from a 2D triangular/quadrangular mesh.

See also: interprete (3) extruder_en3 (3.49)

Usage:
extruder {

> **domaine** *str*
> **direction** *troisf*
> **nb_tranches** *int*

}
where

- **domaine** *str*: Name of the domain.
- **direction** *troisf* (3.47): Direction of the extrude operation.
- **nb_tranches** *int*: Number of elements in the extrusion direction.

3.47 Troisf

Description: Auxiliary class to extrude.

See also: objet_lecture (33)

Usage:
lx **ly** **lz**
where

- **lx** *float*: X direction of the extrude operation.
- **ly** *float*: Y direction of the extrude operation.
- **lz** *float*: Z direction of the extrude operation.

3.48 Extruder_en20

Description: It does the same task as Extruder except that a prism is cut into 20 tetraedra instead of 3. The name of the boundaries will be devant (front) and derriere (back). But you can change these names with the keyword RegroupeBord.

See also: interprete (3)

Usage:
extruder_en20 {

> **domaine** *str*
> [**direction** *troisf*]
> **nb_tranches** *int*

}
where

- **domaine** *str*: Name of the domain.
- **direction** *troisf* (3.47): 0 Direction of the extrude operation.
- **nb_tranches** *int*: Number of elements in the extrusion direction.

3.49 Extruder_en3

Description: Class to create a 3D tetrahedral/hexahedral mesh (a prism is cut in 3) from a 2D triangular/quadrangular mesh. The names of the boundaries (by default, devant (front) and derriere (back)) may be edited by the keyword nom_cl_devant and nom_cl_derriere. If NULL is written for nom_cl, then no boundary condition is generated at this place.
Recommendation : to ensure conformity between meshes (in case of fluid/solid coupling) it is recommended to extrude all the domains at the same time.

See also: extruder (3.46)

Usage:
extruder_en3 {

> **domaine** *n word1 word2 ... wordn*
> [**nom_cl_devant** *str*]
> [**nom_cl_derriere** *str*]
> **direction** *troisf*

nb_tranches *int*

}
where

- **domaine** *n word1 word2 ... wordn*: List of the domains
- **nom_cl_devant** *str*: New name of the first boundary.
- **nom_cl_derriere** *str*: New name of the second boundary.
- **direction** *troisf* (3.47) for inheritance: Direction of the extrude operation.
- **nb_tranches** *int* for inheritance: Number of elements in the extrusion direction.

3.50 End

Synonymous: **fin**

Description: Keyword which must complete the data file. The execution of the data file stops when reaching this keyword.

See also: interprete (3)

Usage:
end

3.51 }

Description: Block's end.

See also: interprete (3)

Usage:
}

3.52 Imprimer_flux

Description: This keyword prints the flux per face at the specified domain boundaries in the data set. The fluxes are written to the .face files at a frequency defined by dt_impr, the evaluation printing frequency (refer to time scheme keywords). By default, fluxes are incorporated onto the edges before being displayed.

See also: interprete (3) imprimer_flux_sum (3.53)

Usage:
imprimer_flux domain_name noms_bord
where

- **domain_name** *str*: Name of the domain.
- **noms_bord** *bloc_lecture* (3.17): List of boundaries, for ex: { Bord1 Bord2 }

3.53 Imprimer_flux_sum

Description: This keyword prints the sum of the flux per face at the domain boundaries defined by the user in the data set. The fluxes are written into the .out files at a frequency defined by dt_impr, the evaluation printing frequency (refer to time scheme keywords).

See also: imprimer_flux (3.52)

Usage:
imprimer_flux_sum domain_name noms_bord
where

- **domain_name** *str*: Name of the domain.
- **noms_bord** *bloc_lecture* (3.17): List of boundaries, for ex: { Bord1 Bord2 }

3.54 Integrer_champ_med

Description: his keyword is used to calculate a flow rate from a velocity MED field read before. The method is either debit_total to calculate the flow rate on the whole surface, either integrale_en_z to calculate flow rates between z=zmin and z=zmax on nb_tranche surfaces. The output file indicates first the flow rate for the whole surface and then lists for each tranche : the height z, the surface average value, the surface area and the flow rate. For the debit_total method, only one tranche is considered.
file :z Sum(u.dS)/Sum(dS) Sum(dS) Sum(u.dS)

See also: interprete (3)

Usage:
integrer_champ_med {

> **champ_med** *str*
> **methode** *str into ['integrale_en_z', 'debit_total']*
> [**zmin** *float*]
> [**zmax** *float*]
> [**nb_tranche** *int*]
> [**fichier_sortie** *str*]

}
where

- **champ_med** *str*
- **methode** *str into ['integrale_en_z', 'debit_total']*: to choose between the integral following z or over the entire height (debit_total corresponds to zmin=-DMAXFLOAT, ZMax=DMAXFLOAT, nb_tranche=1)
- **zmin** *float*
- **zmax** *float*
- **nb_tranche** *int*
- **fichier_sortie** *str*: name of the output file, by default: integrale.

3.55 Interprete_geometrique_base

Description: Class for interpreting a data file

See also: interprete (3) create_domain_from_sous_zone (3.21)

Usage:
interprete_geometrique_base

3.56 Lata_to_med

Description: To convert results file written with LATA format to MED file. Warning: Fields located on faces are not supported yet.

See also: interprete (3)

Usage:
lata_to_med [**format**] **file** **file_med**
where

- **format** *format_lata_to_med* (3.57): generated file post_med.data use format (MED or LATA or LML keyword).
- **file** *str*: LATA file to convert to the new format.
- **file_med** *str*: Name of the MED file.

3.57 Format_lata_to_med

Description: not_set

See also: objet_lecture (33)

Usage:
mot [**format**]
where

- **mot** *str into ['format_post_sup']*
- **format** *str into ['lml', 'lata', 'lata_v1', 'lata_v2', 'med']*: generated file post_med.data use format (MED or LATA or LML keyword).

3.58 Lata_to_other

Description: To convert results file written with LATA format to MED or LML format. Warning: Fields located at faces are not supported yet.

See also: interprete (3)

Usage:
lata_to_other [**format**] **file** **file_post**
where

- **format** *str into ['lml', 'lata', 'lata_v1', 'lata_v2', 'med']*: Results format (MED or LATA or LML keyword).
- **file** *str*: LATA file to convert to the new format.
- **file_post** *str*: Name of file post.

3.59 Lire_ideas

Description: Read a geom in a unv file. 3D tetra mesh elements only may be read by TRUST.

See also: interprete (3)

Usage:
lire_ideas **nom_dom** **file**
where

- **nom_dom** *str*: Name of domain.
- **file** *str*: Name of file.

3.60 Mailler

Description: The Mailler (Mesh) interpretor allows a Domain type object domaine to be meshed with objects objet_1, objet_2, etc...

See also: interprete (3)

Usage:
mailler domaine bloc
where

- **domaine** *str*: Name of domain.
- **bloc** *list_bloc_mailler* (3.61): Instructions to mesh.

3.61 List_bloc_mailler

Description: List of block mesh.

See also: listobj (32.5)

Usage:
{ object1 , object2 .... }
list of *mailler_base* (3.61.1) separeted with ,

3.61.1 Mailler_base

Description: Basic class to mesh.

See also: objet_lecture (33) pave (3.61.2) epsilon (3.61.12) domain (3.61.13)

Usage:

3.61.2 Pave

Description: Class to create a pave (block) with boundaries.

See also: mailler_base (3.61.1)

Usage:
pave name bloc list_bord
where

- **name** *str*: Name of the pave (block).
- **bloc** *bloc_pave* (3.61.3): Definition of the pave (block).
- **list_bord** *list_bord* (3.61.4): Domain boundaries definition.

3.61.3 Bloc_pave

Description: Class to create a pave.

See also: objet_lecture (33)

Usage:
{

[**Origine** *x1 x2 (x3)*]
[**longueurs** *x1 x2 (x3)*]
[**nombre_de_noeuds** *n1 n2 (n3)*]
[**facteurs** *x1 x2 (x3)*]
[**symx**]
[**symy**]
[**symz**]
[**xtanh** *float*]
[**xtanh_dilatation** *int into [-1, 0, 1]*]
[**xtanh_taille_premiere_maille** *float*]
[**ytanh** *float*]
[**ytanh_dilatation** *int into [-1, 0, 1]*]
[**ytanh_taille_premiere_maille** *float*]
[**ztanh** *float*]
[**ztanh_dilatation** *int into [-1, 0, 1]*]
[**ztanh_taille_premiere_maille** *float*]

}
where

- **Origine** *x1 x2 (x3)*: Keyword to define the pave (block) origin, that is to say one of the 8 block points (or 4 in a 2D coordinate system).
- **longueurs** *x1 x2 (x3)*: Keyword to define the block dimensions, that is to say knowing the origin, length along the axes.
- **nombre_de_noeuds** *n1 n2 (n3)*: Keyword to define the discretization (nodenumber) in each direction.
- **facteurs** *x1 x2 (x3)*: Keyword to define stretching factors for mesh discretization in each direction. This is a real number which must be positive (by default 1.0). A stretching factor other than 1 allows refinement on one edge in one direction.
- **symx** : Keyword to define a block mesh that is symmetrical with respect to the YZ plane (respectively Y-axis in 2D) passing through the block centre.
- **symy** : Keyword to define a block mesh that is symmetrical with respect to the XZ plane (respectively X-axis in 2D) passing through the block centre.
- **symz** : Keyword defining a block mesh that is symmetrical with respect to the XY plane passing through the block centre.
- **xtanh** *float*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the X-direction.
- **xtanh_dilatation** *int into [-1, 0, 1]*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the X-direction. xtanh_dilatation: The value may be -1,0,1 (0 by default): 0: coarse mesh at the middle of the channel and smaller near the walls -1: coarse mesh at the left side of the channel and smaller at the right side 1: coarse mesh at the right side of the channel and smaller near the left side of the channel.
- **xtanh_taille_premiere_maille** *float*: Size of the first cell of the mesh with tanh (hyperbolic tangent) variation in the X-direction.
- **ytanh** *float*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the Y-direction.
- **ytanh_dilatation** *int into [-1, 0, 1]*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the Y-direction. ytanh_dilatation: The value may be -1,0,1 (0 by default): 0: coarse

mesh at the middle of the channel and smaller near the walls -1: coarse mesh at the bottom of the channel and smaller near the top 1: coarse mesh at the top of the channel and smaller near the bottom.

- **ytanh_taille_premiere_maille** *float*: Size of the first cell of the mesh with tanh (hyperbolic tangent) variation in the Y-direction.
- **ztanh** *float*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the Z-direction.
- **ztanh_dilatation** *int into [-1, 0, 1]*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the Z-direction. tanh_dilatation: The value may be -1,0,1 (0 by default): 0: coarse mesh at the middle of the channel and smaller near the walls -1: coarse mesh at the back of the channel and smaller near the front 1: coarse mesh at the front of the channel and smaller near the back.
- **ztanh_taille_premiere_maille** *float*: Size of the first cell of the mesh with tanh (hyperbolic tangent) variation in the Z-direction.

3.61.4 List_bord

Description: The block sides.

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *bord_base* (3.61.5)

3.61.5 Bord_base

Description: Basic class for block sides. Block sides that are neither edges nor connectors are not specified. The duplicate nodes of two blocks in contact are automatically recognized and deleted.

See also: objet_lecture (33) bord (3.61.6) raccord (3.61.10) internes (3.61.11)

Usage:

3.61.6 Bord

Description: The block side is not in contact with another block and boundary conditions are applied to it.

See also: bord_base (3.61.5)

Usage:
bord nom defbord
where

- **nom** *str*: Name of block side.
- **defbord** *defbord* (3.61.7): Definition of block side.

3.61.7 Defbord

Description: Class to define an edge.

See also: objet_lecture (33) defbord_2 (3.61.8) defbord_3 (3.61.9)

Usage:

3.61.8 Defbord_2

Description: 1-D edge (straight line) in the 2-D space.

See also: (3.61.7)

Usage:
dir eq pos pos2_min inf1 dir2 inf2 pos2_max
where

- **dir** *str into ['X', 'Y']*: Edge is perpendicular to this direction.
- **eq** *str into ['=']*: Equality sign.
- **pos** *float*: Position value.
- **pos2_min** *float*: Minimal value.
- **inf1** *str into ['<=']*: Less than or equal to sign.
- **dir2** *str into ['X', 'Y']*: Edge is parallel to this direction.
- **inf2** *str into ['<=']*: Less than or equal to sign.
- **pos2_max** *float*: Maximal value.

3.61.9 Defbord_3

Description: 2-D edge (plane) in the 3-D space.

See also: (3.61.7)

Usage:
dir eq pos pos2_min inf1 dir2 inf2 pos2_max pos3_min inf3 dir3 inf4 pos3_max
where

- **dir** *str into ['X', 'Y', 'Z']*: Edge is perpendicular to this direction.
- **eq** *str into ['=']*: Equality sign.
- **pos** *float*: Position value.
- **pos2_min** *float*: Minimal value.
- **inf1** *str into ['<=']*: Less than or equal to sign.
- **dir2** *str into ['X', 'Y']*: Edge is parallel to this direction.
- **inf2** *str into ['<=']*: Less than or equal to sign.
- **pos2_max** *float*: Maximal value.
- **pos3_min** *float*: Minimal value.
- **inf3** *str into ['<=']*: Less than or equal to sign.
- **dir3** *str into ['Y', 'Z']*: Edge is parallel to this direction.
- **inf4** *str into ['<=']*: Less than or equal to sign.
- **pos3_max** *float*: Maximal value.

3.61.10 Raccord

Description: The block side is in contact with the block of another domain (case of two coupled problems).

See also: bord_base (3.61.5)

Usage:
raccord type1 type2 nom defbord
where

- **type1** *str into ['local', 'distant']*: Contact type.

- **type2** *str into ['homogene']*: Contact type.
- **nom** *str*: Name of block side.
- **defbord** *defbord* (3.61.7): Definition of block side.

3.61.11 Internes

Description: To indicate that the block has a set of internal faces (these faces will be duplicated automatically by the program and will be processed in a manner similar to edge faces).
Two boundaries with the same boundary conditions may have the same name (whether or not they belong to the same block).
The keyword Internes (Internal) must be used to execute a calculation with plates, followed by the equation of the surface area covered by the plates.

See also: bord_base (3.61.5)

Usage:
internes nom defbord
where

- **nom** *str*: Name of block side.
- **defbord** *defbord* (3.61.7): Definition of block side.

3.61.12 Epsilon

Description: Two points will be confused if the distance between them is less than eps. By default, eps is set to 1e-12. The keyword Epsilon allows an alternative value to be assigned to eps.

See also: mailler_base (3.61.1)

Usage:
epsilon eps
where

- **eps** *float*: New value of precision.

3.61.13 Domain

Description: Class to reuse a domain.

See also: mailler_base (3.61.1)

Usage:
domain domain_name
where

- **domain_name** *str*: Name of domain.

3.62 Maillerparallel

Description: creates a parallel distributed hexaedral mesh of a parallelipipedic box. It is equivalent to creating a mesh with a single Pave, splitting it with Decouper and reloading it in parallel with Scatter. It only works in 3D at this time. It can also be used for a sequential computation (with all NPARTS=1)}

See also: interprete (3)

Usage:
maillerparallel {

> **domain** *str*
> **nb_nodes** *n n1 n2 ... nn*
> **splitting** *n n1 n2 ... nn*
> **ghost_thickness** *int*
> [**perio_x**]
> [**perio_y**]
> [**perio_z**]
> [**function_coord_x** *str*]
> [**function_coord_y** *str*]
> [**function_coord_z** *str*]
> [**file_coord_x** *str*]
> [**file_coord_y** *str*]
> [**file_coord_z** *str*]
> [**boundary_xmin** *str*]
> [**boundary_xmax** *str*]
> [**boundary_ymin** *str*]
> [**boundary_ymax** *str*]
> [**boundary_zmin** *str*]
> [**boundary_zmax** *str*]

}
where

- **domain** *str*: the name of the domain to mesh (it must be an empty domain object).
- **nb_nodes** *n n1 n2 ... nn*: dimension defines the spatial dimension (currently only dimension=3 is supported), and nX, nY and nZ defines the total number of nodes in the mesh in each direction.
- **splitting** *n n1 n2 ... nn*: dimension is the spatial dimension and npartsX, npartsY and npartsZ are the number of parts created. The product of the number of parts must be equal to the number of processors used for the computation.
- **ghost_thickness** *int*: he number of ghost cells (equivalent to the epaisseur_joint parameter of Decouper.
- **perio_x** : change the splitting method to provide a valid mesh for periodic boundary conditions.
- **perio_y** : change the splitting method to provide a valid mesh for periodic boundary conditions.
- **perio_z** : change the splitting method to provide a valid mesh for periodic boundary conditions.
- **function_coord_x** *str*: By default, the meshing algorithm creates nX nY nZ coordinates ranging between 0 and 1 (eg a unity size box). If function_coord_x} is specified, it is used to transform the [0,1] segment to the coordinates of the nodes. funcX must be a function of the x variable only.
- **function_coord_y** *str*: like function_coord_x for y
- **function_coord_z** *str*: like function_coord_x for z
- **file_coord_x** *str*: Keyword to read the Nx floating point values used as nodes coordinates in the file.

- **file_coord_y** *str*: idem file_coord_x for y
- **file_coord_z** *str*: idem file_coord_x for z

- **boundary_xmin** *str*: the name of the boundary at the minimum X direction. If it not provided, the default boundary names are xmin, xmax, ymin, ymax, zmin and zmax. If the mesh is periodic in a given direction, only the MIN boundary name is used, for both sides of the box.
- **boundary_xmax** *str*
- **boundary_ymin** *str*
- **boundary_ymax** *str*
- **boundary_zmin** *str*
- **boundary_zmax** *str*

3.63 Modif_bord_to_raccord

Description: Keyword to convert a boundary of domain_name domain of kind Bord to a boundary of kind Raccord (named boundary_name). It is useful when using meshes with boundaries of kind Bord defined and to run a coupled calculation.

See also: interprete (3)

Usage:
modif_bord_to_raccord domaine nom_bord
where

- **domaine** *str*: Name of domain
- **nom_bord** *str*: Name of the boundary to transform.

3.64 Moyenne_volumique

Description: This keyword should be used after Resoudre keyword. It computes the convolution product of one or more fields with a given filtering function.

See also: interprete (3)

Usage:
moyenne_volumique {

nom_pb *str*
nom_domaine *str*
noms_champs *n word1 word2 ... wordn*
[**nom_fichier_post** *str*]
[**format_post** *str*]
[**localisation** *str into ['elem', 'som']*]
fonction_filtre *bloc_lecture*

}
where

- **nom_pb** *str*: name of the problem where the source fields will be searched.
- **nom_domaine** *str*: name of the destination domain (for example, it can be a coarser mesh, but for optimal performance in parallel, the domain should be split with the same algorithm as the computation mesh, eg, same tranche parameters for example)
- **noms_champs** *n word1 word2 ... wordn*: name of the source fields (these fields must be accessible from the postraitement) N source_field1 source_field2 ... source_fieldN
- **nom_fichier_post** *str*: indicates the filename where the result is written
- **format_post** *str*: gives the fileformat for the result (by default : lata)

- **localisation** *str into ['elem', 'som']*: indicates where the convolution product should be computed: either on the elements or on the nodes of the destination domain.
- **fonction_filtre** *bloc_lecture* (3.17): to specify the given filter
  Fonction_filtre {
  type filter_type
  demie-largeur l
  [omega w]
  [expression string]
  }

  type filter_type : This parameter specifies the filtering function. Valid filter_type are:
  Boite is a box filter, $f(x,y,z) = (abs(x) < l) * (abs(y) < l) * (abs(z) < l)/(8l^3)$
  Chapeau is a hat filter (product of hat filters in each direction) centered on the origin, the half-width of the filter being l and its integral being 1.
  Quadra is a 2nd order filter.
  Gaussienne is a normalized gaussian filter of standard deviation sigma in each direction (all field elements outside a cubic box defined by clipping_half_width are ignored, hence, taking clipping-_half_width=2.5*sigma yields an integral of 0.99 for a uniform unity field).
  Parser allows a user defined function of the x,y,z variables. All elements outside a cubic box defined by clipping_half_width are ignored. The parser is much slower than the equivalent c++ coded function...

  demie-largeur l : This parameter specifies the half width of the filter
  [omega w] : This parameter must be given for the gaussienne filter. It defines the standard deviation of the gaussian filter.
  [expression string] : This parameter must be given for the parser filter type. This expression will be interpreted by the math parser with the predefined variables x, y and z.

3.65 Nettoiepasnoeuds

Description: Keyword NettoiePasNoeuds does not delete useless nodes (nodes without elements) from a domain.

See also: interprete (3)

Usage:
nettoiepasnoeuds domain_name
where

- **domain_name** *str*: Name of domain.

3.66 Option_vdf

Description: Class of VDF options.

See also: interprete (3)

Usage:
option_vdf {

  [**traitement_coins** *str into ['oui', 'non']*]
  [**p_imposee_aux_faces** *str into ['oui', 'non']*]

}
where

- **traitement_coins** *str into ['oui', 'non']*: Treatment of corners (yes or no). This option modifies slightly the calculations at the outlet of the plane channel. It supposes that the boundary continues after channel outlet (i.e. velocity vector remains parallel to the boundary).
- **p_imposee_aux_faces** *str into ['oui', 'non']*: Pressure imposed at the faces (yes or no).

3.67 Orientefacesbord

Description: Keyword to modify the order of the boundary vertices included in a domain, such that the surface normals are outer pointing.

See also: interprete (3)

Usage:
orientefacesbord domain_name
where

- **domain_name** *str*: Name of domain.

3.68 Partition

Synonymous: **decouper**

Description: Class for parallel calculation to cut a domain for each processor. By default, this keyword is commented in the reference test cases.

See also: interprete (3)

Usage:
partition domaine bloc_decouper
where

- **domaine** *str*: Name of the domain to be cut.
- **bloc_decouper** *bloc_decouper* (3.69): Description how to cut a domain.

3.69 Bloc_decouper

Description: Auxiliary class to cut a domain.

See also: objet_lecture (33)

Usage:
{

[**Partition_tool|partitionneur** *partitionneur_deriv*]
[**larg_joint** *int*]
[**zones_name|nom_zones** *str*]
[**ecrire_decoupage** *str*]
[**ecrire_lata** *str*]
[**nb_parts_tot** *int*]
[**periodique** *n word1 word2 ... wordn*]

[**reorder** *int*]
[**single_hdf**]
[**print_more_infos** *int*]

}
where

- **Partition_tool|partitionneur** *partitionneur_deriv* (23): Defines the partitionning algorithm (the effective C++ object used is 'Partitionneur_ALGORITHM_NAME').
- **larg_joint** *int*: This keyword specifies the thickness of the virtual ghost zone (data known by one processor though not owned by it). The default value is 1 and is generally correct for all algorithms except the QUICK convection scheme that require a thickness of 2. Since the 1.5.5 version, the VEF discretization imply also a thickness of 2 (except VEF P0). Any non-zero positive value can be used, but the amount of data to store and exchange between processors grows quickly with the thickness.
- **zones_name|nom_zones** *str*: Name of the files containing the different partition of the domain. The files will be :
name_0001.Zones
name_0002.Zones
...
name_000n.Zones. If this keyword is not specified, the geometry is not written on disk (you might just want to generate a 'ecrire_decoupage' or 'ecrire_lata').
- **ecrire_decoupage** *str*: After having called the partitionning algorithm, the resulting partition is written on disk in the specified filename. See also partitionneur Fichier_Decoupage. This keyword is useful to change the partition numbers: first, you write the partition into a file with the option ecrire_decoupage. This file contains the zone number for each element's mesh. Then you can easily permute zone numbers in this file. Then read the new partition to create the .Zones files with the Fichier_Decoupage keyword.
- **ecrire_lata** *str*
- **nb_parts_tot** *int*: Keyword to generates N .Zone files, instead of the default number M obtained after the partitionning algorithm. N must be greater or equal to M. This option might be used to perform coupled parallel computations. Supplemental empty zones from M to N-1 are created. This keyword is used when you want to run a parallel calculation on several domains with for example, 2 processors on a first domain and 10 on the second domain because the first domain is very small compare to second one. You will write Nb_parts 2 and Nb_parts_tot 10 for the first domain and Nb_parts 10 for the second domain.
- **periodique** *n word1 word2 ... wordn*: N BOUNDARY_NAME_1 BOUNDARY_NAME_2 ... : N is the number of boundary names given. Periodic boundaries must be declared by this method. The partitionning algorithm will ensure that facing nodes and faces in the periodic boundaries are located on the same processor.
- **reorder** *int*: If this option is set to 1 (0 by default), the partition is renumbered in order that the processes which communicate the most are nearer on the network. This may slighlty improves parallel performance.
- **single_hdf** : Optional keyword to enable you to write the partitioned zones in a single file in hdf5 format.
- **print_more_infos** *int*: If this option is set to 1 (0 by default), print infos about number of remote elements (ghosts) and additional infos about the quality of partitionning. Warning, it slows down the cutting operations.

3.70 Partition_multi

Synonymous: **decouper_multi**

Description: allows to partition multiple domains while accounting for connections via Raccords (allows for easier implementation of thermique_monolithique in parallel). By default, this keyword is commented in the reference test cases.

See also: interprete (3)

Usage:
partition_multi aco domaine1 dom blocdecoupdom1 domaine2 dom2 blocdecoupdom2 raccords blocracc acof
where

- **aco** *str into ['{']: Opening curly bracket.*
- **domaine1** *str into ['domaine']: not set.*
- **dom** *str: Name of the first domain to be cut.*
- **blocdecoupdom1** *bloc_decouper (3.69): Partition bloc for the first domain.*
- **domaine2** *str into ['domaine']: not set.*
- **dom2** *str: Name of the second domain to be cut.*
- **blocdecoupdom2** *bloc_decouper (3.69): Partition bloc for the second domain.*
- **raccords** *str into ['raccords']: not set.*
- **blocracc** *bloc_lecture (3.17): Indicates the joints between both domains. The syntax is { dom1 bord1 dom2 bord2 ... }*
- **acof** *str into ['}']* : Closing curly bracket.

3.71 Phases

Description: Declare the phases that will be considered

See also: interprete (3)

Usage:
phases problem_name phases_def
where

- **problem_name** *str*: Name of problem.
- **phases_def** *bloc_phases* (3.16): bloc to define phases

3.72 Pilote_icoco

Description: not_set

See also: interprete (3)

Usage:
pilote_icoco {

> **pb_name** *str*
> **main** *str*

}
where

- **pb_name** *str*
- **main** *str*

3.73 Polyedriser

Description: cast hexahedra into polyhedra so that the indexing of the mesh vertices is compatible with PolyMAC discretization. Must be used in PolyMAC discretization if a hexahedral mesh has been produced with TRUST's internal mesh generator.

See also: interprete (3)

Usage:
polyedriser domain_name
where

- **domain_name** *str*: Name of domain.

3.74 Porosites

Description: To define the volume porosity and surface porosity that are uniform in every direction in space on a sub-area.
Porosity was only usable in VDF discretization, and now available for VEF P1NC/P0.
Observations :
- Surface porosity values must be given in every direction in space (set this value to 1 if there is no porosity),
- Prior to defining porosity, the problem must have been discretized.
Can 't be used in VEF discretization, use Porosites_champ instead.

See also: interprete (3)

Usage:
porosites pb sous_zone bloc
where

- **pb** *str*: Name of the problem to which the sub-area is attached.
- **sous_zone** *str*: Name of the sub-area to which porosity are allocated.
- **bloc** *bloc_lecture_poro* (3.75): Surface and volume porosity values.

3.75 Bloc_lecture_poro

Description: Surface and volume porosity values.

See also: objet_lecture (33)

Usage:
{

> **volumique** *float*
> **surfacique** *n x1 x2 ... xn*

}
where

- **volumique** *float*: Volume porosity value.
- **surfacique** *n x1 x2 ... xn*: Surface porosity values (in X, Y, Z directions).

3.76 Porosites_champ

Description: The porosity is given at each element and the porosity at each face, Psi(face), is calculated by the average of the porosities of the two neighbour elements Psi(elem1), Psi(elem2) : Psi(face)=2/(1/Psi(elem1)+1/Psi(elem2)).

Keyword Discretize should have already been used to read the object.
See also: interprete (3)

Usage:
porosites_champ pb ch
where

- **pb** *str*: Name of the problem to which the sub-area is attached.
- **ch** *champ_base* (14.1): field used to define the porosity field

3.77 Postraiter_domaine

Description: To write one or more domains in a file with a specified format (MED,LML,LATA).

See also: interprete (3)

Usage:
postraiter_domaine {

> **format** *str into ['lml', 'lata', 'lata_v1', 'lata_v2', 'med']*
> [**file|fichier** *str*]
> [**domaine** *str*]
> [**domaines** *bloc_lecture*]
> [**joints_non_postraites** *int into [0, 1]*]
> [**binaire** *int into [0, 1]*]
> [**ecrire_frontiere** *int into [0, 1]*]

}
where

- **format** *str into ['lml', 'lata', 'lata_v1', 'lata_v2', 'med']*: File format.
- **file|fichier** *str*: The file name can be changed with the fichier option.
- **domaine** *str*: Name of domain
- **domaines** *bloc_lecture* (3.17): Names of domains : { name1 name2 }
- **joints_non_postraites** *int into [0, 1]*: The joints_non_postraites (1 by default) will not write the boundaries between the partitioned mesh.
- **binaire** *int into [0, 1]*: Binary (binaire 1) or ASCII (binaire 0) may be used. By default, it is 0 for LATA and only ASCII is available for LML and only binary is available for MED.
- **ecrire_frontiere** *int into [0, 1]*: This option will write (if set to 1, the default) or not (if set to 0) the boundaries as fields into the file (it is useful to not add the boundaries when writing a domain extracted from another domain)

3.78 Precisiongeom

Description: Class to change the way floating-point number comparison is done. By default, two numbers are equal if their absolute difference is smaller than 1e-10. The keyword is useful to modify this value. Moreover, nodes coordinates will be written in .geom files with this same precision.

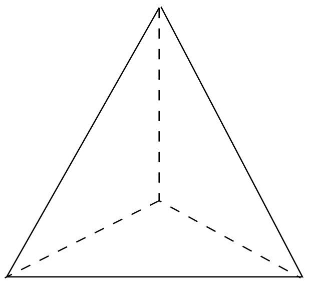
See also: interprete (3)

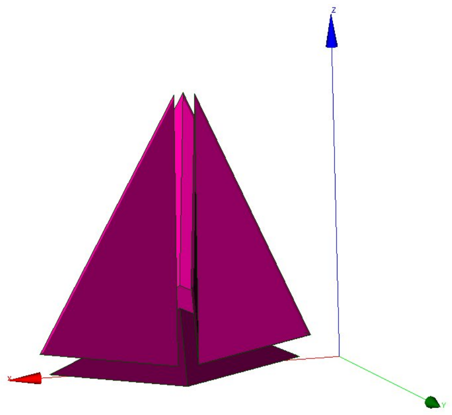
Usage:
precisiongeom precision
where

- **precision** *float*: New value of precision.

3.79 Raffiner_anisotrope

Description: Only for VEF discretizations, allows to cut triangle elements in 3, or tetrahedra in 4 parts, by defining a new summit located at the center of the element:





Note that such a cut creates flat elements (anisotropic).

See also: interprete (3)

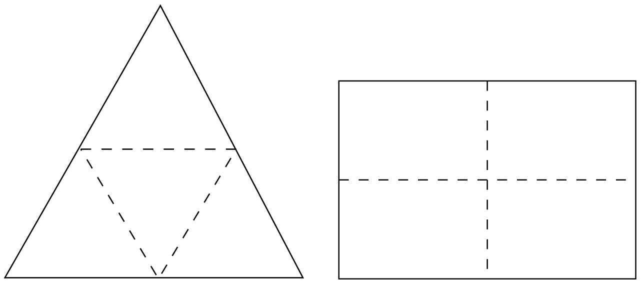
Usage:
raffiner_anisotrope domain_name
where

- **domain_name** *str*: Name of domain.

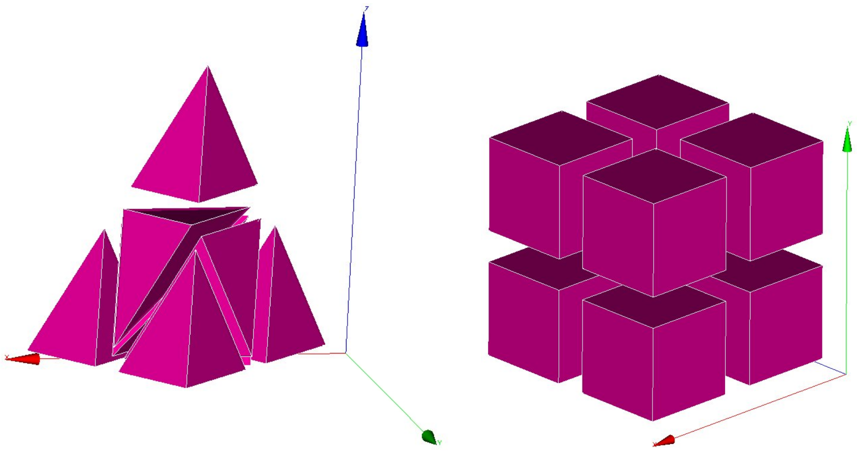
3.80 Raffiner_isotrope

Synonymous: **raffiner_simplexes**

Description: For VDF and VEF discretizations, allows to cut triangles/quadrangles or tetrahedral/hexaedras elements respectively in 4 or 8 new ones by defining new summits located at the middle of edges (and center of faces and elements for quadrangles and hexaedra). Such a cut preserves the shape of original elements (isotropic). For 2D elements:



For 3D elements:



.

See also: interprete (3)

Usage:
raffiner_isotrope domain_name
where

- **domain_name** *str*: Name of domain.

3.81 Read

Synonymous: **lire**

Description: Interpretor to read the a_object objet defined between the braces.

See also: interprete (3)

Usage:
read a_object bloc
where

- **a_object** *str*: Object to be read.
- **bloc** *str*: Definition of the object.

3.82 Read_file

Synonymous: **lire_fichier**

Description: Keyword to read the object name_obj contained in the file filename.
This is notably used when the calculation domain has already been meshed and the mesh contains the file filename, simply write read_file dom filename (where dom is the name of the meshed domain).
If the filename is ;, is to execute a data set given in the file of name name_obj (a space must be entered between the semi-colon and the file name).

See also: interprete (3) read_unsupported_ascii_file_from_icem (3.85) read_file_binary (3.83)

Usage:
read_file name_obj filename
where

- **name_obj** *str*: Name of the object to be read.
- **filename** *str*: Name of the file.

3.83 Read_file_binary

Synonymous: **lire_fichier_bin**

Description: Keyword to read an object name_obj in the unformatted type file filename.

See also: read_file (3.82)

Usage:
read_file_binary name_obj filename
where

- **name_obj** *str*: Name of the object to be read.
- **filename** *str*: Name of the file.

3.84 Lire_tgrid

Description: Keyword to reaf Tgrid/Gambit mesh files. 2D (triangles or quadrangles) and 3D (tetra or hexa elements) meshes, may be read by TRUST.

See also: interprete (3)

Usage:
lire_tgrid dom filename
where

- **dom** *str*: Name of domaine.
- **filename** *str*: Name of file containing the mesh.

3.85 Read_unsupported_ascii_file_from_icem

Description: not_set

See also: read_file (3.82)

Usage:
read_unsupported_ascii_file_from_icem name_obj filename
where

- **name_obj** *str*: Name of the object to be read.
- **filename** *str*: Name of the file.

3.86 Orienter_simplexes

Synonymous: **rectify_mesh**

Description: Keyword to raffine a mesh

See also: interprete (3)

Usage:
orienter_simplexes domain_name
where

- **domain_name** *str*: Name of domain.

3.87 Redresser_hexaedres_vdf

Description: Keyword to convert a domain (named domain_name) with quadrilaterals/VEF hexaedras which looks like rectangles/VDF hexaedras into a domain with real rectangles/VDF hexaedras.

See also: interprete (3)

Usage:
redresser_hexaedres_vdf domain_name
where

- **domain_name** *str*: Name of domain to resequence.

3.88 Refine_mesh

Description: not_set

See also: interprete (3)

Usage:
refine_mesh domaine
where

- **domaine** *str*

3.89 Regroupebord

Description: Keyword to build one boundary new_bord with several boundaries of the domain named domaine.

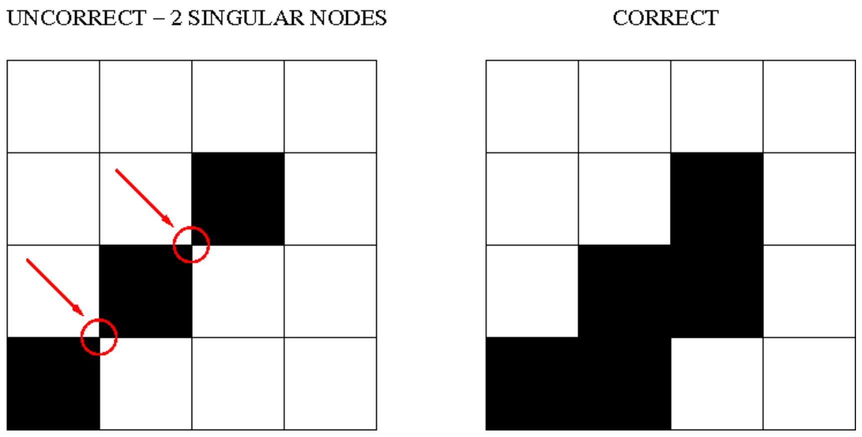
See also: interprete (3)

Usage:
regroupebord domaine new_bord bords
where

- **domaine** *str*: Name of domain
- **new_bord** *str*: Name of the new boundary
- **bords** *bloc_lecture* (3.17): { Bound1 Bound2 }

3.90 Remove_elem

Description: Keyword to remove element from a VDF mesh (named domaine_name), either from an explicit list of elements or from a geometric condition defined by a condition f(x,y)>0 in 2D and f(x,y,z)>0 in 3D. All the new borders generated are gathered in one boundary called : newBord (to rename it, use RegroupeBord keyword. To split it to different boundaries, use DecoupeBord_Pour_Rayonnement keyword). Example of a removed zone of radius 0.2 centered at (x,y)=(0.5,0.5):
Remove_elem dom { fonction $0.2 * 0.2 - (x - 0.5)^2 - (y - 0.5)^2 > 0$ }
Warning : the thickness of removed zone has to be large enough to avoid singular nodes as decribed below :



See also: interprete (3)

Usage:
remove_elem domaine bloc
where

- **domaine** *str*: Name of domain
- **bloc** *remove_elem_bloc* (3.91)

3.91 Remove_elem_bloc

Description: not_set

See also: objet_lecture (33)

Usage:
{

[**liste** *n n1 n2 ... nn*]
[**fonction** *str*]

}
where

- **liste** *n n1 n2 ... nn*
- **fonction** *str*

3.92 Remove_invalid_internal_boundaries

Description: Keyword to suppress an internal boundary of the domain_name domain. Indeed, some mesh tools may define internal boundaries (eg: for post processing task after the calculation) but TRUST does not support it yet.

See also: interprete (3)

Usage:
remove_invalid_internal_boundaries domain_name
where

- **domain_name** *str*: Name of domain.

3.93 Reorienter_tetraedres

Description: This keyword is mandatory for front-tracking computations with the VEF discretization. For each tetrahedral element of the domain, it checks if it has a positive volume. If the volume (determinant of the three vectors) is negative, it swaps two nodes to reverse the orientation of this tetrahedron.

See also: interprete (3)

Usage:
reorienter_tetraedres domain_name
where

- **domain_name** *str*: Name of domain.

3.94 Reorienter_triangles

Description: not_set

See also: interprete (3)

Usage:
reorienter_triangles domain_name
where

- **domain_name** *str*: Name of domain.

3.95 Reordonner

Description: The Reordonner interpretor is required sometimes for a VDF mesh which is not produced by the internal mesher. Example where this is used:
Read_file dom fichier.geom
Reordonner dom
Observations: This keyword is redundant when the mesh that is read is correctly sequenced in the TRUST sense. This significant mesh operation may take some time... The message returned by TRUST is not explicit when the Reordonner (Resequencing) keyword is required but not included in the data set...

See also: interprete (3)

Usage:
reordonner domain_name
where

- **domain_name** *str*: Name of domain to resequence.

3.96 Rotation

Description: Keyword to rotate the geometry of an arbitrary angle around an axis aligned with Ox, Oy or Oz axis.

See also: interprete (3)

Usage:
rotation domain_name dir coord1 coord2 angle
where

- **domain_name** *str*: Name of domain to wich the transformation is applied.
- **dir** *str into ['X', 'Y', 'Z']*: X, Y or Z to indicate the direction of the rotation axis
- **coord1** *float*: coordinates of the center of rotation in the plane orthogonal to the rotation axis. These coordinates must be specified in the direct triad sense.
- **coord2** *float*
- **angle** *float*: angle of rotation (in degrees)

3.97 Scatter

Description: Class to read a partionned mesh in the files during a parallel calculation. The files are in binary format.

See also: interprete (3) scattermed (3.98)

Usage:
scatter file domaine
where

- **file** *str*: Name of file.
- **domaine** *str*: Name of domain.

3.98 Scattermed

Description: This keyword will read the partition of the domain_name domain into a the MED format files file.med created by Medsplitter.

See also: scatter (3.97)

Usage:
scattermed file domaine
where

- **file** *str*: Name of file.
- **domaine** *str*: Name of domain.

3.99 Solve

Synonymous: **resoudre**

Description: Interpretor to start calculation with TRUST.

Keyword Discretize should have already been used to read the object.
See also: interprete (3)

Usage:
solve pb
where

- **pb** *str*: Name of problem to be solved.

3.100 Supprime_bord

Description: Keyword to remove boundaries (named Boundary_name1 Boundary_name2) of the domain named domain_name.

See also: interprete (3)

Usage:
supprime_bord domaine bords
where

- **domaine** *str*: Name of domain
- **bords** *list_nom* (3.101): { Boundary_name1 Boundaray_name2 }

3.101 List_nom

Description: List of name.

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *nom_anonyme* (22.1)

3.102 System

Description: To run Unix commands from the data file. Example: System 'echo The End | mail trust@cea.fr'

See also: interprete (3)

Usage:
system cmd
where

- **cmd** *str*: command to execute.

3.103 Test_solveur

Description: To test several solvers

See also: interprete (3)

Usage:
test_solveur {

> [**fichier_secmem** *str*]
> [**fichier_matrice** *str*]
> [**fichier_solution** *str*]
> [**nb_test** *int*]
> [**impr**]
> [**solveur** *solveur_sys_base*]
> [**fichier_solveur** *str*]
> [**genere_fichier_solveur** *float*]
> [**seuil_verification** *float*]
> [**pas_de_solution_initiale**]
> [**ascii**]

}
where

- **fichier_secmem** *str*: Filename containing the second member B
- **fichier_matrice** *str*: Filename containing the matrix A
- **fichier_solution** *str*: Filename containing the solution x
- **nb_test** *int*: Number of tests to measure the time resolution (one preconditionnement)
- **impr** : To print the convergence solver
- **solveur** *solveur_sys_base* (9.13): To specify a solver
- **fichier_solveur** *str*: To specify a file containing a list of solvers
- **genere_fichier_solveur** *float*: To create a file of the solver with a threshold convergence
- **seuil_verification** *float*: Check if the solution satisfy ||Ax-B||<precision
- **pas_de_solution_initiale** : Resolution isn't initialized with the solution x
- **ascii** : Ascii files

3.104 Testeur

Description: not_set

See also: interprete (3)

Usage:
testeur data
where

- **data** *bloc_lecture* (3.17)

3.105 Testeur_medcoupling

Description: not_set

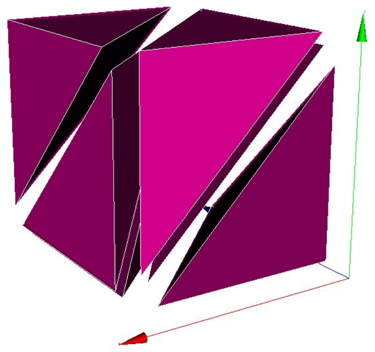
See also: interprete (3)

Usage:
testeur_medcoupling pb_name field_name
where

- **pb_name** *str*: Name of domain.
- **field_name** *str*: Name of domain.

3.106 Tetraedriser

Description: To achieve a tetrahedral mesh based on a mesh comprising blocks, the Tetraedriser (Tetrahedralise) interpretor is used in VEF discretization. Initial block is divided in 6 tetrahedra:



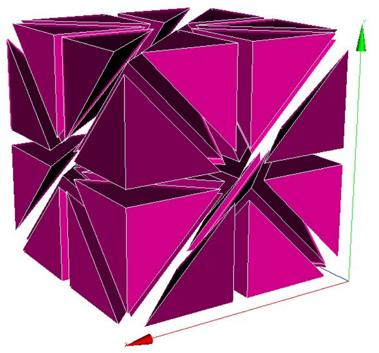
See also: interprete (3) tetraedriser_homogene (3.107) tetraedriser_homogene_fin (3.109) tetraedriser_homogene_compact (3.108) tetraedriser_par_prisme (3.110)

Usage:
tetraedriser domain_name
where

- **domain_name** *str*: Name of domain.

3.107 Tetraedriser_homogene

Description: Use the Tetraedriser_homogene (Homogeneous_Tetrahedralisation) interpretor in VEF discretization to mesh a block in tetrahedrals. Each block hexahedral is no longer divided into 6 tetrahedrals (keyword Tetraedriser (Tetrahedralise)), it is now broken down into 40 tetrahedrals. Thus a block defined with 11 nodes in each X, Y, Z direction will contain 10*10*10*40=40,000 tetrahedrals. This also allows problems in the mesh corners with the P1NC/P1iso/P1bulle or P1/P1 discretization items to be avoided. Initial block is divided in 40 tetrahedra:



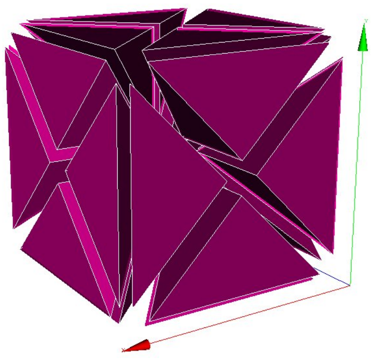
See also: tetraedriser (3.106)

Usage:
tetraedriser_homogene domain_name
where

- **domain_name** *str*: Name of domain.

3.108 Tetraedriser_homogene_compact

Description: This new discretization generates tetrahedral elements from cartesian or non-cartesian hexahedral elements. The process cut each hexahedral in 6 pyramids, each of them being cut then in 4 tetrahedral. So, in comparison with tetra_homogene, less elements (*24 instead of*40) with more homogeneous volumes are generated. Moreover, this process is done in a faster way. Initial block is divided in 24 tetrahedra:



See also: tetraedriser (3.106)

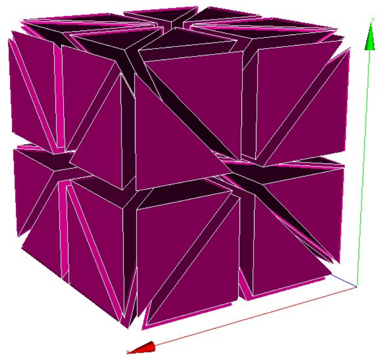
Usage:

tetraedriser_homogene_compact **domain_name**
where

- **domain_name** *str*: Name of domain.

3.109 Tetraedriser_homogene_fin

Description: Tetraedriser_homogene_fin is the recommended option to tetrahedralise blocks. As an extension (subdivision) of Tetraedriser_homogene_compact, this last one cut each initial block in 48 tetrahedra (against 24, previously). This cutting ensures :
- a correct cutting in the corners (in respect to pressure discretization PreP1B),
- a better isotropy of elements than with Tetraedriser_homogene_compact,
- a better alignment of summits (this could have a benefit effect on calculation near walls since first elements in contact with it are all contained in the same constant thickness and ii/ by the way, a 3D cartesian grid based on summits can be engendered and used to realise spectral analysis in HIT for instance). Initial block is divided in 48 tetrahedra:



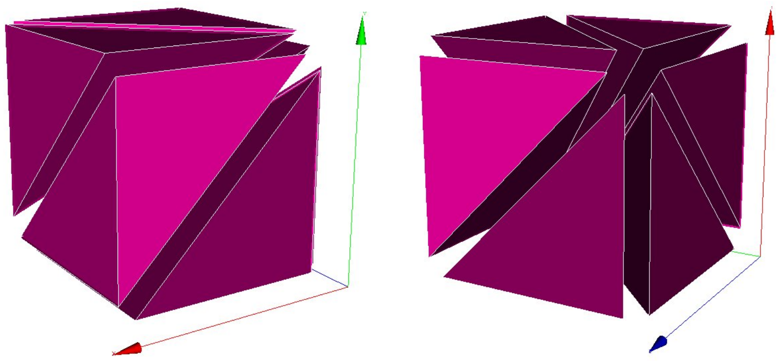
See also: tetraedriser (3.106)

Usage:
tetraedriser_homogene_fin **domain_name**
where

- **domain_name** *str*: Name of domain.

3.110 Tetraedriser_par_prisme

Description: Tetraedriser_par_prisme generates 6 iso-volume tetrahedral element from primary hexahedral one (contrarily to the 5 elements ordinarily generated by tetraedriser). This element is suitable for calculation of gradients at the summit (coincident with the gravity centre of the jointed elements related with) and spectra (due to a better alignment of the points).



Initial block is divided in 6 prismes.

See also: tetraedriser (3.106)

Usage:
tetraedriser_par_prisme domain_name
where

- **domain_name** *str*: Name of domain.

3.111 Transformer

Description: Keyword to transform the coordinates of the geometry.
Exemple to rotate your mesh by a 90o rotation and to scale the z coordinates by a factor 2: Transformer domain_name -y -x 2*z

See also: interprete (3)

Usage:
transformer domain_name formule
where

- **domain_name** *str*: Name of domain.
- **formule** *word1 word2 (word3)*: Function_for_x Function_for_y

$$Function\_forz$$

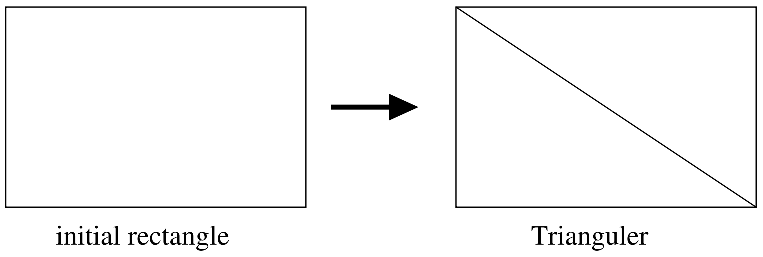
3.112 Trianguler

Description: To achieve a triangular mesh from a mesh comprising rectangles (2 triangles per rectangle). Should be used in VEF discretization. Principle:

See also: interprete (3) trianguler_h (3.114) trianguler_fin (3.113)

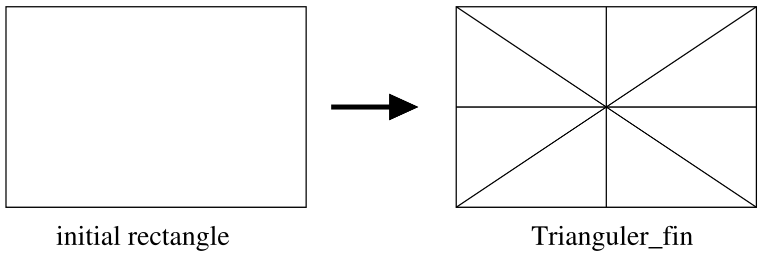
Usage:
trianguler domain_name
where

- **domain_name** *str*: Name of domain.



3.113 Trianguler_fin

Description: Trianguler_fin is the recommended option to triangulate rectangles.
As an extension (subdivision) of Triangulate_h option, this one cut each initial rectangle in 8 triangles (against 4, previously). This cutting ensures :
- a correct cutting in the corners (in respect to pressure discretization PreP1B).
- a better isotropy of elements than with Trianguler_h option.
- a better alignment of summits (this could have a benefit effect on calculation near walls since first elements in contact with it are all contained in the same constant thickness, and, by this way, a 2D cartesian grid based on summits can be engendered and used to realize statistical analysis in plane channel configuration for instance). Principle:



See also: trianguler (3.112)

Usage:
trianguler_fin domain_name
where

- **domain_name** *str*: Name of domain.

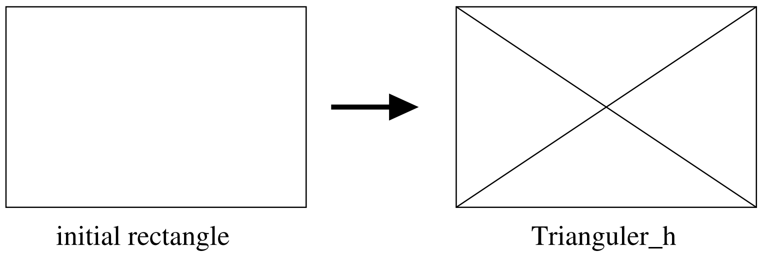
3.114 Trianguler_h

Description: To achieve a triangular mesh from a mesh comprising rectangles (4 triangles per rectangle). Should be used in VEF discretization. Principle:

See also: trianguler (3.112)

Usage:
trianguler_h domain_name
where

- **domain_name** *str*: Name of domain.



3.115 Verifier_qualite_raffinements

Description: not_set

See also: interprete (3)

Usage:
verifier_qualite_raffinements domain_names
where

- **domain_names** *vect_nom* (3.116)

3.116 Vect_nom

Description: Vect of name.

See also: listobj (32.5)

Usage:
n object1 object2 ....
list of *nom_anonyme* (22.1)

3.117 Verifier_simplexes

Description: Keyword to raffine a simplexes

See also: interprete (3)

Usage:
verifier_simplexes domain_name
where

- **domain_name** *str*: Name of domain.

3.118 Verifiercoin

Description: This keyword subdivides inconsistent 2D/3D cells used with VEFPreP1B discretization. Must be used before the mesh is discretized. The Read_file option can be used only if the file.decoupage_som was previously created by TRUST. This option, only in 2D, reverses the common face at two cells (at least one is inconsistent), through the nodes opposed. In 3D, the option has no effect.
The expert_only option deactivates, into the VEFPreP1B divergence operator, the test of inconsistent cells.

See also: interprete (3)

Usage:
verifiercoin domain_name bloc
where

- **domain_name** *str*: Name of the domaine
- **bloc** *verifiercoin_bloc* (3.119)

3.119 Verifiercoin_bloc

Description: not_set

See also: objet_lecture (33)

Usage:
{

[**Lire_fichier|Read_file** *str*]
[**expert_only**]

}
where

- **Lire_fichier|Read_file** *str*: name of the *.decoupage_som file
- **expert_only** : to not check the mesh

3.120 Ecrire

Description: Keyword to write the object of name name_obj to a standard outlet.

See also: interprete (3)

Usage:
ecrire name_obj
where

- **name_obj** *str*: Name of the object to be written.

3.121 Ecrire_fichier_bin

Synonymous: **ecrire_fichier**

Description: Keyword to write the object of name name_obj to a file filename. Since the v1.6.3, the default format is now binary format file.

See also: interprete (3) ecrire_fichier_formatte (3.34)

Usage:
ecrire_fichier_bin name_obj filename
where

- **name_obj** *str*: Name of the object to be written.
- **filename** *str*: Name of the file.

3.122 Ecrire_med

Description: Write a domain to MED format into a file.

See also: interprete (3) ecrire_medfile (3.123)

Usage:
ecrire_med nom_dom file
where

- **nom_dom** *str*: Name of domain.
- **file** *str*: Name of file.

3.123 Ecrire_medfile

Description: Obsolete keyword to write a mesh with MED file API

See also: ecrire_med (3.122)

Usage:
ecrire_medfile nom_dom file
where

- **nom_dom** *str*: Name of domain.
- **file** *str*: Name of file.

4 pb_gen_base

Description: Basic class for problems.

See also: objet_u (34) Pb_base (4.8) probleme_couple (4.9) pbc_med (4.27)

Usage:

4.1 Pb_conduction

Description: Resolution of the heat equation.

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8)

Usage:
Pb_Conduction *str*
Read *str* {

[**Conduction** *conduction*]
[**Post_processing|postraitement** *corps_postraitement*]
[**Post_processings|postraitements** *post_processings*]
[**liste_de_postraitements** *liste_post_ok*]
[**liste_postraitements** *liste_post*]
[**sauvegarde** *format_file*]
[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **Conduction** *conduction* (5.1): Heat equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.2 Corps_postraitement

Description: not_set

See also: post_processing (4.4.3)

Usage:
{

[**definition_champs** *definition_champs*]
[**Probes|sondes** *sondes*]
[**domaine** *str*]
[**format** *str into ['lml', 'lata', 'lata_v1', 'lata_v2', 'med', 'med_major']*]
[**parallele** *str into ['simple', 'multiple', 'mpi-io']*]
[**fields|champs** *champs_posts*]
[**statistiques** *stats_posts*]
[**fichier** *str*]
[**statistiques_en_serie** *stats_serie_posts*]
[**interfaces** *champs_posts*]

}
where

- **definition_champs** *definition_champs* (4.2.1) for inheritance: Keyword to create new or more complex field for advanced postprocessing.

- **Probes|sondes** *sondes* (4.2.3) for inheritance: Probe.
- **domaine** *str* for inheritance: This optional parameter specifies the domain on which the data should be interpolated before it is written in the output file. The default is to write the data on the domain of the current problem (no interpolation).
- **format** *str into ['lml', 'lata', 'lata_v1', 'lata_v2', 'med', 'med_major']* for inheritance: This optional parameter specifies the format of the output file. The basename used for the output file is the basename of the data file. For the fmt parameter, choices are lml or lata. A short description of each format can be found below. The default value is lml.
- **parallele** *str into ['simple', 'multiple', 'mpi-io']* for inheritance: Select simple (single file, sequential write), multiple (several files, parallel write), or mpi-io (single file, parallel write) for LATA format
- **fields|champs** *champs_posts* (4.2.20) for inheritance: Field's write mode.
- **statistiques** *stats_posts* (4.2.23) for inheritance: Statistics between two points fixed : start of integration time and end of integration time.
- **fichier** *str* for inheritance: Name of file.
- **statistiques_en_serie** *stats_serie_posts* (4.2.31) for inheritance: Statistics between two points not fixed : on period of integration.
- **interfaces** *champs_posts* (4.2.20) for inheritance: Keyword to read all the caracteristics of the interfaces. Different kind of interfaces exist as well as different interface intitialisations.

4.2.1 Definition_champs

Description: List of definition champ

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *definition_champ* (4.2.2)

4.2.2 Definition_champ

Description: Keyword to create new complex field for advanced postprocessing.

See also: objet_lecture (33)

Usage:
name **champ_generique**
where

- **name** *str*: The name of the new created field.
- **champ_generique** *champ_generique_base* (7)

4.2.3 Sondes

Description: List of probes.

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *sonde* (4.2.4)

4.2.4 Sonde

Description: Keyword is used to define the probes. Observations: the probe coordinates should be given in Cartesian coordinates (X, Y, Z), including axisymmetric.

See also: objet_lecture (33)

Usage:
nom_sonde [**special**] **nom_inco** **mperiode** **prd** **type**
where

- **nom_sonde** *str*: Name of the file in which the values taken over time will be saved. The complete file name is nom_sonde.son.
- **special** *str into ['grav', 'som', 'nodes', 'chsom', 'gravcl']*: Option to change the positions of the probes. Several options are available:
  grav : each probe is moved to the nearest cell center of the mesh;
  som : each probe is moved to the nearest vertex of the mesh
  nodes : each probe is moved to the nearest face center of the mesh;
  chsom : only available for P1NC sampled field. The values of the probes are calculated according to P1-Conform corresponding field.
  gravcl : Extend to the domain face boundary a cell-located segment probe in order to have the boundary condition for the field. For this type the extreme probe point has to be on the face center of gravity.
- **nom_inco** *str*: Name of the sampled field.
- **mperiode** *str into ['periode']*: Keyword to set the sampled field measurement frequency.
- **prd** *float*: Period value. Every prd seconds, the field value calculated at the previous time step is written to the nom_sonde.son file.
- **type** *sonde_base* (4.2.5): Type of probe.

4.2.5 Sonde_base

Description: Basic probe. Probes refer to sensors that allow a value or several points of the domain to be monitored over time. The probes may be a set of points defined one by one (keyword Points) or a set of points evenly distributed over a straight segment (keyword Segment) or arranged according to a layout (keyword Plan) or according to a parallelepiped (keyword Volume). The fields allow all the values of a physical value on the domain to be known at several moments in time.

See also: objet_lecture (33) points (4.2.6) numero_elem_sur_maitre (4.2.10) position_like (4.2.11) segment (4.2.12) plan (4.2.13) volume (4.2.14) circle (4.2.15) circle_3 (4.2.16) segmentfacesx (4.2.17) segmentfacesy (4.2.18) segmentfacesz (4.2.19)

Usage:
sonde_base

4.2.6 Points

Description: Keyword to define the number of probe points. The file is arranged in columns.

See also: sonde_base (4.2.5) point (4.2.8) segmentpoints (4.2.9)

Usage:
points **points**
where

- **points** *listpoints* (4.2.7): Probe points.

4.2.7 Listpoints

Description: Points.

See also: listobj (32.5)

Usage:
n object1 object2 ....
list of *un_point* (3.19.3)

4.2.8 Point

Description: Point as class-daughter of Points.

See also: points (4.2.6)

Usage:
point points
where

- **points** *listpoints* (4.2.7): Probe points.

4.2.9 Segmentpoints

Description: This keyword is used to define a probe segment from specifics points. The nom_champ field is sampled at ns specifics points.

See also: points (4.2.6)

Usage:
segmentpoints points
where

- **points** *listpoints* (4.2.7): Probe points.

4.2.10 Numero_elem_sur_maitre

Description: Keyword to define a probe at the special element. Useful for min/max sonde.

See also: sonde_base (4.2.5)

Usage:
numero_elem_sur_maitre numero
where

- **numero** *int*: element number

4.2.11 Position_like

Description: Keyword to define a probe at the same position of another probe named autre_sonde.

See also: sonde_base (4.2.5)

Usage:

position_like autre_sonde
where

- **autre_sonde** *str*: Name of the other probe.

4.2.12 Segment

Description: Keyword to define the number of probe segment points. The file is arranged in columns.

See also: sonde_base (4.2.5)

Usage:
segment nbr point_deb point_fin
where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* (3.19.3): First outer probe segment point.
- **point_fin** *un_point* (3.19.3): Second outer probe segment point.

4.2.13 Plan

Description: Keyword to set the number of probe layout points. The file format is type .lml

See also: sonde_base (4.2.5)

Usage:
plan nbr nbr2 point_deb point_fin point_fin_2
where

- **nbr** *int*: Number of probes in the first direction.
- **nbr2** *int*: Number of probes in the second direction.
- **point_deb** *un_point* (3.19.3): First point defining the angle. This angle should be positive.
- **point_fin** *un_point* (3.19.3): Second point defining the angle. This angle should be positive.
- **point_fin_2** *un_point* (3.19.3): Third point defining the angle. This angle should be positive.

4.2.14 Volume

Description: Keyword to define the probe volume in a parallelepiped passing through 4 points and the number of probes in each direction.

See also: sonde_base (4.2.5)

Usage:
volume nbr nbr2 nbr3 point_deb point_fin point_fin_2 point_fin_3
where

- **nbr** *int*: Number of probes in the first direction.
- **nbr2** *int*: Number of probes in the second direction.
- **nbr3** *int*: Number of probes in the third direction.
- **point_deb** *un_point* (3.19.3): Point of origin.
- **point_fin** *un_point* (3.19.3): Point defining the first direction (from point of origin).
- **point_fin_2** *un_point* (3.19.3): Point defining the second direction (from point of origin).
- **point_fin_3** *un_point* (3.19.3): Point defining the third direction (from point of origin).

4.2.15 Circle

Description: Keyword to define several probes located on a circle.

See also: sonde_base (4.2.5)

Usage:
circle nbr point_deb [**direction**] **radius theta1 theta2**
where

- **nbr** *int*: Number of probes between teta1 and teta2 (angles given in degrees).
- **point_deb** *un_point* (3.19.3): Center of the circle.
- **direction** *int into [0, 1, 2]*: Axis normal to the circle plane (0:x axis, 1:y axis, 2:z axis).
- **radius** *float*: Radius of the circle.
- **theta1** *float*: First angle.
- **theta2** *float*: Second angle.

4.2.16 Circle_3

Description: Keyword to define several probes located on a circle (in 3-D space).

See also: sonde_base (4.2.5)

Usage:
circle_3 nbr point_deb direction radius theta1 theta2
where

- **nbr** *int*: Number of probes between teta1 and teta2 (angles given in degrees).
- **point_deb** *un_point* (3.19.3): Center of the circle.
- **direction** *int into [0, 1, 2]*: Axis normal to the circle plane (0:x axis, 1:y axis, 2:z axis).
- **radius** *float*: Radius of the circle.
- **theta1** *float*: First angle.
- **theta2** *float*: Second angle.

4.2.17 Segmentfacesx

Description: Segment probe where points are moved to the nearest x faces

See also: sonde_base (4.2.5)

Usage:
segmentfacesx nbr point_deb point_fin
where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* (3.19.3): First outer probe segment point.
- **point_fin** *un_point* (3.19.3): Second outer probe segment point.

4.2.18 Segmentfacesy

Description: Segment probe where points are moved to the nearest y faces

See also: sonde_base (4.2.5)

Usage:
segmentfacesy nbr point_deb point_fin
where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* (3.19.3): First outer probe segment point.
- **point_fin** *un_point* (3.19.3): Second outer probe segment point.

4.2.19 Segmentfacesz

Description: Segment probe where points are moved to the nearest z faces

See also: sonde_base (4.2.5)

Usage:
segmentfacesz nbr point_deb point_fin
where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* (3.19.3): First outer probe segment point.
- **point_fin** *un_point* (3.19.3): Second outer probe segment point.

4.2.20 Champs_posts

Description: Field's write mode.

See also: objet_lecture (33)

Usage:
[**format**] **mot period fields|champs**
where

- **format** *str into ['binaire', 'formatte']*: Type of file.
- **mot** *str into ['dt_post', 'nb_pas_dt_post']*: Keyword to set the kind of the field's write frequency. Either a time period or a time step period.
- **period** *str*: Value of the period which can be like (2.*t).
- **fields|champs** *champs_a_post* (4.2.21): Post-processed fields.

4.2.21 Champs_a_post

Description: Fields to be post-processed.

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *champ_a_post* (4.2.22)

4.2.22 Champ_a_post

Description: Field to be post-processed.

See also: objet_lecture (33)

Usage:
champ [**localisation**]
where

- **champ** *str*: Name of the post-processed field.
- **localisation** *str into ['elem', 'som', 'faces']*: Localisation of post-processed field values: The two available values are elem, som, or faces (LATA format only) used respectively to select field values at mesh centres (CHAMPMAILLE type field in the lml file) or at mesh nodes (CHAMPPOINT type field in the lml file). If no selection is made, localisation is set to som by default.

4.2.23 Stats_posts

Description: Field's write mode.
Dt_post: This keyword is used to set the calculated statistics write period.
dts: frequency value.
t_deb value: Start of integration time
t_fin value: End of integration time

stat: Set to **Moyenne (average)** to calculate the average of the field *nom_champ* (field name) over time or **Ecart_type (std_deviation)** to calculate the standard deviation (statistic rms) of the field *nom_champ (field_name)* or **Correlation** to calculate the correlation between the two fields *nom_champ* and *second_nom_champ.*

nom_champ: name of the field on which statistical analysis will be performed. Possible keywords are **Vitesse (velocity)**, **Pression (pressure)**, **Temperature**, **Concentration,...**

localisation: localisation of post-processed field values (**elem** or **som**).

Example:
Statistiques Dt_post dtst {
 t_deb 0.1 **t_fin** 0.12
Moyenne Pression
Ecart_type Pression
Correlation Vitesse Vitesse }
It will write every **dt_post** the mean, standard deviation and correlation value:

t<=t_{deb} :
average: $\overline{P(t)} = 0$
std_deviation: $<P(t)$>=0
correlation: $<U(t).V(t)$>=0

$t > t_{\text{deb}}$:
average: $\overline{P(t)} = \frac{1}{t-t_{\text{deb}}} \int_{t_{\text{deb}}}^{t} P(t)\text{dt}$
std_deviation: $<P(t)$>=$\sqrt{\frac{1}{t-t_{\text{deb}}} \int_{t_{\text{deb}}}^{t} \left[P(t) - \overline{P(t)}\right]^2 \text{dt}}$
correlation: $<U(t).V(t)$>=$\frac{1}{t-t_{\text{deb}}} \int_{t_{\text{deb}}}^{t} \left[U(t) - \overline{U(t)}\right] . \left[V(t) - \overline{V(t)}\right] \text{dt}$

See also: objet_lecture (33)

Usage:

mot period fields|champs
where

- **mot** *str into ['dt_post', 'nb_pas_dt_post']*: Keyword to set the kind of the field's write frequency. Either a time period or a time step period.
- **period** *str*: Value of the period which can be like (2.*t).
- **fields|champs** *list_stat_post* (4.2.24): Post-processed fields.

4.2.24 List_stat_post

Description: Post-processing for statistics

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *stat_post_deriv* (4.2.25)

4.2.25 Stat_post_deriv

Description: not_set

See also: objet_lecture (33) t_deb (4.2.26) t_fin (4.2.27) moyenne (4.2.28) ecart_type (4.2.29) correlation (4.2.30)

Usage:
stat_post_deriv

4.2.26 T_deb

Description: not_set

See also: stat_post_deriv (4.2.25)

Usage:
t_deb val
where

- **val** *float*

4.2.27 T_fin

Description: not_set

See also: stat_post_deriv (4.2.25)

Usage:
t_fin val
where

- **val** *float*

4.2.28 Moyenne

Synonymous: **champ_post_statistiques_moyenne**

Description: not_set

See also: stat_post_deriv (4.2.25)

Usage:
moyenne **field** [**localisation**]
where

- **field** *str*
- **localisation** *str into ['elem', 'som', 'faces']*: Localisation of post-processed field value

4.2.29 Ecart_type

Synonymous: **champ_post_statistiques_ecart_type**

Description: not_set

See also: stat_post_deriv (4.2.25)

Usage:
ecart_type **field** [**localisation**]
where

- **field** *str*
- **localisation** *str into ['elem', 'som', 'faces']*: Localisation of post-processed field value

4.2.30 Correlation

Synonymous: **champ_post_statistiques_correlation**

Description: not_set

See also: stat_post_deriv (4.2.25)

Usage:
correlation **first_field** **second_field** [**localisation**]
where

- **first_field** *str*
- **second_field** *str*
- **localisation** *str into ['elem', 'som', 'faces']*: Localisation of post-processed field value

4.2.31 Stats_serie_posts

Description: Post-processing for statistics.
Statistiques_en_serie: This keyword is used to set the statistics. Average on **dt_integr** time interval is post-processed every **dt_integr** seconds

dt_integr value : Period of integration and write period.

stat: Set to **Moyenne (average)** to calculate the average of the field *nom_champ* (field name) over time or **Ecart_type (std_deviation)** to calculate the standard deviation (statistic rms) of the field *nom_champ (field_name)*.

nom_champ: name of the field on which statistical analysis will be performed. Possible keywords are **Vitesse (velocity)**, **Pression (pressure)**, **Temperature**, **Concentration,...**

localisation: localisation of post-processed field values (**elem** or **som**).

Example:

Statistiques_en_serie Dt_integr dtst {
Moyenne Pression
}
Will calculate and write every dtst seconds the mean value:

$$(n+1)\text{dt\_integr} > t > n * \text{dt\_integr}, \overline{P(t)} = \frac{1}{t - n * \text{dt\_integr}} \int_{t_n * \text{dt\_integr}}^{t} P(t)\text{dt}$$

See also: objet_lecture (33)

Usage:
mot dt_integr stat
where

- **mot** *str into ['dt_integr']*: Keyword is used to set the statistics period of integration and write period.
- **dt_integr** *float*: Average on dt_integr time interval is post-processed every dt_integr seconds.
- **stat** *list_stat_post* (4.2.24)

4.3 Post_processings

Synonymous: **postraitements**

Description: Keyword to use several results files. List of objects of post-processing (with name).

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *un_postraitement* (4.3.1)

4.3.1 Un_postraitement

Description: An object of post-processing (with name).

See also: objet_lecture (33)

Usage:
nom post
where

- **nom** *str*: Name of the post-processing.
- **post** *corps_postraitement* (4.2): Definition of the post-processing.

4.4 Liste_post_ok

Description: Keyword to use several results files. List of objects of post-processing (with name)

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *nom_postraitement* (4.4.1)

4.4.1 Nom_postraitement

Description:

See also: objet_lecture (33)

Usage:
nom post
where

- **nom** *str*: Name of the post-processing.
- **post** *postraitement_base* (4.4.2): the post

4.4.2 Postraitement_base

Description: not_set

See also: objet_lecture (33) post_processing (4.4.3)

Usage:

4.4.3 Post_processing

Synonymous: **postraitement**

Description: An object of post-processing (without name).

See also: postraitement_base (4.4.2) corps_postraitement (4.2)

Usage:
post_processing {

[**definition_champs** *definition_champs*]
[**Probes|sondes** *sondes*]
[**domaine** *str*]
[**format** *str into ['lml', 'lata', 'lata_v1', 'lata_v2', 'med', 'med_major']*]
[**parallele** *str into ['simple', 'multiple', 'mpi-io']*]
[**fields|champs** *champs_posts*]
[**statistiques** *stats_posts*]
[**fichier** *str*]
[**statistiques_en_serie** *stats_serie_posts*]
[**interfaces** *champs_posts*]

}
where

- **definition_champs** *definition_champs* (4.2.1): Keyword to create new or more complex field for advanced postprocessing.
- **Probes|sondes** *sondes* (4.2.3): Probe.
- **domaine** *str*: This optional parameter specifies the domain on which the data should be interpolated before it is written in the output file. The default is to write the data on the domain of the current problem (no interpolation).
- **format** *str into ['lml', 'lata', 'lata_v1', 'lata_v2', 'med', 'med_major']*: This optional parameter specifies the format of the output file. The basename used for the output file is the basename of the data file. For the fmt parameter, choices are lml or lata. A short description of each format can be found below. The default value is lml.
- **parallele** *str into ['simple', 'multiple', 'mpi-io']*: Select simple (single file, sequential write), multiple (several files, parallel write), or mpi-io (single file, parallel write) for LATA format
- **fields|champs** *champs_posts* (4.2.20): Field's write mode.
- **statistiques** *stats_posts* (4.2.23): Statistics between two points fixed : start of integration time and end of integration time.
- **fichier** *str*: Name of file.
- **statistiques_en_serie** *stats_serie_posts* (4.2.31): Statistics between two points not fixed : on period of integration.
- **interfaces** *champs_posts* (4.2.20): Keyword to read all the caracteristics of the interfaces. Different kind of interfaces exist as well as different interface intitialisations.

4.5 Liste_post

Description: Keyword to use several results files. List of objects of post-processing (with name)

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *un_postraitement_spec* (4.5.1)

4.5.1 Un_postraitement_spec

Description: An object of post-processing (with type +name).

See also: objet_lecture (33)

Usage:
[**type_un_post**] [**type_postraitement_ft_lata**]
where

- **type_un_post** *type_un_post* (4.5.2)
- **type_postraitement_ft_lata** *type_postraitement_ft_lata* (4.5.3)

4.5.2 Type_un_post

Description: not_set

See also: objet_lecture (33)

Usage:
type **post**
where

- **type** *str into ['postraitement', 'post_processing']*
- **post** *un_postraitement* (4.3.1)

4.5.3 Type_postraitement_ft_lata

Description: not_set

See also: objet_lecture (33)

Usage:
type **nom** **bloc**
where

- **type** *str into ['postraitement_ft_lata', 'postraitement_lata']*
- **nom** *str*: Name of the post-processing.
- **bloc** *str*

4.6 Format_file

Description: File formatted.

See also: objet_lecture (33)

Usage:
[**format**] **name_file**
where

- **format** *str into ['binaire', 'formatte', 'xyz', 'single_hdf']*: Type of file (the file format).
- **name_file** *str*: Name of file.

4.7 Pb_multiphase

Description: A problem that allows the resolution of N-phases with 3*N equations

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8)

Usage:
Pb_Multiphase *str*
Read *str* {

[**correlations** *bloc_lecture*]
QDM_Multiphase *qdm_multiphase*
Masse_Multiphase *masse_multiphase*
Energie_Multiphase *energie_multiphase*
[**Post_processing|postraitement** *corps_postraitement*]
[**Post_processings|postraitements** *post_processings*]
[**liste_de_postraitements** *liste_post_ok*]
[**liste_postraitements** *liste_post*]
[**sauvegarde** *format_file*]
[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **correlations** *bloc_lecture* (3.17): List of correlations used in specific source terms (i.e. interfacial flux, interfacial friction, ...)
- **QDM_Multiphase** *qdm_multiphase* (5.11): Momentum conservation equation for a multi-phase problem where the unknown is the velocity
- **Masse_Multiphase** *masse_multiphase* (5.10): Mass consevation equation for a multi-phase problem where the unknown is the alpha (void fraction)
- **Energie_Multiphase** *energie_multiphase* (5.9): Internal energy conservation equation for a multi-phase problem where the unknown is the temperature
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.8 Pb_base

Description: Resolution of equations on a domain. A problem is defined by creating an object and assigning the problem type that the user wishes to resolve. To enter values for the problem objects created, the Lire (Read) interpretor is used with a data block.

Keyword Discretize should have already been used to read the object.
See also: pb_gen_base (4) pb_thermohydraulique (4.19) pb_hydraulique (4.13) pb_hydraulique_concentration (4.14) pb_thermohydraulique_concentration (4.22) pb_post (4.18) problem_read_generic (4.29) Pb_Conduction (4.1) Pb_Multiphase (4.7) pb_avec_passif (4.11) pb_thermohydraulique_QC (4.20) pb_hydraulique_melange_binaire_QC (4.16) pb_thermohydraulique_WC (4.21) pb_hydraulique_melange_binaire_WC (4.17)

Usage:
Pb_base *str*
Read *str* {

[**Post_processing|postraitement** *corps_postraitement*]
[**Post_processings|postraitements** *post_processings*]
[**liste_de_postraitements** *liste_post_ok*]
[**liste_postraitements** *liste_post*]
[**sauvegarde** *format_file*]
[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **Post_processing|postraitement** *corps_postraitement* (4.2): One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3): List of Postraitement objects (with name).

- **liste_de_postraitements** *liste_post_ok* (4.4): This
- **liste_postraitements** *liste_post* (4.5): This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6): Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6): The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6): Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6): Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.9 Probleme_couple

Description: This instruction causes a probleme_couple type object to be created. This type of object has an associated problem list, that is, the coupling of n problems among them may be processed. Coupling between these problems is carried out explicitly via conditions at particular contact limits. Each problem may be associated either with the Associate keyword or with the Read/groupes keywords. The difference is that in the first case, the four problems exchange values then calculate their timestep, rather in the second case, the same strategy is used for all the problems listed inside one group, but the second group of problem exchange values with the first group of problems after the first group did its timestep. So, the first case may then also be written like this:
Probleme_Couple pbc
Read pbc { groupes { { pb1 , pb2 , pb3 , pb4 } } }
There is a physical environment per problem (however, the same physical environment could be common to several problems).
Each problem is resolved in a domain.
Warning : Presently, coupling requires coincident meshes. In case of non-coincident meshes, boundary condition 'paroi_contact' in VEF returns error message (see paroi_contact for correcting procedure).

See also: pb_gen_base (4)

Usage:
probleme_couple *str*
Read *str* {

[**groupes** *list_list_nom*]

}
where

- **groupes** *list_list_nom* (4.10): { groupes { { pb1 , pb2 } , { pb3 , pb4 } } }

4.10 List_list_nom

Description: pour les groupes

See also: listobj (32.5)

Usage:
{ object1 , object2 .... }
list of *list_un_pb* (32.1) separeted with ,

4.11 Pb_avec_passif

Description: Class to create a classical problem with a scalar transport equation (e.g: temperature or concentration) and an additional set of passive scalars (e.g: temperature or concentration) equations.

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8) pb_thermohydraulique_especes_QC (4.24) pb_thermohydraulique_especes_WC (4.25) pb_thermohydraulique_concentration_scalaires_passifs (4.23) pb_thermohydraulique_scalaires_passifs (4.26) pb_hydraulique_concentration_scalaires_passifs (4.15)

Usage:
pb_avec_passif *str*
Read *str* {

equations_scalaires_passifs *listeqn*
[**Post_processing|postraitement** *corps_postraitement*]
[**Post_processings|postraitements** *post_processings*]
[**liste_de_postraitements** *liste_post_ok*]
[**liste_postraitements** *liste_post*]
[**sauvegarde** *format_file*]
[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **equations_scalaires_passifs** *listeqn* (4.12): Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.

- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.12 Listeqn

Description: List of equations.

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *eqn_base* (5.21)

4.13 Pb_hydraulique

Description: Resolution of the Navier-Stokes equations.

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8)

Usage:
pb_hydraulique *str*
Read *str* {

> **navier_stokes_standard** *navier_stokes_standard*
> [**Post_processing|postraitement** *corps_postraitement*]
> [**Post_processings|postraitements** *post_processings*]
> [**liste_de_postraitements** *liste_post_ok*]
> [**liste_postraitements** *liste_post*]
> [**sauvegarde** *format_file*]

[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **navier_stokes_standard** *navier_stokes_standard* (5.27): Navier-Stokes equations.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.14 Pb_hydraulique_concentration

Description: Resolution of Navier-Stokes/multiple constituent transport equations.

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8)

Usage:
pb_hydraulique_concentration *str*
Read *str* {

[**navier_stokes_standard** *navier_stokes_standard*]
[**convection_diffusion_concentration** *convection_diffusion_concentration*]
[**Post_processing|postraitement** *corps_postraitement*]
[**Post_processings|postraitements** *post_processings*]
[**liste_de_postraitements** *liste_post_ok*]
[**liste_postraitements** *liste_post*]
[**sauvegarde** *format_file*]
[**sauvegarde_simple** *format_file*]

[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **navier_stokes_standard** *navier_stokes_standard* (5.27): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.14): Constituent transport vectorial equation (concentration diffusion convection).
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.15 Pb_hydraulique_concentration_scalaires_passifs

Description: Resolution of Navier-Stokes/multiple constituent transport equations with the additional passive scalar equations.

Keyword Discretize should have already been used to read the object.
See also: pb_avec_passif (4.11)

Usage:
pb_hydraulique_concentration_scalaires_passifs *str*
Read *str* {

[**navier_stokes_standard** *navier_stokes_standard*]
[**convection_diffusion_concentration** *convection_diffusion_concentration*]
equations_scalaires_passifs *listeqn*
[**Post_processing|postraitement** *corps_postraitement*]
[**Post_processings|postraitements** *post_processings*]
[**liste_de_postraitements** *liste_post_ok*]

[**liste_postraitements** *liste_post*]
[**sauvegarde** *format_file*]
[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **navier_stokes_standard** *navier_stokes_standard* (5.27): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.14): Constituent transport equations (concentration diffusion convection).
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.16 Pb_hydraulique_melange_binaire_qc

Description: Resolution of a binary mixture problem for a quasi-compressible fluid with an iso-thermal condition.
Keywords for the unknowns other than pressure, velocity, fraction_massique are :
masse_volumique : density
pression : reduced pressure
pression_tot : total pressure.

Keyword Discretize should have already been used to read the object.

See also: Pb_base (4.8)

Usage:
pb_hydraulique_melange_binaire_QC *str*
Read *str* {

> **navier_stokes_QC** *navier_stokes_qc*
> **convection_diffusion_espece_binaire_QC** *convection_diffusion_espece_binaire_qc*
> [**Post_processing|postraitement** *corps_postraitement*]
> [**Post_processings|postraitements** *post_processings*]
> [**liste_de_postraitements** *liste_post_ok*]
> [**liste_postraitements** *liste_post*]
> [**sauvegarde** *format_file*]
> [**sauvegarde_simple** *format_file*]
> [**reprise** *format_file*]
> [**resume_last_time** *format_file*]

}
where

- **navier_stokes_QC** *navier_stokes_qc* (5.22): Navier-Stokes equation for a quasi-compressible fluid.
- **convection_diffusion_espece_binaire_QC** *convection_diffusion_espece_binaire_qc* (5.15): Species conservation equation for a binary quasi-compressible fluid.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.17 Pb_hydraulique_melange_binaire_wc

Description: Resolution of a binary mixture problem for a weakly-compressible fluid with an iso-thermal condition.

Keywords for the unknowns other than pressure, velocity, fraction_massique are :
masse_volumique : density
pression : reduced pressure
pression_tot : total pressure
pression_hydro : hydro-static pressure
pression_eos : pressure used in state equation.

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8)

Usage:
pb_hydraulique_melange_binaire_WC *str*
Read *str* {

> **navier_stokes_WC** *navier_stokes_wc*
> **convection_diffusion_espece_binaire_WC** *convection_diffusion_espece_binaire_wc*
> [**Post_processing|postraitement** *corps_postraitement*]
> [**Post_processings|postraitements** *post_processings*]
> [**liste_de_postraitements** *liste_post_ok*]
> [**liste_postraitements** *liste_post*]
> [**sauvegarde** *format_file*]
> [**sauvegarde_simple** *format_file*]
> [**reprise** *format_file*]
> [**resume_last_time** *format_file*]

}
where

- **navier_stokes_WC** *navier_stokes_wc* (5.26): Navier-Stokes equation for a weakly-compressible fluid.
- **convection_diffusion_espece_binaire_WC** *convection_diffusion_espece_binaire_wc* (5.16): Species conservation equation for a binary weakly-compressible fluid.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.

- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.18 Pb_post

Description: not_set

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8)

Usage:
pb_post *str*
Read *str* {

> [**Post_processing|postraitement** *corps_postraitement*]
> [**Post_processings|postraitements** *post_processings*]
> [**liste_de_postraitements** *liste_post_ok*]
> [**liste_postraitements** *liste_post*]
> [**sauvegarde** *format_file*]
> [**sauvegarde_simple** *format_file*]
> [**reprise** *format_file*]
> [**resume_last_time** *format_file*]

}
where

- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.19 Pb_thermohydraulique

Description: Resolution of thermohydraulic problem.

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8)

Usage:
pb_thermohydraulique *str*
Read *str* {

[**navier_stokes_standard** *navier_stokes_standard*]
[**convection_diffusion_temperature** *convection_diffusion_temperature*]
[**Post_processing|postraitement** *corps_postraitement*]
[**Post_processings|postraitements** *post_processings*]
[**liste_de_postraitements** *liste_post_ok*]
[**liste_postraitements** *liste_post*]
[**sauvegarde** *format_file*]
[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **navier_stokes_standard** *navier_stokes_standard* (5.27): Navier-Stokes equations.
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.19): Energy equation (temperature diffusion convection).
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.20 Pb_thermohydraulique_qc

Description: Resolution of thermo-hydraulic problem for a quasi-compressible fluid.
Keywords for the unknowns other than pressure, velocity, temperature are :
masse_volumique : density
enthalpie : enthalpy
pression : reduced pressure
pression_tot : total pressure.

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8)

Usage:
pb_thermohydraulique_QC *str*
Read *str* {

> **navier_stokes_QC** *navier_stokes_qc*
> **convection_diffusion_chaleur_QC** *convection_diffusion_chaleur_qc*
> [**Post_processing|postraitement** *corps_postraitement*]
> [**Post_processings|postraitements** *post_processings*]
> [**liste_de_postraitements** *liste_post_ok*]
> [**liste_postraitements** *liste_post*]
> [**sauvegarde** *format_file*]
> [**sauvegarde_simple** *format_file*]
> [**reprise** *format_file*]
> [**resume_last_time** *format_file*]

}
where

- **navier_stokes_QC** *navier_stokes_qc* (5.22): Navier-Stokes equation for a quasi-compressible fluid.
- **convection_diffusion_chaleur_QC** *convection_diffusion_chaleur_qc* (5.12): Temperature equation for a quasi-compressible fluid.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the

name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.

- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.21 Pb_thermohydraulique_wc

Description: Resolution of thermo-hydraulic problem for a weakly-compressible fluid.
Keywords for the unknowns other than pressure, velocity, temperature are :
masse_volumique : density
pression : reduced pressure
pression_tot : total pressure
pression_hydro : hydro-static pressure
pression_eos : pressure used in state equation.

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8)

Usage:
pb_thermohydraulique_WC *str*
Read *str* {

> **navier_stokes_WC** *navier_stokes_wc*
> **convection_diffusion_chaleur_WC** *convection_diffusion_chaleur_wc*
> [**Post_processing|postraitement** *corps_postraitement*]
> [**Post_processings|postraitements** *post_processings*]
> [**liste_de_postraitements** *liste_post_ok*]
> [**liste_postraitements** *liste_post*]
> [**sauvegarde** *format_file*]
> [**sauvegarde_simple** *format_file*]
> [**reprise** *format_file*]
> [**resume_last_time** *format_file*]

}
where

- **navier_stokes_WC** *navier_stokes_wc* (5.26): Navier-Stokes equation for a weakly-compressible fluid.
- **convection_diffusion_chaleur_WC** *convection_diffusion_chaleur_wc* (5.13): Temperature equation for a weakly-compressible fluid.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.

- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.22 Pb_thermohydraulique_concentration

Description: Resolution of Navier-Stokes/energy/multiple constituent transport equations.

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8)

Usage:
pb_thermohydraulique_concentration *str*
Read *str* {

[**navier_stokes_standard** *navier_stokes_standard*]
[**convection_diffusion_concentration** *convection_diffusion_concentration*]
[**convection_diffusion_temperature** *convection_diffusion_temperature*]
[**Post_processing|postraitement** *corps_postraitement*]
[**Post_processings|postraitements** *post_processings*]
[**liste_de_postraitements** *liste_post_ok*]
[**liste_postraitements** *liste_post*]
[**sauvegarde** *format_file*]
[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **navier_stokes_standard** *navier_stokes_standard* (5.27): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.14): Constituent transport equations (concentration diffusion convection).
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.19): Energy equation (temperature diffusion convection).
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.

- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.23 Pb_thermohydraulique_concentration_scalaires_passifs

Description: Resolution of Navier-Stokes/energy/multiple constituent transport equations, with the additional passive scalar equations.

Keyword Discretize should have already been used to read the object.
See also: pb_avec_passif (4.11)

Usage:
pb_thermohydraulique_concentration_scalaires_passifs *str*
Read *str* {

[**navier_stokes_standard** *navier_stokes_standard*]
[**convection_diffusion_concentration** *convection_diffusion_concentration*]
[**convection_diffusion_temperature** *convection_diffusion_temperature*]
equations_scalaires_passifs *listeqn*
[**Post_processing|postraitement** *corps_postraitement*]
[**Post_processings|postraitements** *post_processings*]
[**liste_de_postraitements** *liste_post_ok*]
[**liste_postraitements** *liste_post*]
[**sauvegarde** *format_file*]
[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **navier_stokes_standard** *navier_stokes_standard* (5.27): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.14): Constituent transport equations (concentration diffusion convection).
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.19): Energy equations (temperature diffusion convection).
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.

- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.24 Pb_thermohydraulique_especes_qc

Description: Resolution of thermo-hydraulic problem for a multi-species quasi-compressible fluid.

Keyword Discretize should have already been used to read the object.
See also: pb_avec_passif (4.11)

Usage:
pb_thermohydraulique_especes_QC *str*
Read *str* {

> **navier_stokes_QC** *navier_stokes_qc*
> **convection_diffusion_chaleur_QC** *convection_diffusion_chaleur_qc*
> **equations_scalaires_passifs** *listeqn*
> [**Post_processing|postraitement** *corps_postraitement*]
> [**Post_processings|postraitements** *post_processings*]
> [**liste_de_postraitements** *liste_post_ok*]
> [**liste_postraitements** *liste_post*]
> [**sauvegarde** *format_file*]
> [**sauvegarde_simple** *format_file*]
> [**reprise** *format_file*]
> [**resume_last_time** *format_file*]

}
where

- **navier_stokes_QC** *navier_stokes_qc* (5.22): Navier-Stokes equation for a quasi-compressible fluid.

- **convection_diffusion_chaleur_QC** *convection_diffusion_chaleur_qc* (5.12): Temperature equation for a quasi-compressible fluid.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.25 Pb_thermohydraulique_especes_wc

Description: Resolution of thermo-hydraulic problem for a multi-species weakly-compressible fluid.

Keyword Discretize should have already been used to read the object.
See also: pb_avec_passif (4.11)

Usage:
pb_thermohydraulique_especes_WC *str*
Read *str* {

> **navier_stokes_WC** *navier_stokes_wc*
> **convection_diffusion_chaleur_WC** *convection_diffusion_chaleur_wc*
> **equations_scalaires_passifs** *listeqn*
> [**Post_processing|postraitement** *corps_postraitement*]
> [**Post_processings|postraitements** *post_processings*]
> [**liste_de_postraitements** *liste_post_ok*]
> [**liste_postraitements** *liste_post*]
> [**sauvegarde** *format_file*]

[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **navier_stokes_WC** *navier_stokes_wc* (5.26): Navier-Stokes equation for a weakly-compressible fluid.
- **convection_diffusion_chaleur_WC** *convection_diffusion_chaleur_wc* (5.13): Temperature equation for a weakly-compressible fluid.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.26 Pb_thermohydraulique_scalaires_passifs

Description: Resolution of thermohydraulic problem, with the additional passive scalar equations.

Keyword Discretize should have already been used to read the object.
See also: pb_avec_passif (4.11)

Usage:
pb_thermohydraulique_scalaires_passifs *str*
Read *str* {

[**navier_stokes_standard** *navier_stokes_standard*]
[**convection_diffusion_temperature** *convection_diffusion_temperature*]
equations_scalaires_passifs *listeqn*
[**Post_processing|postraitement** *corps_postraitement*]
[**Post_processings|postraitements** *post_processings*]
[**liste_de_postraitements** *liste_post_ok*]
[**liste_postraitements** *liste_post*]
[**sauvegarde** *format_file*]
[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **navier_stokes_standard** *navier_stokes_standard* (5.27): Navier-Stokes equations.
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.19): Energy equations (temperature diffusion convection).
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.27 Pbc_med

Description: Allows to read med files and post-process them.

See also: pb_gen_base (4)

Usage:
pbc_med list_info_med
where

- **list_info_med** *list_info_med* (4.28)

4.28 List_info_med

Description: not_set

See also: listobj (32.5)

Usage:
{ object1 , object2 .... }
list of *info_med* (4.28.1) separeted with ,

4.28.1 Info_med

Description: not_set

See also: objet_lecture (33)

Usage:
file_med domaine pb_post
where

- **file_med** *str*: Name of the MED file.
- **domaine** *str*: Name of domain.
- **pb_post** *pb_post* (4.18)

4.29 Problem_read_generic

Description: The probleme_read_generic differs rom the rest of the TRUST code : The problem does not state the number of equations that are enclosed in the problem. As the list of equations to be solved in the generic read problem is declared in the data file and not pre-defined in the structure of the problem, each equation has to be distinctively associated with the problem with the Associate keyword.

Keyword Discretize should have already been used to read the object.
See also: Pb_base (4.8)

Usage:
problem_read_generic *str*
Read *str* {

[**Post_processing|postraitement** *corps_postraitement*]
[**Post_processings|postraitements** *post_processings*]
[**liste_de_postraitements** *liste_post_ok*]
[**liste_postraitements** *liste_post*]
[**sauvegarde** *format_file*]
[**sauvegarde_simple** *format_file*]
[**reprise** *format_file*]
[**resume_last_time** *format_file*]

}
where

- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

5 mor_eqn

Description: Class of equation pieces (morceaux d'equation).

See also: objet_u (34) eqn_base (5.21)

Usage:

5.1 Conduction

Description: Heat equation.

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
Conduction *str*
Read *str* {

[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]

[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format:
  n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.2 Bloc_convection

Description: not_set

See also: objet_lecture (33)

Usage:
aco operateur acof
where

- **aco** *str into ['{']: Opening curly bracket.*
- **operateur** *convection_deriv (5.2.1)*
- **acof** *str into ['}']* : Closing curly bracket.

5.2.1 Convection_deriv

Description: not_set

See also: objet_lecture (33) amont (5.2.2) amont_old (5.2.3) centre (5.2.4) centre4 (5.2.5) centre_old (5.2.6) di_l2 (5.2.7) ef (5.2.8) muscl3 (5.2.10) ef_stab (5.2.11) generic (5.2.14) kquick (5.2.15) muscl (5.2.16) muscl_old (5.2.17) muscl_new (5.2.18) negligeable (5.2.19) quick (5.2.20) ale (5.2.21) btd (5.2.22) supg (5.2.23)

Usage:
convection_deriv

5.2.2 Amont

Description: Keyword for upwind scheme for VDF or VEF discretizations. In VEF discretization equivalent to generic amont for TRUST version 1.5 or later. The previous upwind scheme can be used with the obsolete in future amont_old keyword.

See also: convection_deriv (5.2.1)

Usage:
amont

5.2.3 Amont_old

Description: Only for VEF discretization, obsolete keyword, see amont.

See also: convection_deriv (5.2.1)

Usage:
amont_old

5.2.4 Centre

Description: For VDF and VEF discretizations.

See also: convection_deriv (5.2.1)

Usage:
centre

5.2.5 Centre4

Description: For VDF and VEF discretizations.

See also: convection_deriv (5.2.1)

Usage:
centre4

5.2.6 Centre_old

Description: Only for VEF discretization.

See also: convection_deriv (5.2.1)

Usage:
centre_old

5.2.7 Di_l2

Description: Only for VEF discretization.

See also: convection_deriv (5.2.1)

Usage:
di_l2

5.2.8 Ef

Description: For VEF calculations, a centred convective scheme based on Finite Elements formulation can be called through the following data:

Convection { EF transportant_bar val transporte_bar val antisym val filtrer_resu val }

This scheme is 2nd order accuracy (and get better the property of kinetic energy conservation). Due to possible problems of instabilities phenomena, this scheme has to be coupled with stabilisation process (see Source_Qdm_lambdaup).These two last data are equivalent from a theoretical point of view in variationnal writing to : div((u. grad ub , vb) - (u. grad vb, ub)), where vb corresponds to the filtered reference test functions.

Remark:
This class requires to define a filtering operator : see solveur_bar

See also: convection_deriv (5.2.1)

Usage:
ef [**mot1**] [**bloc_ef**]
where

- **mot1** *str into ['defaut_bar']*: equivalent to transportant_bar 0 transporte_bar 1 filtrer_resu 1 antisym 1
- **bloc_ef** *bloc_ef* (5.2.9)

5.2.9 Bloc_ef

Description: not_set

See also: objet_lecture (33)

Usage:
mot1 val1 mot2 val2 mot3 val3 mot4 val4
where

- **mot1** *str into ['transportant_bar', 'transporte_bar', 'filtrer_resu', 'antisym']*
- **val1** *int into [0, 1]*
- **mot2** *str into ['transportant_bar', 'transporte_bar', 'filtrer_resu', 'antisym']*
- **val2** *int into [0, 1]*
- **mot3** *str into ['transportant_bar', 'transporte_bar', 'filtrer_resu', 'antisym']*
- **val3** *int into [0, 1]*
- **mot4** *str into ['transportant_bar', 'transporte_bar', 'filtrer_resu', 'antisym']*
- **val4** *int into [0, 1]*

5.2.10 Muscl3

Description: Keyword for a scheme using a ponderation between muscl and center schemes in VEF.

See also: convection_deriv (5.2.1)

Usage:
muscl3 {

> [**alpha** *float*]

}
where

- **alpha** *float*: To weight the scheme centering with the factor double (between 0 (full centered) and 1 (muscl), by default 1).

5.2.11 Ef_stab

Description: Keyword for a VEF convective scheme.

See also: convection_deriv (5.2.1)

Usage:
ef_stab {

> [**alpha** *float*]
> [**test** *int*]
> [**tdivu**]
> [**old**]
> [**volumes_etendus**]
> [**volumes_non_etendus**]
> [**amont_sous_zone** *str*]
> [**alpha_sous_zone** *listsous_zone_valeur*]

}
where

- **alpha** *float*: To weight the scheme centering with the factor double (between 0 (full centered) and 1 (mix between upwind and centered), by default 1). For scalar equation, it is adviced to use alpha=1 and for the momentum equation, alpha=0.2 is adviced.
- **test** *int*: Developer option to compare old and new version of EF_stab
- **tdivu** : To have the convective operator calculated as div(TU)-TdivU(=UgradT).
- **old** : To use old version of EF_stab scheme (default no).
- **volumes_etendus** : Option for the scheme to use the extended volumes (default, yes).
- **volumes_non_etendus** : Option for the scheme to not use the extended volumes (default, no).
- **amont_sous_zone** *str*: Option to degenerate EF_stab scheme into Amont (upwind) scheme in the sub zone of name sz_name. The sub zone may be located arbitrarily in the domain but the more often this option will be activated in a zone where EF_stab scheme generates instabilities as for free outlet for example.
- **alpha_sous_zone** *listsous_zone_valeur* (5.2.12): Option to change locally the alpha value on N sub-zones named sub_zone_name_I. Generally, it is used to prevent from a local divergence by increasing locally the alpha parameter.

5.2.12 Listsous_zone_valeur

Description: List of groups of two words.

See also: listobj (32.5)

Usage:
n object1 object2 ....
list of *sous_zone_valeur* (5.2.13)

5.2.13 Sous_zone_valeur

Description: Two words.

See also: objet_lecture (33)

Usage:
sous_zone valeur
where

- **sous_zone** *str*: sous zone
- **valeur** *float*: value

5.2.14 Generic

Description: Keyword for generic calling of upwind and muscl convective scheme in VEF discretization. For muscl scheme, limiters and order for fluxes calculations have to be specified. The available limiters are : minmod - vanleer -vanalbada - chakravarthy - superbee, and the order of accuracy is 1 or 2. Note that chakravarthy is a non-symmetric limiter and superbee may engender results out of physical limits. By consequence, these two limiters are not recommended.
Examples:
convection { generic amont }
convection { generic muscl minmod 1 }
convection { generic muscl vanleer 2 }

In case of results out of physical limits with muscl scheme (due for instance to strong non-conformal velocity flow field), user can redefine in data file a lower order and a smoother limiter, as : convection { generic muscl minmod 1 }

See also: convection_deriv (5.2.1)

Usage:
generic type [**limiteur**] [**ordre**] [**alpha**]
where

- **type** *str into ['amont', 'muscl', 'centre']*: type of scheme
- **limiteur** *str into ['minmod', 'vanleer', 'vanalbada', 'chakravarthy', 'superbee']*: type of limiter
- **ordre** *int into [1, 2, 3]*: order of accuracy
- **alpha** *float*: alpha

5.2.15 Kquick

Description: Only for VEF discretization.

See also: convection_deriv (5.2.1)

Usage:
kquick

5.2.16 Muscl

Description: Keyword for muscl scheme in VEF discretization equivalent to generic muscl vanleer 2 for the 1.5 version or later. The previous muscl scheme can be used with the obsolete in future muscl_old keyword.

See also: convection_deriv (5.2.1)

Usage:
muscl

5.2.17 Muscl_old

Description: Only for VEF discretization.

See also: convection_deriv (5.2.1)

Usage:
muscl_old

5.2.18 Muscl_new

Description: Only for VEF discretization.

See also: convection_deriv (5.2.1)

Usage:
muscl_new

5.2.19 Negligeable

Description: For VDF and VEF discretizations. Suppresses the convection operator.

See also: convection_deriv (5.2.1)

Usage:
negligeable

5.2.20 Quick

Description: Only for VDF discretization.

See also: convection_deriv (5.2.1)

Usage:
quick

5.2.21 Ale

Description: A convective scheme for ALE (Arbitrary Lagrangian-Eulerian) framework.

See also: convection_deriv (5.2.1)

Usage:
ale opconv
where

- **opconv** *bloc_convection* (5.2): Choice between: amont and muscl
  Example: convection { ALE { amont } }

5.2.22 Btd

Description: Only for EF discretization.

See also: convection_deriv (5.2.1)

Usage:
btd {

> **btd** *float*
> **facteur** *float*

}
where

- **btd** *float*
- **facteur** *float*

5.2.23 Supg

Description: Only for EF discretization.

See also: convection_deriv (5.2.1)

Usage:
supg {

> **facteur** *float*

}
where

- **facteur** *float*

5.3 Bloc_diffusion

Description: not_set

See also: objet_lecture (33)

Usage:
aco [**operateur**] [**op_implicite**] **acof**
where

- **aco** *str into ['{']: Opening curly bracket.*
- **operateur** *diffusion_deriv (5.3.1): if none is specified, the diffusive scheme used is a 2nd-order scheme.*
- **op_implicite** *op_implicite (5.3.9): To have diffusive implicitation, it use Uzawa algorithm. Very useful when viscosity has large variations.*
- **acof** *str into ['}']* : Closing curly bracket.

5.3.1 Diffusion_deriv

Description: not_set

See also: objet_lecture (33) negligeable (5.3.2) p1b (5.3.3) p1ncp1b (5.3.4) stab (5.3.5) standard (5.3.6) option (5.3.8)

Usage:
diffusion_deriv

5.3.2 Negligeable

Description: the diffusivity will not taken in count

See also: diffusion_deriv (5.3.1)

Usage:
negligeable

5.3.3 P1b

Description: not_set

See also: diffusion_deriv (5.3.1)

Usage:
p1b

5.3.4 P1ncp1b

Description: not_set

See also: diffusion_deriv (5.3.1)

Usage:

5.3.5 Stab

Description: keyword allowing consistent and stable calculations even in case of obtuse angle meshes.

See also: diffusion_deriv (5.3.1)

Usage:
stab {

  [**standard** *int*]
  [**info** *int*]

[**new_jacobian** *int*]
[**nu** *int*]
[**nut** *int*]
[**nu_transp** *int*]
[**nut_transp** *int*]

}
where

- **standard** *int*: to recover the same results as calculations made by standard laminar diffusion operator. However, no stabilization technique is used and calculations may be unstable when working with obtuse angle meshes (by default 0)
- **info** *int*: developer option to get the stabilizing ratio (by default 0)
- **new_jacobian** *int*: when implicit time schemes are used, this option defines a new jacobian that may be more suitable to get stationary solutions (by default 0)
- **nu** *int*: (respectively nut 1) takes the molecular viscosity (resp. eddy viscosity) into account in the velocity gradient part of the diffusion expression (by default nu=1 and nut=1)
- **nut** *int*
- **nu_transp** *int*: (respectively nut_transp 1) takes the molecular viscosity (resp. eddy viscosity) into account in the transposed velocity gradient part of the diffusion expression (by default nu_transp=0 and nut_transp=1)
- **nut_transp** *int*

5.3.6 Standard

Description: A new keyword, intended for LES calculations, has been developed to optimise and parameterise each term of the diffusion operator. Remark:

1. This class requires to define a filtering operator : see solveur_bar
2. The former (original) version: diffusion { } -which omitted some of the term of the diffusion operator- can be recovered by using the following parameters in the new class :
diffusion { standard grad_Ubar 0 nu 1 nut 1 nu_transp 0 nut_transp 1 filtrer_resu 0}.

See also: diffusion_deriv (5.3.1)

Usage:
standard [**mot1**] [**bloc_diffusion_standard**]
where

- **mot1** *str into ['defaut_bar']*: equivalent to grad_Ubar 1 nu 1 nut 1 nu_transp 1 nut_transp 1 filtrer_resu 1
- **bloc_diffusion_standard** *bloc_diffusion_standard* (5.3.7)

5.3.7 Bloc_diffusion_standard

Description: grad_Ubar 1 makes the gradient calculated through the filtered values of velocity (P1-conform).
nu 1 (respectively nut 1) takes the molecular viscosity (eddy viscosity) into account in the velocity gradient part of the diffusion expression.
nu_transp 1 (respectively nut_transp 1) takes the molecular viscosity (eddy viscosity) into account according in the TRANSPOSED velocity gradient part of the diffusion expression.
filtrer_resu 1 allows to filter the resulting diffusive fluxes contribution.

See also: objet_lecture (33)

Usage:

mot1 val1 mot2 val2 mot3 val3 mot4 val4 mot5 val5 mot6 val6
where

- **mot1** *str into ['grad_Ubar', 'nu', 'nut', 'nu_transp', 'nut_transp', 'filtrer_resu']*
- **val1** *int into [0, 1]*
- **mot2** *str into ['grad_Ubar', 'nu', 'nut', 'nu_transp', 'nut_transp', 'filtrer_resu']*
- **val2** *int into [0, 1]*
- **mot3** *str into ['grad_Ubar', 'nu', 'nut', 'nu_transp', 'nut_transp', 'filtrer_resu']*
- **val3** *int into [0, 1]*
- **mot4** *str into ['grad_Ubar', 'nu', 'nut', 'nu_transp', 'nut_transp', 'filtrer_resu']*
- **val4** *int into [0, 1]*
- **mot5** *str into ['grad_Ubar', 'nu', 'nut', 'nu_transp', 'nut_transp', 'filtrer_resu']*
- **val5** *int into [0, 1]*
- **mot6** *str into ['grad_Ubar', 'nu', 'nut', 'nu_transp', 'nut_transp', 'filtrer_resu']*
- **val6** *int into [0, 1]*

5.3.8 Option

Description: not_set

See also: diffusion_deriv (5.3.1)

Usage:
option bloc_lecture
where

- **bloc_lecture** *bloc_lecture* (3.17)

5.3.9 Op_implicite

Description: not_set

See also: objet_lecture (33)

Usage:
implicite mot solveur
where

- **implicite** *str into ['implicite']*
- **mot** *str into ['solveur']*
- **solveur** *solveur_sys_base* (9.13)

5.4 Condinits

Description: Initial conditions.

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *condinit* (5.4.1)

5.4.1 Condinit

Description: Initial condition.

See also: objet_lecture (33)

Usage:
nom ch
where

- **nom** *str*: Name of initial condition field.
- **ch** *champ_base* (14.1): Type field and the initial values.

5.5 Condlims

Description: Boundary conditions.

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *condlimlu* (5.5.1)

5.5.1 Condlimlu

Description: Boundary condition specified.

See also: objet_lecture (33)

Usage:
bord cl
where

- **bord** *str*: Name of the edge where the boundary condition applies.
- **cl** *condlim_base* (11): Boundary condition at the boundary called bord (edge).

5.6 Sources

Description: The sources.

See also: listobj (32.5)

Usage:
{ object1 , object2 .... }
list of *source_base* (28) separeted with ,

5.7 Ecrire_fichier_xyz_valeur_param

Description: not_set

Keyword Discretize should have already been used to read the object.
See also: listobj (32.5)

Usage:

n object1 , object2 ....
list of *ecrire_fichier_xyz_valeur_item* (5.7.1) separeted with ,

5.7.1 Ecrire_fichier_xyz_valeur_item

Description: To write the values of a field for some boundaries in a text file.
The name of the files is pb_name_field_name_time.dat
Several Ecrire_fichier_xyz_valeur keywords may be written into an equation to write several fields. This kind of files may be read by Champ_don_lu or Champ_front_lu for example.

See also: objet_lecture (33)

Usage:
name **dt_ecrire_fic** [**bords**]
where

- **name** *str*: Name of the field to write (Champ_Inc, Champ_Fonc or a post_processed field).
- **dt_ecrire_fic** *float*: Time period for printing in the file.
- **bords** *bords_ecrire* (5.7.2): to post-process only on some boundaries

5.7.2 Bords_ecrire

Description: not_set

See also: objet_lecture (33)

Usage:
chaine **bords**
where

- **chaine** *str into ['bords']*
- **bords** *n word1 word2 ... wordn*: Keyword to post-process only on some boundaries :
  bords nb_bords boundary1 ... boundaryn
  where
  nb_bords : number of boundaries
  boundary1 ... boundaryn : name of the boundaries.

5.8 Parametre_equation_base

Description: Basic class for parametre_equation

See also: objet_lecture (33) parametre_implicite (5.8.1) parametre_diffusion_implicite (5.8.2)

Usage:

5.8.1 Parametre_implicite

Description: Keyword to change for this equation only the parameter of the implicit scheme used to solve the problem.

See also: parametre_equation_base (5.8)

Usage:
parametre_implicite {

[**seuil_convergence_implicite** *float*]
[**seuil_convergence_solveur** *float*]
[**solveur** *solveur_sys_base*]
[**resolution_explicite**]
[**equation_non_resolue**]
[**equation_frequence_resolue** *str*]

}
where

- **seuil_convergence_implicite** *float*: Keyword to change for this equation only the value of seuil-_convergence_implicite used in the implicit scheme.
- **seuil_convergence_solveur** *float*: Keyword to change for this equation only the value of seuil-_convergence_solveur used in the implicit scheme
- **solveur** *solveur_sys_base* (9.13): Keyword to change for this equation only the solver used in the implicit scheme
- **resolution_explicite** : To solve explicitly the equation whereas the scheme is an implicit scheme.
- **equation_non_resolue** : Keyword to specify that the equation is not solved.
- **equation_frequence_resolue** *str*: Keyword to specify that the equation is solved only every n time steps (n is an integer or given by a time-dependent function f(t)).

5.8.2 Parametre_diffusion_implicite

Description: To specify additional parameters for the equation when using impliciting diffusion

See also: parametre_equation_base (5.8)

Usage:
parametre_diffusion_implicite {

[**crank** *int into [0, 1]*]
[**preconditionnement_diag** *int into [0, 1]*]
[**niter_max_diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**solveur** *solveur_sys_base*]

}
where

- **crank** *int into [0, 1]*: Use (1) or not (0, default) a Crank Nicholson method for the diffusion implicitation algorithm. Setting crank to 1 increases the order of the algorithm from 1 to 2.
- **preconditionnement_diag** *int into [0, 1]*: The CG used to solve the implicitation of the equation diffusion operator is not preconditioned by default. If this option is set to 1, a diagonal preconditionning is used. Warning: this option is not necessarily more efficient, depending on the treated case.
- **niter_max_diffusion_implicite** *int*: Change the maximum number of iterations for the CG (Conjugate Gradient) algorithm when solving the diffusion implicitation of the equation.
- **seuil_diffusion_implicite** *float*: Change the threshold convergence value used by default for the CG resolution for the diffusion implicitation of this equation.
- **solveur** *solveur_sys_base* (9.13): Method (different from the default one, Conjugate Gradient) to solve the linear system.

5.9 Energie_multiphase

Description: Internal energy conservation equation for a multi-phase problem where the unknown is the temperature

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
Energie_Multiphase *str*
Read *str* {

> [**convection** *bloc_convection*]
> [**diffusion** *bloc_diffusion*]
> [**initial_conditions|conditions_initiales** *condinits*]
> [**boundary_conditions|conditions_limites** *condlims*]
> [**sources** *sources*]
> [**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
> [**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
> [**parametre_equation** *parametre_equation_base*]
> [**equation_non_resolue** *str*]

}
where

- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format:
  n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.10 Masse_multiphase

Description: Mass consevation equation for a multi-phase problem where the unknown is the alpha (void fraction)

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
Masse_Multiphase *str*
Read *str* {

> [**convection** *bloc_convection*]
> [**diffusion** *bloc_diffusion*]
> [**initial_conditions|conditions_initiales** *condinits*]
> [**boundary_conditions|conditions_limites** *condlims*]
> [**sources** *sources*]
> [**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
> [**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
> [**parametre_equation** *parametre_equation_base*]
> [**equation_non_resolue** *str*]

}
where

- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format:
  n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.11 Qdm_multiphase

Description: Momentum conservation equation for a multi-phase problem where the unknown is the velocity

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
QDM_Multiphase *str*
Read *str* {

> [**solveur_pression** *solveur_sys_base*]
> [**evanescence** *bloc_lecture*]
> [**convection** *bloc_convection*]
> [**diffusion** *bloc_diffusion*]
> [**initial_conditions|conditions_initiales** *condinits*]
> [**boundary_conditions|conditions_limites** *condlims*]
> [**sources** *sources*]
> [**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
> [**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
> [**parametre_equation** *parametre_equation_base*]
> [**equation_non_resolue** *str*]

}
where

- **solveur_pression** *solveur_sys_base* (9.13): Linear pressure system resolution method.
- **evanescence** *bloc_lecture* (3.17): Management of the vanishing phase (when alpha tends to 0 or 1)
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format:
  n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.

Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.12 Convection_diffusion_chaleur_qc

Description: Temperature equation for a quasi-compressible fluid.

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
convection_diffusion_chaleur_QC *str*
Read *str* {

[**mode_calcul_convection** *str into ['ancien', 'divuT_moins_Tdivu', 'divrhouT_moins_Tdivrhou']*]
[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **mode_calcul_convection** *str into ['ancien', 'divuT_moins_Tdivu', 'divrhouT_moins_Tdivrhou']*: Option to set the form of the convective operator
divrhouT_moins_Tdivrhou (the default since 1.6.8): rho.u.gradT = div(rho.u.T)- Tdiv(rho.u.1)
ancien: u.gradT = div(u.T) - T.div(u)
divuT_moins_Tdivu : u.gradT = div(u.T) - Tdiv(u.1)
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format:
n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.13 Convection_diffusion_chaleur_wc

Description: Temperature equation for a weakly-compressible fluid.

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
convection_diffusion_chaleur_WC *str*
Read *str* {

[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format:
  n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.14 Convection_diffusion_concentration

Description: Constituent transport vectorial equation (concentration diffusion convection).

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
convection_diffusion_concentration *str*
Read *str* {

[**nom_inconnue** *str*]
[**masse_molaire** *float*]
[**alias** *str*]
[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **nom_inconnue** *str*: Keyword Nom_inconnue will rename the unknown of this equation with the given name. In the postprocessing part, the concentration field will be accessible with this name. This is usefull if you want to track more than one concentration (otherwise, only the concentration field in the first concentration equation can be accessed).
- **masse_molaire** *float*
- **alias** *str*
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format:
  n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.15 Convection_diffusion_espece_binaire_qc

Description: Species conservation equation for a binary quasi-compressible fluid.

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
convection_diffusion_espece_binaire_QC *str*
Read *str* {

[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.16 Convection_diffusion_espece_binaire_wc

Description: Species conservation equation for a binary weakly-compressible fluid.

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
convection_diffusion_espece_binaire_WC *str*
Read *str* {

[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.17 Convection_diffusion_espece_multi_qc

Description: Species conservation equation for a multi-species quasi-compressible fluid.

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
convection_diffusion_espece_multi_QC *str*
Read *str* {

[**espece** *espece*]
[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **espece** *espece* (3.36): Assosciate a species (with its properties) to the equation
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format:
  n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.18 Convection_diffusion_espece_multi_wc

Description: Species conservation equation for a multi-species weakly-compressible fluid.

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
convection_diffusion_espece_multi_WC *str*
Read *str* {

[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.19 Convection_diffusion_temperature

Description: Energy equation (temperature diffusion convection).

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21)

Usage:
convection_diffusion_temperature *str*
Read *str* {

[**penalisation_l2_ftd** *pp*]
[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **penalisation_l2_ftd** *pp* (5.20): to activate or not (the default is Direct Forcing method) the Penalized Direct Forcing method to impose the specified temperature on the solid-fluid interface.
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...

x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.20 Pp

Description: not_set

See also: listobj (32.5)

Usage:
{ object1 object2 .... }
list of *penalisation_l2_ftd_lec* (5.20.1)

5.20.1 Penalisation_l2_ftd_lec

Description: not_set

See also: objet_lecture (33)

Usage:
[**postraiter_gradient_pression_sans_masse**] [**correction_matrice_projection_initiale**] [**correction_calcul_pression_initiale**] [**correction_vitesse_projection_initiale**] [**correction_matrice_pression**] [**matrice_pression_penalisee_H1**] [**correction_vitesse_modifie**] [**correction_pression_modifie**] [**gradient_pression_qdm_modifie**] **bord** **val**
where

- **postraiter_gradient_pression_sans_masse** *int*: (IBM advanced) avoid mass matrix multiplication for the gradient postprocessing
- **correction_matrice_projection_initiale** *int*: (IBM advanced) fix matrix of initial projection for PDF
- **correction_calcul_pression_initiale** *int*: (IBM advanced) fix initial pressure computation for PDF
- **correction_vitesse_projection_initiale** *int*: (IBM advanced) fix initial velocity computation for PDF
- **correction_matrice_pression** *int*: (IBM advanced) fix pressure matrix for PDF
- **matrice_pression_penalisee_H1** *int*: (IBM advanced) fix pressure matrix for PDF
- **correction_vitesse_modifie** *int*: (IBM advanced) fix velocity for PDF
- **correction_pression_modifie** *int*: (IBM advanced) fix pressure for PDF
- **gradient_pression_qdm_modifie** *int*: (IBM advanced) fix pressure gradient
- **bord** *str*
- **val** *n x1 x2 ... xn*

5.21 Eqn_base

Description: Basic class for equations.

Keyword Discretize should have already been used to read the object.
See also: mor_eqn (5) navier_stokes_standard (5.27) convection_diffusion_temperature (5.19) convection-_diffusion_concentration (5.14) Conduction (5.1) QDM_Multiphase (5.11) Masse_Multiphase (5.10) Energie-_Multiphase (5.9) convection_diffusion_chaleur_QC (5.12) convection_diffusion_chaleur_WC (5.13) convection-_diffusion_espece_multi_QC (5.17) convection_diffusion_espece_binaire_QC (5.15) convection_diffusion-_espece_binaire_WC (5.16) convection_diffusion_espece_multi_WC (5.18)

Usage:
eqn_base *str*
Read *str* {

[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **convection** *bloc_convection* (5.2): Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3): Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4): Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5): Boundary conditions.
- **sources** *sources* (5.6): To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7): This keyword is used to write the values of a field only for some boundaries in a text file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7): This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n-_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8): Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str*: The equation will not be solved while condition(t) is verified if equation-_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.22 Navier_stokes_qc

Description: Navier-Stokes equation for a quasi-compressible fluid.

Keyword Discretize should have already been used to read the object.
See also: navier_stokes_standard (5.27)

Usage:
navier_stokes_QC *str*
Read *str* {

[**methode_calcul_pression_initiale** *str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']*]
[**projection_initiale** *int*]
[**solveur_pression** *solveur_sys_base*]
[**solveur_bar** *solveur_sys_base*]
[**dt_projection** *deuxmots*]
[**seuil_divU** *floatfloat*]
[**traitement_particulier** *traitement_particulier*]
[**correction_matrice_projection_initiale** *int*]
[**correction_calcul_pression_initiale** *int*]
[**correction_vitesse_projection_initiale** *int*]
[**correction_matrice_pression** *int*]
[**correction_vitesse_modifie** *int*]
[**gradient_pression_qdm_modifie** *int*]
[**correction_pression_modifie** *int*]
[**postraiter_gradient_pression_sans_masse**]
[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **methode_calcul_pression_initiale** *str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']* for inheritance: Keyword to select an option for the pressure calculation before the fist time step. Options are : avec_les_cl (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), avec_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicited when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (9.13) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (9.13) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source_Qdm_lambdaup). A file (solveur.bar) is then created and used for inversion procedure. Syntax is

the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).

- **dt_projection** *deuxmots* (5.23) for inheritance: nb value : This keyword checks every nb time-steps the equality of velocity divergence to zero. value is the criteria convergency for the solver used.
- **seuil_divU** *floatfloat* (5.24) for inheritance: value factor : this keyword is intended to minimise the number of iterations during the pressure system resolution. The convergence criteria during this step ('seuil' in solveur_pression) is dynamically adapted according to the mass conservation. At tn , the linear system Ax=B is considered as solved if the residual ||Ax-B||<seuil(tn). For tn+1, the threshold value seuil(tn+1) will be evualated as:
  If (|max(DivU)*dt|<value)
  Seuil(tn+1)= Seuil(tn)*factor
  Else
  Seuil(tn+1)= Seuil(tn)*factor
  Endif
  The first parameter (value) is the mass evolution the user is ready to accept per timestep, and the second one (factor) is the factor of evolution for 'seuil' (for example 1.1, so 10
- **traitement_particulier** *traitement_particulier* (5.25) for inheritance: Keyword to post-process particular values.
- **correction_matrice_projection_initiale** *int* for inheritance: (IBM advanced) fix matrix of initial projection for PDF
- **correction_calcul_pression_initiale** *int* for inheritance: (IBM advanced) fix initial pressure computation for PDF
- **correction_vitesse_projection_initiale** *int* for inheritance: (IBM advanced) fix initial velocity computation for PDF
- **correction_matrice_pression** *int* for inheritance: (IBM advanced) fix pressure matrix for PDF
- **correction_vitesse_modifie** *int* for inheritance: (IBM advanced) fix velocity for PDF
- **gradient_pression_qdm_modifie** *int* for inheritance: (IBM advanced) fix pressure gradient
- **correction_pression_modifie** *int* for inheritance: (IBM advanced) fix pressure for PDF
- **postraiter_gradient_pression_sans_masse** for inheritance: (IBM advanced) avoid mass matrix multiplication for the gradient postprocessing
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format:
  n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is

verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.23 Deuxmots

Description: Two words.

See also: objet_lecture (33)

Usage:
mot_1 mot_2
where

- **mot_1** *str*: First word.
- **mot_2** *str*: Second word.

5.24 Floatfloat

Description: Two reals.

See also: objet_lecture (33)

Usage:
a b
where

- **a** *float*: First real.
- **b** *float*: Second real.

5.25 Traitement_particulier

Description: Auxiliary class to post-process particular values.

See also: objet_lecture (33)

Usage:
aco trait_part acof
where

- **aco** *str into ['{']: Opening curly bracket.*
- **trait_part** *traitement_particulier_base (5.25.1): Type of traitement_particulier.*
- **acof** *str into ['}']* : Closing curly bracket.

5.25.1 Traitement_particulier_base

Description: Basic class to post-process particular values.

See also: objet_lecture (33) temperature (5.25.2) canal (5.25.3) ec (5.25.4) thi (5.25.5) chmoy_faceperio (5.25.6)

Usage:

5.25.2 Temperature

Description: not_set

See also: traitement_particulier_base (5.25.1)

Usage:
temperature {

> **bord** *str*
> **direction** *int*

}
where

- **bord** *str*
- **direction** *int*

5.25.3 Canal

Description: Keyword for statistics on a periodic plane channel.

See also: traitement_particulier_base (5.25.1)

Usage:
canal {

> [**dt_impr_moy_spat** *float*]
> [**dt_impr_moy_temp** *float*]
> [**debut_stat** *float*]
> [**fin_stat** *float*]
> [**pulsation_w** *float*]
> [**nb_points_par_phase** *int*]
> [**reprise** *str*]

}
where

- **dt_impr_moy_spat** *float*: Period to print the spatial average (default value is 1e6).
- **dt_impr_moy_temp** *float*: Period to print the temporal average (default value is 1e6).
- **debut_stat** *float*: Time to start the temporal averaging (default value is 1e6).
- **fin_stat** *float*: Time to end the temporal averaging (default value is 1e6).
- **pulsation_w** *float*: Pulsation for phase averaging (in case of pulsating forcing term) (no default value).
- **nb_points_par_phase** *int*: Number of samples to represent phase average all along a period (no default value).
- **reprise** *str*: val_moy_temp_xxxxxx.sauv : Keyword to resume a calculation with previous averaged quantities.
  Note that for thermal and turbulent problems, averages on temperature and turbulent viscosity are automatically calculated. To resume a calculation with phase averaging, val_moy_temp_xxxxxx.sauv_phase file is required on the directory where the job is submitted (this last file will be then automatically loaded by TRUST).

5.25.4 Ec

Description: Keyword to print total kinetic energy into the referential linked to the domain (keyword Ec). In the case where the domain is moving into a Galilean referential, the keyword Ec_dans_repere_fixe will print total kinetic energy in the Galilean referential whereas Ec will print the value calculated into the moving referential linked to the domain

See also: traitement_particulier_base (5.25.1)

Usage:
ec {

> [**Ec**]
> [**Ec_dans_repere_fixe**]
> [**periode** *float*]

}
where

- **Ec**
- **Ec_dans_repere_fixe**
- **periode** *float*: periode is the keyword to set the period of printing into the file datafile_Ec.son or datafile_Ec_dans_repere_fixe.son.

5.25.5 Thi

Description: Keyword for a THI (Homogeneous Isotropic Turbulence) calculation.

See also: traitement_particulier_base (5.25.1)

Usage:
thi {

> **init_Ec** *int*
> [**val_Ec** *float*]
> [**facon_init** *int into [0, 1]*]
> [**calc_spectre** *int into [0, 1]*]
> [**periode_calc_spectre** *float*]
> [**3D** *int into [0, 1]*]
> [**1D** *int into [0, 1]*]
> [**conservation_Ec**]
> [**longueur_boite** *float*]

}
where

- **init_Ec** *int*: Keyword to renormalize initial velocity so that kinetic energy equals to the value given by keyword val_Ec.
- **val_Ec** *float*: Keyword to impose a value for kinetic energy by velocity renormalizated if init_Ec value is 1.
- **facon_init** *int into [0, 1]*: Keyword to specify how kinetic energy is computed (0 or 1).
- **calc_spectre** *int into [0, 1]*: Calculate or not the spectrum of kinetic energy.
  Files called Sorties_THI are written with inside four columns :
  time:t global_kinetic_energy:Ec enstrophy:D skewness:S
  If calc_spectre is set to 1, a file Sorties_THI2_2 is written with three columns :

time:t kinetic_energy_at_kc=32 enstrophy_at_kc=32
If calc_spectre is set to 1, a file spectre_xxxxx is written with two columns at each time xxxxx : frequency:k energy:E(k).

- **periode_calc_spectre** *float*: Period for calculating spectrum of kinetic energy
- **3D** *int into [0, 1]*: Calculate or not the 3D spectrum
- **1D** *int into [0, 1]*: Calculate or not the 1D spectrum
- **conservation_Ec** : If set to 1, velocity field will be changed as to have a constant kinetic energy (default 0)
- **longueur_boite** *float*: Length of the calculation domain

5.25.6 Chmoy_faceperio

Description: non documente

See also: traitement_particulier_base (5.25.1)

Usage:
chmoy_faceperio bloc
where

- **bloc** *bloc_lecture* (3.17)

5.26 Navier_stokes_wc

Description: Navier-Stokes equation for a weakly-compressible fluid.

Keyword Discretize should have already been used to read the object.
See also: navier_stokes_standard (5.27)

Usage:
navier_stokes_WC *str*
Read *str* {

[**methode_calcul_pression_initiale** *str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']*]
[**projection_initiale** *int*]
[**solveur_pression** *solveur_sys_base*]
[**solveur_bar** *solveur_sys_base*]
[**dt_projection** *deuxmots*]
[**seuil_divU** *floatfloat*]
[**traitement_particulier** *traitement_particulier*]
[**correction_matrice_projection_initiale** *int*]
[**correction_calcul_pression_initiale** *int*]
[**correction_vitesse_projection_initiale** *int*]
[**correction_matrice_pression** *int*]
[**correction_vitesse_modifie** *int*]
[**gradient_pression_qdm_modifie** *int*]
[**correction_pression_modifie** *int*]
[**postraiter_gradient_pression_sans_masse**]
[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]

[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **methode_calcul_pression_initiale** *str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']* for inheritance: Keyword to select an option for the pressure calculation before the fist time step. Options are : avec_les_cl (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), avec_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicited when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (9.13) for inheritance: Linear pressure system resolution method.

- **solveur_bar** *solveur_sys_base* (9.13) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source-_Qdm_lambdaup). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt_projection** *deuxmots* (5.23) for inheritance: nb value : This keyword checks every nb time-steps the equality of velocity divergence to zero. value is the criteria convergency for the solver used.
- **seuil_divU** *floatfloat* (5.24) for inheritance: value factor : this keyword is intended to minimise the number of iterations during the pressure system resolution. The convergence criteria during this step ('seuil' in solveur_pression) is dynamically adapted according to the mass conservation. At tn , the linear system Ax=B is considered as solved if the residual ||Ax-B||<seuil(tn). For tn+1, the threshold value seuil(tn+1) will be evualated as:
  If (|max(DivU)*dt|<value)
  Seuil(tn+1)= Seuil(tn)*factor
  Else
  Seuil(tn+1)= Seuil(tn)*factor
  Endif
  The first parameter (value) is the mass evolution the user is ready to accept per timestep, and the second one (factor) is the factor of evolution for 'seuil' (for example 1.1, so 10
- **traitement_particulier** *traitement_particulier* (5.25) for inheritance: Keyword to post-process particular values.
- **correction_matrice_projection_initiale** *int* for inheritance: (IBM advanced) fix matrix of initial projection for PDF
- **correction_calcul_pression_initiale** *int* for inheritance: (IBM advanced) fix initial pressure computation for PDF
- **correction_vitesse_projection_initiale** *int* for inheritance: (IBM advanced) fix initial velocity computation for PDF
- **correction_matrice_pression** *int* for inheritance: (IBM advanced) fix pressure matrix for PDF
- **correction_vitesse_modifie** *int* for inheritance: (IBM advanced) fix velocity for PDF
- **gradient_pression_qdm_modifie** *int* for inheritance: (IBM advanced) fix pressure gradient
- **correction_pression_modifie** *int* for inheritance: (IBM advanced) fix pressure for PDF
- **postraiter_gradient_pression_sans_masse** for inheritance: (IBM advanced) avoid mass matrix multiplication for the gradient postprocessing
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.

- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format:
  n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

5.27 Navier_stokes_standard

Description: Navier-Stokes equations.

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.21) navier_stokes_QC (5.22) navier_stokes_WC (5.26)

Usage:
navier_stokes_standard *str*
Read *str* {

[**methode_calcul_pression_initiale** *str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']*]
[**projection_initiale** *int*]
[**solveur_pression** *solveur_sys_base*]
[**solveur_bar** *solveur_sys_base*]
[**dt_projection** *deuxmots*]
[**seuil_divU** *floatfloat*]
[**traitement_particulier** *traitement_particulier*]
[**correction_matrice_projection_initiale** *int*]
[**correction_calcul_pression_initiale** *int*]
[**correction_vitesse_projection_initiale** *int*]
[**correction_matrice_pression** *int*]
[**correction_vitesse_modifie** *int*]
[**gradient_pression_qdm_modifie** *int*]

[**correction_pression_modifie** *int*]
[**postraiter_gradient_pression_sans_masse**]
[**convection** *bloc_convection*]
[**diffusion** *bloc_diffusion*]
[**initial_conditions|conditions_initiales** *condinits*]
[**boundary_conditions|conditions_limites** *condlims*]
[**sources** *sources*]
[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]
[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]
[**parametre_equation** *parametre_equation_base*]
[**equation_non_resolue** *str*]

}
where

- **methode_calcul_pression_initiale** *str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']*: Keyword to select an option for the pressure calculation before the fist time step. Options are : avec_les_cl (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), avec_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicited when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection_initiale** *int*: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (9.13): Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (9.13): This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source_Qdm_lambdaup). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt_projection** *deuxmots* (5.23): nb value : This keyword checks every nb time-steps the equality of velocity divergence to zero. value is the criteria convergency for the solver used.
- **seuil_divU** *floatfloat* (5.24): value factor : this keyword is intended to minimise the number of iterations during the pressure system resolution. The convergence criteria during this step ('seuil' in solveur_pression) is dynamically adapted according to the mass conservation. At tn , the linear system Ax=B is considered as solved if the residual ||Ax-B||<seuil(tn). For tn+1, the threshold value seuil(tn+1) will be evualated as:
  If (|max(DivU)*dt|<value)
  Seuil(tn+1)= Seuil(tn)*factor
  Else
  Seuil(tn+1)= Seuil(tn)*factor
  Endif
  The first parameter (value) is the mass evolution the user is ready to accept per timestep, and the second one (factor) is the factor of evolution for 'seuil' (for example 1.1, so 10
- **traitement_particulier** *traitement_particulier* (5.25): Keyword to post-process particular values.
- **correction_matrice_projection_initiale** *int*: (IBM advanced) fix matrix of initial projection for PDF
- **correction_calcul_pression_initiale** *int*: (IBM advanced) fix initial pressure computation for PDF
- **correction_vitesse_projection_initiale** *int*: (IBM advanced) fix initial velocity computation for PDF
- **correction_matrice_pression** *int*: (IBM advanced) fix pressure matrix for PDF
- **correction_vitesse_modifie** *int*: (IBM advanced) fix velocity for PDF
- **gradient_pression_qdm_modifie** *int*: (IBM advanced) fix pressure gradient
- **correction_pression_modifie** *int*: (IBM advanced) fix pressure for PDF

- **postraiter_gradient_pression_sans_masse** : (IBM advanced) avoid mass matrix multiplication for the gradient postprocessing
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (5.5) for inheritance: Boundary conditions.
- **sources** *sources* (5.6) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file with the following format:
  n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.7) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following format: n_valeur
  x_1 y_1 [z_1] val_1
  ...
  x_n y_n [z_n] val_n
  The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (5.8) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
  Navier_Sokes_Standard
  { equation_non_resolue (t>t0)*(t<t1) }

6 /*

6.1 /*

Description: bloc of Comment in a data file.

See also: objet_u (34)

Usage:
/* comm
where

- **comm** *str*: Text to be commented.

7 champ_generique_base

Description: not_set

See also: objet_u (34) champ_post_de_champs_post (7.1) champ_post_refchamp (7.17) predefini (7.15)

Usage:

7.1 Champ_post_de_champs_post

Description: not_set

See also: champ_generique_base (7) champ_post_operateur_eqn (7.5) champ_post_transformation (7.19) champ_post_operateur_base (7.4) champ_post_statistiques_base (7.6) champ_post_extraction (7.10) champ-_post_morceau_equation (7.13) champ_post_tparoi_vef (7.18) champ_post_interpolation (7.12) champ-_post_reduction_0d (7.16)

Usage:
champ_post_de_champs_post *str*
Read *str* {

[**source** *champ_generique_base*]
[**nom_source** *str*]
[**source_reference** *str*]
[**sources_reference** *list_nom_virgule*]
[**sources** *listchamp_generique*]

}
where

- **source** *champ_generique_base* (7): the source field.
- **nom_source** *str*: To name a source field with the nom_source keyword
- **source_reference** *str*
- **sources_reference** *list_nom_virgule* (7.2)
- **sources** *listchamp_generique* (7.3): sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.2 List_nom_virgule

Description: List of name.

See also: listobj (32.5)

Usage:
{ object1 , object2 .... }
list of *nom_anonyme* (22.1) separeted with ,

7.3 Listchamp_generique

Description: XXX

See also: listobj (32.5)

Usage:
{ object1 , object2 .... }
list of *champ_generique_base* (7) separeted with ,

7.4 Champ_post_operateur_base

Description: not_set

See also: champ_post_de_champs_post (7.1) champ_post_operateur_gradient (7.11) champ_post_operateur-_divergence (7.8)

Usage:
champ_post_operateur_base *str*
Read *str* {

[**source** *champ_generique_base*]
[**nom_source** *str*]
[**source_reference** *str*]
[**sources_reference** *list_nom_virgule*]
[**sources** *listchamp_generique*]

}
where

- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.5 Champ_post_operateur_eqn

Synonymous: **operateur_eqn**

Description: not_set

See also: champ_post_de_champs_post (7.1)

Usage:
champ_post_operateur_eqn *str*
Read *str* {

[**numero_op** *int*]
[**numero_source** *int*]
[**sans_solveur_masse**]
[**compo** *int*]
[**source** *champ_generique_base*]
[**nom_source** *str*]
[**source_reference** *str*]
[**sources_reference** *list_nom_virgule*]
[**sources** *listchamp_generique*]

}
where

- **numero_op** *int*
- **numero_source** *int*
- **sans_solveur_masse**
- **compo** *int*: If you want to post-process only one component of a vector field, you can specify the number of the component after compo keyword. By default, it is set to -1 which means that all the components will be post-processed. This feature is not available in VDF disretization.
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance

- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.6 Champ_post_statistiques_base

Description: not_set

See also: champ_post_de_champs_post (7.1) correlation (7.7) moyenne (7.14) ecart_type (7.9)

Usage:
champ_post_statistiques_base *str*
Read *str* {

> **t_deb** *float*
> **t_fin** *float*
> [**source** *champ_generique_base*]
> [**nom_source** *str*]
> [**source_reference** *str*]
> [**sources_reference** *list_nom_virgule*]
> [**sources** *listchamp_generique*]

}
where

- **t_deb** *float*: Start of integration time
- **t_fin** *float*: End of integration time
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.7 Correlation

Synonymous: **champ_post_statistiques_correlation**

Description: to calculate the correlation between the two fields.

See also: champ_post_statistiques_base (7.6)

Usage:
correlation *str*
Read *str* {

> **t_deb** *float*
> **t_fin** *float*
> [**source** *champ_generique_base*]
> [**nom_source** *str*]
> [**source_reference** *str*]
> [**sources_reference** *list_nom_virgule*]
> [**sources** *listchamp_generique*]

}
where

- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.8 Champ_post_operateur_divergence

Synonymous: **divergence**

Description: To calculate divergency of a given field.

See also: champ_post_operateur_base (7.4)

Usage:
champ_post_operateur_divergence *str*
Read *str* {

[**source** *champ_generique_base*]
[**nom_source** *str*]
[**source_reference** *str*]
[**sources_reference** *list_nom_virgule*]
[**sources** *listchamp_generique*]

}
where

- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.9 Ecart_type

Synonymous: **champ_post_statistiques_ecart_type**

Description: to calculate the standard deviation (statistic rms) of the field nom_champ.

See also: champ_post_statistiques_base (7.6)

Usage:
ecart_type *str*
Read *str* {

t_deb *float*
t_fin *float*

[**source** *champ_generique_base*]
[**nom_source** *str*]
[**source_reference** *str*]
[**sources_reference** *list_nom_virgule*]
[**sources** *listchamp_generique*]

}
where

- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.10 Champ_post_extraction

Synonymous: **extraction**

Description: To create a surface field (values at the boundary) of a volume field

See also: champ_post_de_champs_post (7.1)

Usage:
champ_post_extraction *str*
Read *str* {

domaine *str*
nom_frontiere *str*
[**methode** *str into ['trace', 'champ_frontiere']*]
[**source** *champ_generique_base*]
[**nom_source** *str*]
[**source_reference** *str*]
[**sources_reference** *list_nom_virgule*]
[**sources** *listchamp_generique*]

}
where

- **domaine** *str*: name of the volume field
- **nom_frontiere** *str*: boundary name where the values of the volume field will be picked
- **methode** *str into ['trace', 'champ_frontiere']*: name of the extraction method (trace by_default or champ_frontiere)
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.11 Champ_post_operateur_gradient

Synonymous: **gradient**

Description: To calculate gradient of a given field.

See also: champ_post_operateur_base (7.4)

Usage:
champ_post_operateur_gradient *str*
Read *str* {

[**source** *champ_generique_base*]
[**nom_source** *str*]
[**source_reference** *str*]
[**sources_reference** *list_nom_virgule*]
[**sources** *listchamp_generique*]

}
where

- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.12 Champ_post_interpolation

Synonymous: **interpolation**

Description: To create a field which is an interpolation of the field given by the keyword source.

See also: champ_post_de_champs_post (7.1)

Usage:
champ_post_interpolation *str*
Read *str* {

localisation *str*
[**methode** *str*]
[**domaine** *str*]
[**optimisation_sous_maillage** *str into ['default', 'yes', 'no']*]
[**source** *champ_generique_base*]
[**nom_source** *str*]
[**source_reference** *str*]
[**sources_reference** *list_nom_virgule*]
[**sources** *listchamp_generique*]

}
where

- **localisation** *str*: type_loc indicate where is done the interpolation (elem for element or som for node).

- **methode** *str*: The optional keyword methode is limited to calculer_champ_post for the moment.
- **domaine** *str*: the domain name where the interpolation is done (by default, the calculation domain)
- **optimisation_sous_maillage** *str into ['default', 'yes', 'no']*
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.13 Champ_post_morceau_equation

Synonymous: **morceau_equation**

Description: To calculate a field related to a piece of equation. For the moment, the field which can be calculated is the stability time step of an operator equation. The problem name and the unknown of the equation should be given by Source refChamp { Pb_Champ problem_name unknown_field_of_equation }

See also: champ_post_de_champs_post (7.1)

Usage:
champ_post_morceau_equation *str*
Read *str* {

> **type** *str*
> **numero** *int*
> **option** *str into ['stabilite', 'flux_bords', 'flux_surfacique_bords']*
> [**compo** *int*]
> [**source** *champ_generique_base*]
> [**nom_source** *str*]
> [**source_reference** *str*]
> [**sources_reference** *list_nom_virgule*]
> [**sources** *listchamp_generique*]

}
where

- **type** *str*: can only be operateur for equation operators.
- **numero** *int*: numero will be 0 (diffusive operator) or 1 (convective operator).
- **option** *str into ['stabilite', 'flux_bords', 'flux_surfacique_bords']*: option is stability for time steps or flux_bords for boundary fluxes or flux_surfacique_bords for boundary surfacic fluxes
- **compo** *int*: compo will specify the number component of the boundary flux (for boundary fluxes, in this case compo permits to specify the number component of the boundary flux choosen).
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.14 Moyenne

Synonymous: **champ_post_statistiques_moyenne**

Description: to calculate the average of the field over time

See also: champ_post_statistiques_base (7.6)

Usage:
moyenne *str*
Read *str* {

[**moyenne_convergee** *champ_base*]
t_deb *float*
t_fin *float*
[**source** *champ_generique_base*]
[**nom_source** *str*]
[**source_reference** *str*]
[**sources_reference** *list_nom_virgule*]
[**sources** *listchamp_generique*]

}
where

- **moyenne_convergee** *champ_base* (14.1): This option allows to read a converged time averaged field in a .xyz file in order to calculate, when resuming the calculation, the statistics fields (rms, correlation) which depend on this average. In that case, the time averaged field is not updated during the resume of calculation. In this case, the time averaged field must be fully converged to avoid errors when calculating high order statistics.
- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.15 Predefini

Description: This keyword is used to post process predefined postprocessing fields.

See also: champ_generique_base (7)

Usage:
predefini *str*
Read *str* {

pb_champ *deuxmots*

}
where

- **pb_champ** *deuxmots* (5.23): { Pb_champ nom_pb nom_champ } : nom_pb is the problem name and nom_champ is the selected field name. The available keywords for the field name are: energie_cinetique_totale, energie_cinetique_elem, viscosite_turbulente, viscous_force_x, viscous_force_y, viscous_force_z, pressure_force_x, pressure_force_y, pressure_force_z, total_force_x, total_force_y, total_force_z, viscous_force, pressure_force, total_force

7.16 Champ_post_reduction_0d

Synonymous: **reduction_0d**

Description: To calculate the min, max, sum, average, weighted sum, weighted average, weighted sum by porosity, weighted average by porosity, euclidian norm, normalized euclidian norm, L1 norm, L2 norm of a field.

See also: champ_post_de_champs_post (7.1)

Usage:
champ_post_reduction_0d *str*
Read *str* {

> **methode** *str into ['min', 'max', 'moyenne', 'average', 'moyenne_ponderee', 'weighted_average', 'somme', 'sum', 'somme_ponderee', 'weighted_sum', 'somme_ponderee_porosite', 'weighted_sum-_porosity', 'euclidian_norm', 'normalized_euclidian_norm', 'L1_norm', 'L2_norm', 'valeur_a_gauche', 'left_value']*
> [**source** *champ_generique_base*]
> [**nom_source** *str*]
> [**source_reference** *str*]
> [**sources_reference** *list_nom_virgule*]
> [**sources** *listchamp_generique*]

}
where

- **methode** *str into ['min', 'max', 'moyenne', 'average', 'moyenne_ponderee', 'weighted_average', 'somme', 'sum', 'somme_ponderee', 'weighted_sum', 'somme_ponderee_porosite', 'weighted_sum-_porosity', 'euclidian_norm', 'normalized_euclidian_norm', 'L1_norm', 'L2_norm', 'valeur_a_gauche', 'left_value']*: name of the reduction method:
  - min for the minimum value,
  - max for the maximum value,
  - average (or moyenne) for a mean,
  - weighted_average (or moyenne_ponderee) for a mean ponderated by integration volumes, e.g: cell volumes for temperature and pressure in VDF, volumes around faces for velocity and temperature in VEF,
  - sum (or somme) for the sum of all the values of the field,
  - weighted_sum (or somme_ponderee) for a weighted sum (integral),
  - weighted_average_porosity (or moyenne_ponderee_porosite) and weighted_sum_porosity (or somme-_ponderee_porosite) for the mean and sum weighted by the volumes of the elements, only for ELEM localisation,
  - euclidian_norm for the euclidian norm,
  - normalized_euclidian_norm for the euclidian norm normalized,
  - L1_norm for norm L1,
  - L2_norm for norm L2
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.17 Champ_post_refchamp

Synonymous: **refchamp**

Description: Field of prolem

See also: champ_generique_base (7)

Usage:
champ_post_refchamp *str*
Read *str* {

> **pb_champ** *deuxmots*
> [**nom_source** *str*]

}
where

- **pb_champ** *deuxmots* (5.23): { Pb_champ nom_pb nom_champ } : nom_pb is the problem name and nom_champ is the selected field name.
- **nom_source** *str*: The alias name for the field

7.18 Champ_post_tparoi_vef

Synonymous: **tparoi_vef**

Description: This keyword is used to post process (only for VEF discretization) the temperature field with a slight difference on boundaries with Neumann condition where law of the wall is applied on the temperature field. nom_pb is the problem name and field_name is the selected field name. A keyword (temperature_physique) is available to post process this field without using Definition_champs.

See also: champ_post_de_champs_post (7.1)

Usage:
champ_post_tparoi_vef *str*
Read *str* {

> [**source** *champ_generique_base*]
> [**nom_source** *str*]
> [**source_reference** *str*]
> [**sources_reference** *list_nom_virgule*]
> [**sources** *listchamp_generique*]

}
where

- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

7.19 Champ_post_transformation

Synonymous: **transformation**

Description: To create a field with a transformation.

See also: champ_post_de_champs_post (7.1)

Usage:
champ_post_transformation *str*
Read *str* {

> **methode** *str into ['produit_scalaire', 'norme', 'vecteur', 'formule', 'composante']*
> [**expression** *n word1 word2 ... wordn*]
> [**numero** *int*]
> [**localisation** *str*]
> [**source** *champ_generique_base*]
> [**nom_source** *str*]
> [**source_reference** *str*]
> [**sources_reference** *list_nom_virgule*]
> [**sources** *listchamp_generique*]

}
where

- **methode** *str into ['produit_scalaire', 'norme', 'vecteur', 'formule', 'composante']*: methode norme : will calculate the norm of a vector given by a source field
  methode produit_scalaire : will calculate the dot product of two vectors given by two sources fields
  methode composante numero integer : will create a field by extracting the integer component of a field given by a source field
  methode formule expression 1 : will create a scalar field located to elements using expressions with x,y,z,t parameters and field names given by a source field or several sources fields.
  methode vecteur expression N f1(x,y,z,t) fN(x,y,z,t) : will create a vector field located to elements by defining its N components with N expressions with x,y,z,t parameters and field names given by a source field or several sources fields.
- **expression** *n word1 word2 ... wordn*: see methodes formule and vecteur
- **numero** *int*: see methode composante
- **localisation** *str*: type_loc indicate where is done the interpolation (elem for element or som for node). The optional keyword methode is limited to calculer_champ_post for the moment
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.2) for inheritance
- **sources** *listchamp_generique* (7.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

8 chimie

Description: Keyword to describe the chmical reactions

See also: objet_u (34)

Usage:
chimie *str*
Read *str* {

reactions *reactions*
[**modele_micro_melange** *int*]
[**constante_modele_micro_melange** *float*]
[**espece_en_competition_micro_melange** *str*]

}
where

- **reactions** *reactions* (8.1): list of reactions
- **modele_micro_melange** *int*: modele_micro_melange (0 by default)
- **constante_modele_micro_melange** *float*: constante of modele (1 by default)
- **espece_en_competition_micro_melange** *str*: espece in competition in reactions

8.1 Reactions

Description: list of reactions

See also: listobj (32.5)

Usage:
{ object1 , object2 .... }
list of *reaction* (8.1.1) separeted with ,

8.1.1 Reaction

Description: Keyword to describe reaction:
w =K pow(T,beta) exp(-Ea/(R T)) Π pow(Reactif_i,activitivity_i).
If K_inv >0,
w= K pow(T,beta) exp(-Ea/(R T)) (Π pow(Reactif_i,activitivity_i) - Kinv/exp(-c_r_Ea/(R T)) Π pow(Produit-_i,activitivity_i))

See also: objet_lecture (33)

Usage:
{

reactifs *str*
produits *str*
[**constante_taux_reaction** *float*]
[**coefficients_activites** *bloc_lecture*]
enthalpie_reaction *float*
energie_activation *float*
exposant_beta *float*
[**contre_reaction** *float*]
[**contre_energie_activation** *float*]

}
where

- **reactifs** *str*: LHS of equation (ex CH4+2*O2)
- **produits** *str*: RHS of equation (ex CO2+2*H20)
- **constante_taux_reaction** *float*: constante of cinetic K
- **coefficients_activites** *bloc_lecture* (3.17): coefficients od ativity (exemple { CH4 1 O2 2 })
- **enthalpie_reaction** *float*: DH
- **energie_activation** *float*: Ea
- **exposant_beta** *float*: Beta

- **contre_reaction** *float*: K_inv
- **contre_energie_activation** *float*: c_r_Ea

9 class_generic

Description: not_set

See also: objet_u (34) dt_start (9.6) solveur_sys_base (9.13)

Usage:

9.1 Amgx

Description: Solver via AmgX API

See also: petsc (9.11)

Usage:
amgx **solveur** **option_solveur** [**atol**] [**rtol**]
where

- **solveur** *str*
- **option_solveur** *bloc_lecture* (3.17)
- **atol** *float*: Absolute threshold for convergence (same as seuil option)
- **rtol** *float*: Relative threshold for convergence

9.2 Cholesky

Description: Cholesky direct method.

See also: solveur_sys_base (9.13)

Usage:
cholesky *str*
Read *str* {

[**impr**]
[**quiet**]

}
where

- **impr** : Keyword which may be used to print the resolution time.
- **quiet** : To disable printing of information

9.3 Dt_calc

Description: The time step at first iteration is calculated in agreement with CFL condition.

See also: dt_start (9.6)

Usage:
dt_calc

9.4 Dt_fixe

Description: The first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).

See also: dt_start (9.6)

Usage:
dt_fixe value
where

- **value** *float*: first time step.

9.5 Dt_min

Description: The first iteration is based on dt_min.

See also: dt_start (9.6)

Usage:
dt_min

9.6 Dt_start

Description: not_set

See also: class_generic (9) dt_calc (9.3) dt_min (9.5) dt_fixe (9.4)

Usage:
dt_start

9.7 Gcp_ns

Description: not_set

See also: gcp (9.12)

Usage:
gcp_ns *str*
Read *str* {

> **solveur0** *solveur_sys_base*
> **solveur1** *solveur_sys_base*
> [**precond** *precond_base*]
> [**precond_nul**]
> **seuil** *float*
> [**impr**]
> [**quiet**]
> [**save_matrix|save_matrice**]
> [**optimized**]
> [**nb_it_max** *int*]

}
where

- **solveur0** *solveur_sys_base* (9.13): Solver type.
- **solveur1** *solveur_sys_base* (9.13): Solver type.
- **precond** *precond_base* (24) for inheritance: Keyword to define system preconditioning in order to accelerate resolution by the conjugated gradient. Many parallel preconditioning methods are not equivalent to their sequential counterpart, and you should therefore expect differences, especially when you select a high value of the final residue (seuil). The result depends on the number of processors and on the mesh splitting. It is sometimes useful to run the solver with no preconditioning at all. In particular:
  - when the solver does not converge during initial projection,
  - when comparing sequential and parallel computations.
  With no preconditioning, except in some particular cases (no open boundary), the sequential and the parallel computations should provide exactly the same results within fpu accuracy. If not, there might be a coding error or the system of equations is singular.
- **precond_nul** for inheritance: Keyword to not use a preconditioning method.
- **seuil** *float* for inheritance: Value of the final residue. The gradient ceases iteration when the Euclidean residue standard ||Ax-B|| is less than this value.
- **impr** for inheritance: Keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **quiet** for inheritance: To not displaying any outputs of the solver.
- **save_matrix|save_matrice** for inheritance: to save the matrix in a file.
- **optimized** for inheritance: This keyword triggers a memory and network optimized algorithms useful for strong scaling (when computing less than 100 000 elements per processor). The matrix and the vectors are duplicated, common items removed and only virtual items really used in the matrix are exchanged.
  Warning: this is experimental and known to fail in some VEF computations (L2 projection step will not converge). Works well in VDF.
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gcp.

9.8 Gen

Description: not_set

See also: solveur_sys_base (9.13)

Usage:
gen *str*
Read *str* {

> **solv_elem** *str*
> **precond** *precond_base*
> [**seuil** *float*]
> [**impr**]
> [**save_matrix|save_matrice**]
> [**quiet**]
> [**nb_it_max** *int*]
> [**force**]

}
where

- **solv_elem** *str*: To specify a solver among gmres or bicgstab.
- **precond** *precond_base* (24): The only preconditionner that we can specify is ilu.
- **seuil** *float*: Value of the final residue. The solver ceases iterations when the Euclidean residue standard ||Ax-B|| is less than this value. default value 1e-12.

- **impr** : Keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **save_matrix|save_matrice** : To save the matrix in a file.
- **quiet** : To not displaying any outputs of the solver.
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the GEN solver.
- **force** : Keyword to set ipar[5]=-1 in the GEN solver. This is helpful if you notice that the solver does not perform more than 100 iterations. If this keyword is specified in the datafile, you should provide nb_it_max.

9.9 Gmres

Description: Gmres method (for non symetric matrix).

See also: solveur_sys_base (9.13)

Usage:
gmres *str*
Read *str* {

[**impr**]
[**quiet**]
[**seuil** *float*]
[**diag**]
[**nb_it_max** *int*]
[**controle_residu** *int into [0, 1]*]
[**save_matrix|save_matrice**]
[**dim_espace_krilov** *int*]

}
where

- **impr** : Keyword which may be used to print the convergence.
- **quiet** : To disable printing of information
- **seuil** *float*: Convergence value.
- **diag** : Keyword to use diagonal preconditionner (in place of pilut that is not parallel).
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** *int into [0, 1]*: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.
- **save_matrix|save_matrice** : to save the matrix in a file.
- **dim_espace_krilov** *int*

9.10 Optimal

Description: Optimal is a solver which tests several solvers of the previous list to choose the fastest one for the considered linear system.

See also: solveur_sys_base (9.13)

Usage:
optimal *str*
Read *str* {

seuil *float*
[**impr**]

[**quiet**]
[**save_matrix|save_matrice**]
[**frequence_recalc** *int*]
[**nom_fichier_solveur** *str*]
[**fichier_solveur_non_recree**]

}
where

- **seuil** *float*: Convergence threshold
- **impr** : To print the convergency of the fastest solver
- **quiet** : To disable printing of information
- **save_matrix|save_matrice** : To save the linear system (A, x, B) into a file
- **frequence_recalc** *int*: To set a time step period (by default, 100) for re-checking the fatest solver
- **nom_fichier_solveur** *str*: To specify the file containing the list of the tested solvers
- **fichier_solveur_non_recree** : To avoid the creation of the file containing the list

9.11 Petsc

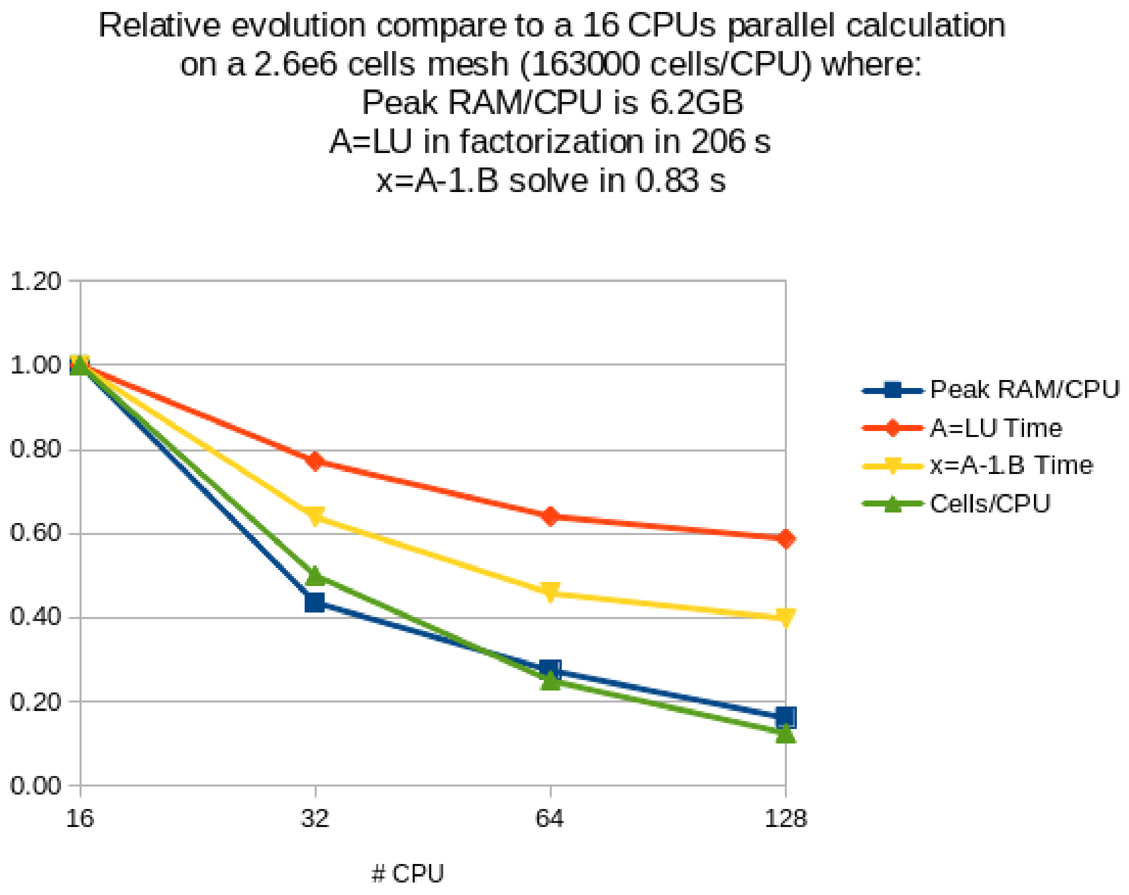
Description: Solver via Petsc API

Usage:

Solveur_pression Petsc *Solver* { **precond** *Precond*
[**seuil** *seuil* | **nb_it_max** *integer*]
[**impr** | **quiet**]
[**save_matrix** | **read_matrix**]
}

Solver : Several solvers through PETSc API are available :
GCP : Conjugate Gradient
PIPECG : Pipelined Conjugate Gradient (possible reduced CPU cost during massive parallel calculation due to a single non-blocking reduction per iteration, if TRUST is built with a MPI-3 implementation).
GMRES : Generalized Minimal Residual
BICGSTAB : Stabilized Bi-Conjugate Gradient
IBICGSTAB : Improved version of previous one for massive parallel computations (only a single global reduction operation instead of the usual 3 or 4).
CHOLESKY : Parallelized version of Cholesky from MUMPS library. This solver accepts since the 1.6.7 version an option to select a different ordering than the automatic selected one by MUMPS (and printed by using the **impr** option). The possible choices are **Metis | Scotch | PT-Scotch | Parmetis**. The two last options can only be used during a parallel calculation, whereas the two first are available for sequential or parallel calculations. It seems that the CPU cost of A=LU factorization but also of the backward/forward elimination steps may sometimes be reduced by selecting a different ordering (Scotch seems often the best for b/f elimination) than the default one. Notice that this solver requires a huge amont of memory compared to iterative methods. To know how many RAM you will need by core, then use the **impr** option to have detailled informations during the analysis phase and before the factorisation phase (in the following output, you will learn that the largest memory is taken by the 0th CPU with 108MB):
...
** Rank of proc needing largest memory in IC facto : 0
**** Estimated corresponding MBYTES for IC facto : 108**
...

Thanks to the following graph, you read that in order to solve for instance a flow on a mesh with 2.6e6 cells, you will need to run a parallel calculation on 32 CPUs if you have cluster nodes with only 4GB/core (6.2GB*0.42˜2.6GB) :



CHOLESKY_OUT_OF_CORE : Same as the previous one but with a written LU decomposition of disk (save RAM memory but add an extra CPU cost during Ax=B solve)
CHOLESKY_SUPERLU : Parallelized Cholesky from SUPERLU_DIST library (less CPU and RAM efficient than the previous one)
CHOLESKY_PASTIX : Parallelized Cholesky from PASTIX library
CHOLESKY_UMFPACK : Sequential Cholesky from UMFPACK library (seems fast).
CLI { string } : Command Line Interface. Should be used only by advanced users, to access the whole solver/preconditioners from the PETSC API. To find all the available options, run your calculation with the -ksp_view -help options:

```
trust datafile [N]  –ksp_view –help
. . .
Preconditioner (PC) Options -------------------------------------------------
  -pc_type Preconditioner:(one of) none jacobi pbjacobi bjacobi sor lu shell mg
      eisenstat ilu icc cholesky asm ksp composite redundant nn mat fieldsplit galerkin openmp spai hypre
tfs (PCSetType)
  HYPRE preconditioner options
  -pc_hypre_type <pilut> (choose one of) pilut parasails boomeramg
  HYPRE ParaSails Options
```

-pc_hypre_parasails_nlevels <1>: Number of number of levels (None)
-pc_hypre_parasails_thresh <0.1>: Threshold (None)
-pc_hypre_parasails_filter <0.1>: filter (None)
-pc_hypre_parasails_loadbal <0>: Load balance (None)
-pc_hypre_parasails_logging: <FALSE> Print info to screen (None)
-pc_hypre_parasails_reuse: <FALSE> Reuse nonzero pattern in preconditioner (None)
-pc_hypre_parasails_sym <nonsymmetric> (choose one of) nonsymmetric SPD nonsymmetric,SPD
Krylov Method (KSP) Options ---
-ksp_type Krylov method:(one of) cg cgne stcg gltr richardson chebychev gmres tcqmr bcgs bcgsl cgs tfqmr cr lsqr preonly qcg bicg fgmres minres symmlq lgmres lcd (KSPSetType)
-ksp_max_it <10000>: Maximum number of iterations (KSPSetTolerances)
-ksp_rtol <0>: Relative decrease in residual norm (KSPSetTolerances)
-ksp_atol <1e-12>: Absolute value of residual norm (KSPSetTolerances)
-ksp_divtol <10000>: Residual norm increase cause divergence (KSPSetTolerances)
-ksp_converged_use_initial_residual_norm: Use initial residual residual norm for computing relative convergence
-ksp_monitor_singular_value <stdout>: Monitor singular values (KSPMonitorSet)
-ksp_monitor_short <stdout>: Monitor preconditioned residual norm with fewer digits (KSPMonitorSet)
-ksp_monitor_draw: Monitor graphically preconditioned residual norm (KSPMonitorSet)
-ksp_monitor_draw_true_residual: Monitor graphically true residual norm (KSPMonitorSet)

Example to use the multigrid method as a solver, not only as a preconditioner:
Solveur_pression Petsc CLI { -ksp_type richardson -pc_type hypre -pc_hypre_type boomeramg -ksp_atol 1.e-7 }

Precond : Several preconditioners are available :
NULL { } : No preconditioner used
BLOCK_JACOBI_ICC { **level** k **ordering natural** | **rcm** } : Incomplete Cholesky factorization for symmetric matrix with the PETSc implementation. The integer k is the factorization level (default value, 1). In parallel, the factorization is done by block (one per processor by default). The ordering of the local matrix is **natural** by default, but **rcm** ordering, which reduces the bandwith of the local matrix, may interestingly improves the quality of the decomposition and reduces the number of iterations.
SSOR { **omega** double } : Symmetric Successive Over Relaxation algorithm. **omega** (default value, 1.5) defines the relaxation factor.
EISENTAT { **omega** double } : SSOR version with Eisenstat trick which reduces the number of computations and thus CPU cost
SPAI { **level** nlevels **epsilon** thresh } : Spai Approximate Inverse algorithm from Parasails Hypre library. Two parameters are available, nlevels and thresh.
PILUT { **level** k **epsilon** thresh }: Dual Threashold Incomplete LU factorization. The integer k is the factorization level and **epsilon** is the drop tolerance.
DIAG { } : Diagonal (Jacobi) preconditioner.
BOOMERAMG { } : Multigrid preconditioner (no option is available yet, look at CLI command and Petsc documentation to try other options).

seuil corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard ||Ax-B|| is less than the value *seuil*.

nb_it_max integer : In order to specify a given number of iterations instead of a condition on the residue with the keyword **seuil**. May be useful when defining a PETSc solver for the implicit time scheme where convergence is very fast: 5 or less iterations seems enough.

impr is the keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).

quiet is a keyword which is used to not displaying any outputs of the solver.

save_matrix|read_matrix are the keywords to save|read into a file the constant matrix A of the linear system Ax=B solved (eg: matrix from the pressure linear system for an incompressible flow). It is useful when you want to minimize the MPI communications on massive parallel calculation. Indeed, in VEF discretization, the overlapping width (generaly 2, specified with the **largeur_joint** option in the partition keyword **partition**) can be reduced to 1, once the matrix has been properly assembled and saved. The cost of the MPI communications in TRUST itself (not in PETSc) will be reduced with length messages divided by 2. So the strategy is:
I) Partition your VEF mesh with a **largeur_joint** value of 2
II) Run your parallel calculation on 0 time step, to build and save the matrix with the **save_matrix** option. A file named *Matrix_NBROWS_rows_NCPUS_cpus.petsc* will be saved to the disk (where NBROWS is the number of rows of the matrix and NCPUS the number of CPUs used).
III) Partition your VEF mesh with a **largeur_joint** value of 1
IV) Run your parallel calculation completly now and substitute the **save_matrix** option by the **read_matrix** option. Some interesting gains have been noticed when the cost of linear system solve with PETSc is small compared to all the other operations.

TIPS:
A) Solver for symmetric linear systems (e.g: Pressure system from Navier-Stokes equations):
-The **CHOLESKY** parallel solver is from MUMPS library. It offers better performance than all others solvers if you have enough RAM for your calculation. A parallel calculation on a cluster with 4GBytes on each processor, 40000 cells/processor seems the upper limit. Seems to be very slow to initialize above 500 cpus/cores.

-When running a parallel calculation with a high number of cpus/cores (typically more than 500) where preconditioner scalabilty is the key for CPU performance, consider **BICGSTAB** with **BLOCK_JACOBI_ICC(1)** as preconditioner or if not converges, **GCP** with **BLOCK_JACOBI_ICC(1)** as preconditioner.

-For other situations, the first choice should be **GCP/SSOR**. In order to fine tune the solver choice, each one of the previous list should be considered. Indeed, the CPU speed of a solver depends of a lot of parameters. You may give a try to the **OPTIMAL** solver to help you to find the fastest solver on your study.

B) Solver for non symmetric linear systems (e.g.: Implicit schemes):
The **BICGSTAB/DIAG** solver seems to offer the best performances.

Additional information is available into the PETSC documentation available on:
$TRUST_ROOT/lib/src/LIBPETSC/petsc/*/docs/manual.pdf

See also: solveur_sys_base (9.13) amgx (9.1)

Usage:
petsc solveur option_solveur [**atol**] [**rtol**]
where

- **solveur** *str*
- **option_solveur** *bloc_lecture* (3.17)
- **atol** *float*: Absolute threshold for convergence (same as seuil option)
- **rtol** *float*: Relative threshold for convergence

9.12 Gcp

Description: Preconditioned conjugated gradient.

See also: solveur_sys_base (9.13) gcp_ns (9.7)

Usage:
gcp *str*
Read *str* {

> [**precond** *precond_base*]
> [**precond_nul**]
> **seuil** *float*
> [**impr**]
> [**quiet**]
> [**save_matrix|save_matrice**]
> [**optimized**]
> [**nb_it_max** *int*]

}
where

- **precond** *precond_base* (24): Keyword to define system preconditioning in order to accelerate resolution by the conjugated gradient. Many parallel preconditioning methods are not equivalent to their sequential counterpart, and you should therefore expect differences, especially when you select a high value of the final residue (seuil). The result depends on the number of processors and on the mesh splitting. It is sometimes useful to run the solver with no preconditioning at all. In particular:
  - when the solver does not converge during initial projection,
  - when comparing sequential and parallel computations.
  With no preconditioning, except in some particular cases (no open boundary), the sequential and the parallel computations should provide exactly the same results within fpu accuracy. If not, there might be a coding error or the system of equations is singular.
- **precond_nul** : Keyword to not use a preconditioning method.
- **seuil** *float*: Value of the final residue. The gradient ceases iteration when the Euclidean residue standard ||Ax-B|| is less than this value.
- **impr** : Keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **quiet** : To not displaying any outputs of the solver.
- **save_matrix|save_matrice** : to save the matrix in a file.
- **optimized** : This keyword triggers a memory and network optimized algorithms useful for strong scaling (when computing less than 100 000 elements per processor). The matrix and the vectors are duplicated, common items removed and only virtual items really used in the matrix are exchanged. Warning: this is experimental and known to fail in some VEF computations (L2 projection step will not converge). Works well in VDF.
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the Gcp.

9.13 Solveur_sys_base

Description: Basic class to solve the linear system.

See also: class_generic (9) optimal (9.10) gen (9.8) petsc (9.11) gcp (9.12) cholesky (9.2) gmres (9.9)

Usage:

10

10.1

Description: Comments in a data file.

See also: objet_u (34)

Usage:
# comm
where

- **comm** *str*: Text to be commented.

11 condlim_base

Description: Basic class of boundary conditions.

See also: objet_u (34) paroi_fixe (11.31) symetrie (11.39) periodique (11.36) paroi_adiabatique (11.21) dirichlet (11.4) neumann (11.20) paroi_contact (11.22) paroi_contact_fictif (11.23) paroi_echange_contact-_vdf (11.27) paroi_echange_externe_impose (11.28) paroi_echange_global_impose (11.30) Paroi (11.3) paroi_flux_impose (11.33) frontiere_ouverte_fraction_massique_imposee (11.8) paroi_echange_contact-_correlation_vdf (11.25) paroi_echange_contact_correlation_vef (11.26) Neumann_homogene (11.1)

Usage:
condlim_base

11.1 Neumann_homogene

Description: Homogeneous neumann boundary condition

See also: condlim_base (11) Neumann_paroi_adiabatique (11.2)

Usage:
Neumann_homogene

11.2 Neumann_paroi_adiabatique

Description: Adiabatic wall neumann boundary condition

See also: Neumann_homogene (11.1)

Usage:
Neumann_paroi_adiabatique

11.3 Paroi

Description: Impermeability condition at a wall called bord (edge) (standard flux zero). This condition must be associated with a wall type hydraulic condition.

See also: condlim_base (11)

Usage:
Paroi

11.4 Dirichlet

Description: Dirichlet condition at the boundary called bord (edge) : 1). For Navier-Stokes equations, velocity imposed at the boundary; 2). For scalar transport equation, scalar imposed at the boundary.

See also: condlim_base (11) paroi_defilante (11.24) paroi_knudsen_non_negligeable (11.34) frontiere_ouverte_vitesse_imposee (11.18) frontiere_ouverte_temperature_imposee (11.17) frontiere_ouverte_concentration_imposee (11.7) paroi_temperature_imposee (11.35) scalaire_impose_paroi (11.37)

Usage:
dirichlet

11.5 Entree_temperature_imposee_h

Description: Particular case of class frontiere_ouverte_temperature_imposee for enthalpy equation.

See also: frontiere_ouverte_temperature_imposee (11.17)

Usage:
entree_temperature_imposee_h ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.6 Frontiere_ouverte

Description: Boundary outlet condition on the boundary called bord (edge) (diffusion flux zero). This condition must be associated with a boundary outlet hydraulic condition.

See also: neumann (11.20)

Usage:
frontiere_ouverte var_name ch
where

- **var_name** *str into ['T_ext', 'C_ext', 'Y_ext', 'K_Eps_ext', 'Fluctu_Temperature_ext', 'Flux_Chaleur_Turb_ext', 'V2_ext', 'a_ext']*: Field name.
- **ch** *champ_front_base* (15.1): Boundary field type.

11.7 Frontiere_ouverte_concentration_imposee

Description: Imposed concentration condition at an open boundary called bord (edge) (situation corresponding to a fluid inlet). This condition must be associated with an imposed inlet velocity condition.

See also: dirichlet (11.4)

Usage:
frontiere_ouverte_concentration_imposee ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.8 Frontiere_ouverte_fraction_massique_imposee

Description: not_set

See also: condlim_base (11)

Usage:
frontiere_ouverte_fraction_massique_imposee ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.9 Frontiere_ouverte_gradient_pression_impose

Description: Normal imposed pressure gradient condition on the open boundary called bord (edge). This boundary condition may be only used in VDF discretization. The imposed $\partial P/\partial n$ value is expressed in Pa.m-1.

See also: neumann (11.20) frontiere_ouverte_gradient_pression_impose_vefprep1b (11.10)

Usage:
frontiere_ouverte_gradient_pression_impose ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.10 Frontiere_ouverte_gradient_pression_impose_vefprep1b

Description: Keyword for an outlet boundary condition in VEF P1B/P1NC on the gradient of the pressure.

See also: frontiere_ouverte_gradient_pression_impose (11.9)

Usage:
frontiere_ouverte_gradient_pression_impose_vefprep1b ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.11 Frontiere_ouverte_gradient_pression_libre_vef

Description: Class for outlet boundary condition in VEF like Orlansky. There is no reference for pressure for theses boundary conditions so it is better to add pressure condition (with Frontiere_ouverte_pression-_imposee) on one or two cells (for symmetry in a channel) of the boundary where Orlansky conditions are imposed.

See also: neumann (11.20)

Usage:
frontiere_ouverte_gradient_pression_libre_vef

11.12 Frontiere_ouverte_gradient_pression_libre_vefprep1b

Description: Class for outlet boundary condition in VEF P1B/P1NC like Orlansky.

See also: neumann (11.20)

Usage:
frontiere_ouverte_gradient_pression_libre_vefprep1b

11.13 Frontiere_ouverte_pression_imposee

Description: Imposed pressure condition at the open boundary called bord (edge). The imposed pressure field is expressed in Pa.

See also: neumann (11.20)

Usage:
frontiere_ouverte_pression_imposee ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.14 Frontiere_ouverte_pression_imposee_orlansky

Description: This boundary condition may only be used with VDF discretization. There is no reference for pressure for this boundary condition so it is better to add pressure condition (with Frontiere_ouverte-_pression_imposee) on one or two cells (for symetry in a channel) of the boundary where Orlansky conditions are imposed.

See also: neumann (11.20)

Usage:
frontiere_ouverte_pression_imposee_orlansky

11.15 Frontiere_ouverte_pression_moyenne_imposee

Description: Class for open boundary with pressure mean level imposed.

See also: neumann (11.20)

Usage:
frontiere_ouverte_pression_moyenne_imposee pext
where

- **pext** *float*: Mean pressure.

11.16 Frontiere_ouverte_rho_u_impose

Description: This keyword is used to designate a condition of imposed mass rate at an open boundary called bord (edge). The imposed mass rate field at the inlet is vectorial and the imposed velocity values are expressed in kg.s-1. This boundary condition can be used only with the Quasi compressible model.

See also: frontiere_ouverte_vitesse_imposee_sortie (11.19)

Usage:
frontiere_ouverte_rho_u_impose ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.17 Frontiere_ouverte_temperature_imposee

Description: Imposed temperature condition at the open boundary called bord (edge) (in the case of fluid inlet). This condition must be associated with an imposed inlet velocity condition. The imposed temperature value is expressed in oC or K.

See also: dirichlet (11.4) entree_temperature_imposee_h (11.5)

Usage:
frontiere_ouverte_temperature_imposee ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.18 Frontiere_ouverte_vitesse_imposee

Description: Class for velocity-inlet boundary condition. The imposed velocity field at the inlet is vectorial and the imposed velocity values are expressed in m.s-1.

See also: dirichlet (11.4) frontiere_ouverte_vitesse_imposee_sortie (11.19)

Usage:
frontiere_ouverte_vitesse_imposee ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.19 Frontiere_ouverte_vitesse_imposee_sortie

Description: Sub-class for velocity boundary condition. The imposed velocity field at the open boundary is vectorial and the imposed velocity values are expressed in m.s-1.

See also: frontiere_ouverte_vitesse_imposee (11.18) frontiere_ouverte_rho_u_impose (11.16)

Usage:
frontiere_ouverte_vitesse_imposee_sortie ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.20 Neumann

Description: Neumann condition at the boundary called bord (edge) : 1). For Navier-Stokes equations, constraint imposed at the boundary; 2). For scalar transport equation, flux imposed at the boundary.

See also: condlim_base (11) frontiere_ouverte_gradient_pression_libre_vef (11.11) frontiere_ouverte_gradient-_pression_libre_vefprep1b (11.12) frontiere_ouverte_gradient_pression_impose (11.9) frontiere_ouverte-_pression_imposee (11.13) frontiere_ouverte_pression_imposee_orlansky (11.14) frontiere_ouverte_pression-_moyenne_imposee (11.15) frontiere_ouverte (11.6) sortie_libre_temperature_imposee_h (11.38)

Usage:
neumann

11.21 Paroi_adiabatique

Description: Normal zero flux condition at the wall called bord (edge).

See also: condlim_base (11)

Usage:
paroi_adiabatique

11.22 Paroi_contact

Description: Thermal condition between two domains. Important: the name of the boundaries in the two domains should be the same. (Warning: there is also an old limitation not yet fixed on the sequential algorithm in VDF to detect the matching faces on the two boundaries: faces should be ordered in the same way). The kind of condition depends on the discretization. In VDF, it is a heat exchange condition, and in VEF, a temperature condition.
Such a coupling requires coincident meshes for the moment. In case of non-coincident meshes, run is stopped and two external files are automatically generated in VEF (connectivity_failed_boundary_name and connectivity_failed_pb_name.med). In 2D, the keyword Decouper_bord_coincident associated to the connectivity_failed_boundary_name file allows to generate a new coincident mesh.
In 3D, for a first preliminary cut domain with HOMARD (fluid for instance), the second problem associated to pb_name (solide in a fluid/solid coupling problem) has to be submitted to HOMARD cutting procedure with connectivity_failed_pb_name.med.
Such a procedure works as while the primary refined mesh (fluid in our example) impacts the fluid/solid interface with a compact shape as described below (values 2 or 4 indicates the number of division from primary faces obtained in fluid domain at the interface after HOMARD cutting):
2-2-2-2-2-2
2-4-4-4-4-4-2 2-2-2
2-4-4-4-4-2 2-4-2
2-2-2-2-2 2-2
OK

2-2   2-2-2
2-4-2 2-2
2-2 2-2
NOT OK

See also: condlim_base (11)

Usage:
paroi_contact autrepb nameb

where

- **autrepb** *str*: Name of other problem.
- **nameb** *str*: boundary name of the remote problem which should be the same than the local name

11.23 Paroi_contact_fictif

Description: This keyword is derivated from paroi_contact and is especially dedicated to compute coupled fluid/solid/fluid problem in case of thin material. Thanks to this option, solid is considered as a fictitious media (no mesh, no domain associated), and coupling is performed by considering instantaneous thermal equilibrium in it (for the moment).

See also: condlim_base (11)

Usage:
paroi_contact_fictif autrepb nameb conduct_fictif ep_fictive
where

- **autrepb** *str*: Name of other problem.
- **nameb** *str*: Name of bord.
- **conduct_fictif** *float*: thermal conductivity
- **ep_fictive** *float*: thickness of the fictitious media

11.24 Paroi_defilante

Description: Keyword to designate a condition where tangential velocity is imposed on the wall called bord (edge). If the velocity components set by the user is not tangential, projection is used.

See also: dirichlet (11.4)

Usage:
paroi_defilante ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.25 Paroi_echange_contact_correlation_vdf

Description: Class to define a thermohydraulic 1D model which will apply to a boundary of 2D or 3D domain.
Warning : For parallel calculation, the only possible partition will be according the axis of the model with the keyword Tranche.

See also: condlim_base (11)

Usage:
paroi_echange_contact_correlation_vdf *str*
Read *str* {

> **dir** *int*
> **tinf** *float*
> **tsup** *float*
> **lambda** *str*

rho *str*
cp *float*
dt_impr *float*
mu *str*
debit *float*
dh *float*
volume *str*
nu *str*
[**reprise_correlation**]

}
where

- **dir** *int*: Direction (0 : axis X, 1 : axis Y, 2 : axis Z) of the 1D model.
- **tinf** *float*: Inlet fluid temperature of the 1D model (oC or K).
- **tsup** *float*: Outlet fluid temperature of the 1D model (oC or K).
- **lambda** *str*: Thermal conductivity of the fluid (W.m-1.K-1).
- **rho** *str*: Mass density of the fluid (kg.m-3) which may be a function of the temperature T.
- **cp** *float*: Calorific capacity value at a constant pressure of the fluid (J.kg-1.K-1).
- **dt_impr** *float*: Printing period in name_of_data_file_time.dat files of the 1D model results.
- **mu** *str*: Dynamic viscosity of the fluid (kg.m-1.s-1) which may be a function of thetemperature T.
- **debit** *float*: Surface flow rate (kg.s-1.m-2) of the fluid into the channel.
- **dh** *float*: Hydraulic diameter may be a function f(x) with x position along the 1D axis (xinf <= x <= xsup)
- **volume** *str*: Exact volume of the 1D domain (m3) which may be a function of the hydraulic diameter (Dh) and the lateral surface (S) of the meshed boundary.
- **nu** *str*: Nusselt number which may be a function of the Reynolds number (Re) and the Prandtl number (Pr).
- **reprise_correlation** : Keyword in the case of a resuming calculation with this correlation.

11.26 Paroi_echange_contact_correlation_vef

Description: Class to define a thermohydraulic 1D model which will apply to a boundary of 2D or 3D domain.
Warning : For parallel calculation, the only possible partition will be according the axis of the model with the keyword Tranche_geom.

See also: condlim_base (11)

Usage:
paroi_echange_contact_correlation_vef *str*
Read *str* {

dir *int*
tinf *float*
tsup *float*
lambda *str*
rho *str*
cp *float*
dt_impr *float*
mu *str*
debit *float*
dh *float*
n *int*

surface *str*
nu *str*
xinf *float*
xsup *float*
[**emissivite_pour_rayonnement_entre_deux_plaques_quasi_infinies** *float*]
[**reprise_correlation**]

}
where

- **dir** *int*: Direction (0 : axis X, 1 : axis Y, 2 : axis Z) of the 1D model.
- **tinf** *float*: Inlet fluid temperature of the 1D model (oC or K).
- **tsup** *float*: Outlet fluid temperature of the 1D model (oC or K).
- **lambda** *str*: Thermal conductivity of the fluid (W.m-1.K-1).
- **rho** *str*: Mass density of the fluid (kg.m-3) which may be a function of the temperature T.
- **cp** *float*: Calorific capacity value at a constant pressure of the fluid (J.kg-1.K-1).
- **dt_impr** *float*: Printing period in name_of_data_file_time.dat files of the 1D model results.
- **mu** *str*: Dynamic viscosity of the fluid (kg.m-1.s-1) which may be a function of thetemperature T.
- **debit** *float*: Surface flow rate (kg.s-1.m-2) of the fluid into the channel.
- **dh** *float*: Hydraulic diameter may be a function f(x) with x position along the 1D axis (xinf <= x <= xsup)
- **n** *int*: Number of 1D cells of the 1D mesh.
- **surface** *str*: Section surface of the channel which may be function f(Dh,x) of the hydraulic diameter (Dh) and x position along the 1D axis (xinf <= x <= xsup)
- **nu** *str*: Nusselt number which may be a function of the Reynolds number (Re) and the Prandtl number (Pr).
- **xinf** *float*: Position of the inlet of the 1D mesh on the axis direction.
- **xsup** *float*: Position of the outlet of the 1D mesh on the axis direction.
- **emissivite_pour_rayonnement_entre_deux_plaques_quasi_infinies** *float*: Coefficient of emissivity for radiation between two quasi infinite plates.
- **reprise_correlation** : Keyword in the case of a resuming calculation with this correlation.

11.27 Paroi_echange_contact_vdf

Description: Boundary condition type to model the heat flux between two problems. Important: the name of the boundaries in the two problems should be the same.

See also: condlim_base (11)

Usage:
paroi_echange_contact_vdf autrepb nameb temp h
where

- **autrepb** *str*: Name of other problem.
- **nameb** *str*: Name of bord.
- **temp** *str*: Name of field.
- **h** *float*: Value assigned to a coefficient (expressed in W.K-1m-2) that characterises the contact between the two mediums. In order to model perfect contact, h must be taken to be infinite. This value must obviously be the same in both the two problems blocks.
  The surface thermal flux exchanged between the two mediums is represented by :
  fi = h (T1-T2) where 1/h = d1/lambda1 + 1/val_h_contact + d2/lambda2
  where di : distance between the node where Ti and the wall is found.

11.28 Paroi_echange_externe_impose

Description: External type exchange condition with a heat exchange coefficient and an imposed external temperature.

See also: condlim_base (11) paroi_echange_externe_impose_h (11.29)

Usage:
paroi_echange_externe_impose h_imp himpc text ch
where

- **h_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ_front_base* (15.1): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* (15.1): Boundary field type.

11.29 Paroi_echange_externe_impose_h

Description: Particular case of class paroi_echange_externe_impose for enthalpy equation.

See also: paroi_echange_externe_impose (11.28)

Usage:
paroi_echange_externe_impose_h h_imp himpc text ch
where

- **h_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ_front_base* (15.1): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* (15.1): Boundary field type.

11.30 Paroi_echange_global_impose

Description: Global type exchange condition (internal) that is to say that diffusion on the first fluid mesh is not taken into consideration.

See also: condlim_base (11)

Usage:
paroi_echange_global_impose h_imp himpc text ch
where

- **h_imp** *str*: Global exchange coefficient value. The global exchange coefficient value is expressed in W.m-2.K-1.
- **himpc** *champ_front_base* (15.1): Boundary field type.
- **text** *str*: External temperature value. The external temperature value is expressed in oC or K.
- **ch** *champ_front_base* (15.1): Boundary field type.

11.31 Paroi_fixe

Description: Keyword to designate a situation of adherence to the wall called bord (edge) (normal and tangential velocity at the edge is zero).

See also: condlim_base (11) paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets (11.32)

Usage:
paroi_fixe

11.32 Paroi_fixe_iso_genepi2_sans_contribution_aux_vitesses_sommets

Description: Boundary condition to obtain iso Geneppi2, without interest

See also: paroi_fixe (11.31)

Usage:
paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets

11.33 Paroi_flux_impose

Description: Normal flux condition at the wall called bord (edge). The surface area of the flux (W.m-1 in 2D or W.m-2 in 3D) is imposed at the boundary according to the following convention: a positive flux is a flux that enters into the domain according to convention.

See also: condlim_base (11)

Usage:
paroi_flux_impose ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.34 Paroi_knudsen_non_negligeable

Description: Boundary condition for number of Knudsen (Kn) above 0.001 where slip-flow condition appears: the velocity near the wall depends on the shear stress : Kn=l/L with l is the mean-free-path of the molecules and L a characteristic length scale.
U(y=0)-Uwall=k(dU/dY)
Where k is a coefficient given by several laws:
Mawxell : k=(2-s)*l/s
Bestok&Karniadakis :k=(2-s)/s*L*Kn/(1+Kn)
Xue&Fan :k=(2-s)/s*L*tanh(Kn)
s is a value between 0 and 2 named accomodation coefficient. s=1 seems a good value.
Warning : The keyword is available for VDF calculation only for the moment.

See also: dirichlet (11.4)

Usage:
paroi_knudsen_non_negligeable name_champ_1 champ_1 name_champ_2 champ_2
where

- **name_champ_1** *str into ['vitesse_paroi', 'k']*: Field name.
- **champ_1** *champ_front_base* (15.1): Boundary field type.
- **name_champ_2** *str into ['vitesse_paroi', 'k']*: Field name.
- **champ_2** *champ_front_base* (15.1): Boundary field type.

11.35 Paroi_temperature_imposee

Description: Imposed temperature condition at the wall called bord (edge).

See also: dirichlet (11.4) temperature_imposee_paroi (11.40)

Usage:
paroi_temperature_imposee ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.36 Periodique

Description: 1). For Navier-Stokes equations, this keyword is used to indicate that the horizontal inlet velocity values are the same as the outlet velocity values, at every moment. As regards meshing, the inlet and outlet edges bear the same name.; 2). For scalar transport equation, this keyword is used to set a periodic condition on scalar. The two edges dealing with this periodic condition bear the same name.

See also: condlim_base (11)

Usage:
periodique

11.37 Scalaire_impose_paroi

Description: Imposed temperature condition at the wall called bord (edge).

See also: dirichlet (11.4)

Usage:
scalaire_impose_paroi ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.38 Sortie_libre_temperature_imposee_h

Description: Open boundary for heat equation with enthalpy as unknown.

See also: neumann (11.20)

Usage:
sortie_libre_temperature_imposee_h ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

11.39 Symetrie

Description: 1). For Navier-Stokes equations, this keyword is used to designate a symmetry condition concerning the velocity at the boundary called bord (edge) (normal velocity at the edge equal to zero and tangential velocity gradient at the edge equal to zero); 2). For scalar transport equation, this keyword is used to set a symmetry condition on scalar on the boundary named bord (edge).

See also: condlim_base (11)

Usage:
symetrie

11.40 Temperature_imposee_paroi

Description: Imposed temperature condition at the wall called bord (edge).

See also: paroi_temperature_imposee (11.35)

Usage:
temperature_imposee_paroi ch
where

- **ch** *champ_front_base* (15.1): Boundary field type.

12 discretisation_base

Description: Basic class for space discretization of thermohydraulic turbulent problems.

See also: objet_u (34) vdf (12.4) vef (12.5) covimac (12.1) polymac (12.3) ef (12.2)

Usage:

12.1 Covimac

Description: covimac discretization.

See also: discretisation_base (12)

Usage:

12.2 Ef

Description: Element Finite discretization.

See also: discretisation_base (12)

Usage:

12.3 Polymac

Description: polymac discretization.

See also: discretisation_base (12)

Usage:

12.4 Vdf

Description: Finite difference volume discretization.

See also: discretisation_base (12)

Usage:

12.5 Vef

Description: Finite element volume discretization (P1NC/P0 element)
Warning: it becomes an obsolete discretization.

See also: discretisation_base (12) vefprep1b (12.6)

Usage:

12.6 Vefprep1b

Description: Finite element volume discretization (P1NC/P1-bubble element). Since the 1.5.5 version, several new discretizations are available thanks to the optional keyword Read. By default, the VEFPreP1B keyword is equivalent to the former VEFPreP1B formulation (v1.5.4 and sooner). P0P1 (if used with the strong formulation for imposed pressure boundary) is equivalent to VEFPreP1B but the convergence is slower. VEFPreP1B dis is equivalent to VEFPreP1B dis Read dis { P0 P1 Changement_de_base_P1Bulle 1 Cl_pression_sommet_faible 0 }

See also: vef (12.5)

Usage:
vefprep1b *str*
Read *str* {

[**changement_de_base_p1bulle** *int*]
[**p0**]
[**p1**]
[**pa**]
[**modif_div_face_dirichlet** *int*]
[**cl_pression_sommet_faible** *int*]

}
where

- **changement_de_base_p1bulle** *int*: (into=[0,1]) changement_de_base_p1bulle 1 This option may be used to have the P1NC/P0P1 formulation (value set to 0) or the P1NC/P1Bulle formulation (value set to 1, the default).
- **p0** : Pressure nodes are added on element centres
- **p1** : Pressure nodes are added on vertices
- **pa** : Only available in 3D, pressure nodes are added on bones
- **modif_div_face_dirichlet** *int*: (into=[0,1]) This option (by default 0) is used to extend control volumes for the momentum equation.

- **cl_pression_sommet_faible** *int*: (into=[0,1]) This option is used to specify a strong formulation (value set to 0, the default) or a weak formulation (value set to 1) for an imposed pressure boundary condition. The first formulation converges quicker and is stable in general cases. The second formulation should be used if there are several outlet boundaries with Neumann condition (see Ecoulement_Neumann test case for example).

13 domaine

Description: Keyword to create a domain.

See also: objet_u (34)

Usage:

14 champ_base

14.1 Champ_base

Description: Basic class of fields.

See also: objet_u (34) champ_don_base (14.6) champ_ostwald (14.20) champ_input_base (14.18) champ_fonc_med (14.11)

Usage:

14.2 Champ_fonc_med_tabule

Description: not_set

See also: champ_fonc_med (14.11)

Usage:
Champ_Fonc_MED_Tabule [**use_existing_domain**] [**last_time**] [**option**] **filename** **domain_name** **field_name** **location** **time**
where

- **use_existing_domain** *str into ['use_existing_domain']*
- **last_time** *str into ['last_time']*: to use the last time of the MED file instead of the specified time.
- **option** *decoup* (14.3): Keyword for a partition file
- **filename** *str*: Name of the .med file.
- **domain_name** *str*: Name of the domain.
- **field_name** *str*: Name of the problem unknown.
- **location** *str into ['som', 'elem']*: To indicate where the field has been post-processed.
- **time** *float*: Time of the field in the .med file.

14.3 Decoup

Description: Optional keyword

See also: objet_lecture (33)

Usage:
key **nom**

where

- **key** *str into ['decoup']*: Name of for a partition file
- **nom** *str*: Name of for a partition file

14.4 Champ_fonc_medfile

Description: Obsolete keyword to read a field with MED file API

See also: champ_fonc_med (14.11)

Usage:
Champ_Fonc_MEDfile [**use_existing_domain**] [**last_time**] [**option**] **filename** **domain_name** **field_name** **location** **time**
where

- **use_existing_domain** *str into ['use_existing_domain']*
- **last_time** *str into ['last_time']*: to use the last time of the MED file instead of the specified time.
- **option** *decoup* (14.3): Keyword for a partition file
- **filename** *str*: Name of the .med file.
- **domain_name** *str*: Name of the domain.
- **field_name** *str*: Name of the problem unknown.
- **location** *str into ['som', 'elem']*: To indicate where the field has been post-processed.
- **time** *float*: Time of the field in the .med file.

14.5 Champ_tabule_morceaux

Description: set Tabulated field by sub-zone

See also: champ_don_base (14.6)

Usage:
Champ_Tabule_Morceaux **dom_name** **nb_comp** **data**
where

- **dom_name** *str*: Name of the domain
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* (3.17): subzone_1 nb_comp InputFieldName { table_dim InputFieldVal_1 InputFieldVal-_2 .... OutputFieldVal_1 OutputFieldVal_2 ... } subzone_2 nb_comp InputFieldName { table-_dim InputFieldVal_1 InputFieldVal_2 .... OutputFieldVal_1 OutputFieldVal_2 ... } ..... subzone-_n nb_comp InputFieldName { table_dim InputFieldVal_1 InputFieldVal_2 .... OutputFieldVal_1 OutputFieldVal_2 ... }

14.6 Champ_don_base

Description: Basic class for data fields (not calculated), p.e. physics properties.

See also: champ_base (14.1) uniform_field (14.30) champ_uniforme_morceaux (14.24) champ_fonc_xyz (14.27) champ_fonc_txyz (14.26) champ_don_lu (14.7) init_par_partie (14.28) champ_tabule_temps (14.23) champ_fonc_t (14.14) champ_fonc_tabule (14.15) champ_init_canal_sinal (14.16) champ_som_lu_vdf (14.21) champ_som_lu_vef (14.22) tayl_green (14.29) Champ_Tabule_Morceaux (14.5) champ_fonc_fonction-_txyz_morceaux (14.10) champ_fonc_reprise (14.12)

Usage:

14.7 Champ_don_lu

Description: Field to read a data field (values located at the center of the cells) in a file.

See also: champ_don_base (14.6)

Usage:
champ_don_lu dom nb_comp file
where

- **dom** *str*: Name of the domain.
- **nb_comp** *int*: Number of field components.
- **file** *str*: Name of the file.
  This file has the following format:
  nb_val_lues -> Number of values readen in th file
  Xi Yi Zi -> Coordinates readen in the file
  Ui Vi Wi -> Value of the field

14.8 Champ_fonc_fonction

Description: Field that is a function of another field.

See also: champ_fonc_tabule (14.15) champ_fonc_fonction_txyz (14.9)

Usage:
champ_fonc_fonction problem_name inco expression
where

- **problem_name** *str*: Name of problem.
- **inco** *str*: Name of the field (for example: temperature).
- **expression** *n word1 word2 ... wordn*: Number of field components followed by the analytical expression for each field component.

14.9 Champ_fonc_fonction_txyz

Description: this refers to a field that is a function of another field and time and/or space coordinates

See also: champ_fonc_fonction (14.8)

Usage:
champ_fonc_fonction_txyz problem_name inco expression
where

- **problem_name** *str*: Name of problem.
- **inco** *str*: Name of the field (for example: temperature).
- **expression** *n word1 word2 ... wordn*: Number of field components followed by the analytical expression for each field component.

14.10 Champ_fonc_fonction_txyz_morceaux

Description: Field defined by analytical functions in each sub-zone. It makes possible the definition of a field that depends on the time and the space.

See also: champ_don_base (14.6)

Usage:
champ_fonc_fonction_txyz_morceaux problem_name inco nb_comp data
where

- **problem_name** *str*: Name of the problem.
- **inco** *str*: Name of the field (for example: temperature).
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* (3.17): { Defaut val_def sous_zone_1 val_1 ... sous_zone_i val_i } By default, the value val_def is assigned to the field. It takes the sous_zone_i identifier Sous_Zone (sub_area) type object function, val_i. Sous_Zone (sub_area) type objects must have been previously defined if the operator wishes to use a champ_fonc_fonction_txyz_morceaux type object.

14.11 Champ_fonc_med

Description: Field to read a data field in a MED-format file .med at a specified time. It is very useful, for example, to resume a calculation with a new or refined geometry. The field post-processed on the new geometry at med format is used as initial condition for the resume.

See also: champ_base (14.1) Champ_Fonc_MEDfile (14.4) Champ_Fonc_MED_Tabule (14.2)

Usage:
champ_fonc_med [**use_existing_domain**] [**last_time**] [**option**] **filename domain_name field-_name location time**
where

- **use_existing_domain** *str into ['use_existing_domain']*
- **last_time** *str into ['last_time']*: to use the last time of the MED file instead of the specified time.
- **option** *decoup* (14.3): Keyword for a partition file
- **filename** *str*: Name of the .med file.
- **domain_name** *str*: Name of the domain.
- **field_name** *str*: Name of the problem unknown.
- **location** *str into ['som', 'elem']*: To indicate where the field has been post-processed.
- **time** *float*: Time of the field in the .med file.

14.12 Champ_fonc_reprise

Description: This field is used to read a data field in a save file (.xyz or .sauv) at a specified time. It is very useful, for example, to run a thermohydraulic calculation with velocity initial condition read into a save file from a previous hydraulic calculation.

See also: champ_don_base (14.6)

Usage:
champ_fonc_reprise [**format**] **filename pb_name champ** [**fonction**] **temps**
where

- **format** *str into ['binaire', 'formatte', 'xyz', 'single_hdf']*: Type of file (the file format). If xyz format is activated, the .xyz file from the previous calculation will be given for filename, and if formatte or binaire is choosen, the .sauv file of the previous calculation will be specified for filename. In the case of a parallel calculation, if the mesh partition does not changed between the previous calculation and the next one, the binaire format should be preferred, because is faster than the xyz format. If single-_hdf is used, the same constraints/advantages as binaire apply, but a single (HDF5) file is produced on the filesystem instead of having one file per processor.

- **filename** *str*: Name of the save file.
- **pb_name** *str*: Name of the problem.
- **champ** *str*: Name of the problem unknown. It may also be the temporal average of a problem unknown (like moyenne_vitesse, moyenne_temperature,...)
- **fonction** *fonction_champ_reprise* (14.13): Optional keyword to apply a function on the field being read in the save file (e.g. to read a temperature field in Celsius units and convert it for the calculation on Kelvin units, you will use: fonction 1 273.+val)
- **temps** *str*: Time of the saved field in the save file or last_time. If you give the keyword last_time instead, the last time saved in the save file will be used.

14.13 Fonction_champ_reprise

Description: not_set

See also: objet_lecture (33)

Usage:
mot fonction
where

- **mot** *str into ['fonction']*
- **fonction** *n word1 word2 ... wordn*: n f1(val) f2(val) ... fn(val)] time

14.14 Champ_fonc_t

Description: Field that is constant in space and is a function of time.

See also: champ_don_base (14.6)

Usage:
champ_fonc_t val
where

- **val** *n word1 word2 ... wordn*: Values of field components (time dependant functions).

14.15 Champ_fonc_tabule

Description: Field that is tabulated as a function of another field.

See also: champ_don_base (14.6) champ_fonc_fonction (14.8)

Usage:
champ_fonc_tabule inco dim bloc
where

- **inco** *str*: Name of the field (for example: temperature).
- **dim** *int*: Number of field components.
- **bloc** *bloc_lecture* (3.17): Values (the table (the value of the field at any time is calculated by linear interpolation from this table) or the analytical expression (with keyword expression to use an analytical expression)).

14.16 Champ_init_canal_sinal

Description: For a parabolic profile on U velocity with an unpredictable disturbance on V and W and a sinusoidal disturbance on V velocity.

See also: champ_don_base (14.6)

Usage:
champ_init_canal_sinal dim bloc
where

- **dim** *int*: Number of field components.
- **bloc** *bloc_lec_champ_init_canal_sinal* (14.17): Parameters for the class champ_init_canal_sinal.

14.17 Bloc_lec_champ_init_canal_sinal

Description: Parameters for the class champ_init_canal_sinal.
in 2D:
U=ucent*y(2h-y)/h/h
V=ampli_bruit*rand+ampli_sin*sin(omega*x)
rand: unpredictable value between -1 and 1.
in 3D:
U=ucent*y(2h-y)/h/h
V=ampli_bruit*rand1+ampli_sin*sin(omega*x)
W=ampli_bruit*rand2
rand1 and rand2: unpredictables values between -1 and 1.

See also: objet_lecture (33)

Usage:
{

> **ucent** *float*
> **h** *float*
> **ampli_bruit** *float*
> [**ampli_sin** *float*]
> **omega** *float*
> [**dir_flow** *int into [0, 1, 2]*]
> [**dir_wall** *int into [0, 1, 2]*]
> [**min_dir_flow** *float*]
> [**min_dir_wall** *float*]

}
where

- **ucent** *float*: Velocity value at the center of the channel.
- **h** *float*: Half hength of the channel.
- **ampli_bruit** *float*: Amplitude for the disturbance.
- **ampli_sin** *float*: Amplitude for the sinusoidal disturbance (by default equals to ucent/10).
- **omega** *float*: Value of pulsation for the of the sinusoidal disturbance.
- **dir_flow** *int into [0, 1, 2]*: Flow direction for the initialization of the flow in a channel.
  - if dir_flow=0, the flow direction is X
  - if dir_flow=1, the flow direction is Y
  - if dir_flow=2, the flow direction is Z
  Default value for dir_flow is 0

- **dir_wall** *int into [0, 1, 2]*: Wall direction for the initialization of the flow in a channel.
  - if dir_wall=0, the normal to the wall is in X direction
  - if dir_wall=1, the normal to the wall is in Y direction
  - if dir_wall=2, the normal to the wall is in Z direction
  Default value for dir_flow is 1
- **min_dir_flow** *float*: Value of the minimum coordinate in the flow direction for the initialization of the flow in a channel. Default value for dir_flow is 0.
- **min_dir_wall** *float*: Value of the minimum coordinate in the wall direction for the initialization of the flow in a channel. Default value for dir_flow is 0.

14.18 Champ_input_base

Description: not_set

See also: champ_base (14.1) champ_input_p0 (14.19)

Usage:
champ_input_base *str*
Read *str* {

> **nb_comp** *int*
> **nom** *str*
> [**initial_value** *n x1 x2 ... xn*]
> **probleme** *str*
> [**sous_zone** *str*]

}
where

- **nb_comp** *int*
- **nom** *str*
- **initial_value** *n x1 x2 ... xn*
- **probleme** *str*
- **sous_zone** *str*

14.19 Champ_input_p0

Description: not_set

See also: champ_input_base (14.18)

Usage:
champ_input_p0 *str*
Read *str* {

> **nb_comp** *int*
> **nom** *str*
> [**initial_value** *n x1 x2 ... xn*]
> **probleme** *str*
> [**sous_zone** *str*]

}
where

- **nb_comp** *int* for inheritance

- **nom** *str* for inheritance
- **initial_value** *n x1 x2 ... xn* for inheritance
- **probleme** *str* for inheritance
- **sous_zone** *str* for inheritance

14.20 Champ_ostwald

Description: This keyword is used to define the viscosity variation law:
Mu(T)= K(T)*(D:D/2)**((n-1)/2)

See also: champ_base (14.1)

Usage:
champ_ostwald

14.21 Champ_som_lu_vdf

Description: Keyword to read in a file values located at the nodes of a mesh in VDF discretization.

See also: champ_don_base (14.6)

Usage:
champ_som_lu_vdf domain_name dim tolerance file
where

- **domain_name** *str*: Name of the domain.
- **dim** *int*: Value of the dimension of the field.
- **tolerance** *float*: Value of the tolerance to check the coordinates of the nodes.
- **file** *str*: name of the file
  This file has the following format:
  Xi Yi Zi -> Coordinates of the node
  Ui Vi Wi -> Value of the field on this node
  Xi+1 Yi+1 Zi+1 -> Next point
  Ui+1 Vi+1 Zi+1 -> Next value ...

14.22 Champ_som_lu_vef

Description: Keyword to read in a file values located at the nodes of a mesh in VEF discretization.

See also: champ_don_base (14.6)

Usage:
champ_som_lu_vef domain_name dim tolerance file
where

- **domain_name** *str*: Name of the domain.
- **dim** *int*: Value of the dimension of the field.
- **tolerance** *float*: Value of the tolerance to check the coordinates of the nodes.
- **file** *str*: Name of the file.
  This file has the following format:
  Xi Yi Zi -> Coordinates of the node
  Ui Vi Wi -> Value of the field on this node
  Xi+1 Yi+1 Zi+1 -> Next point
  Ui+1 Vi+1 Zi+1 -> Next value ...

14.23 Champ_tabule_temps

Description: Field that is constant in space and tabulated as a function of time.

See also: champ_don_base (14.6)

Usage:
champ_tabule_temps dim bloc
where

- **dim** *int*: Number of field components.
- **bloc** *bloc_lecture* (3.17): Values as a table. The value of the field at any time is calculated by linear interpolation from this table.

14.24 Champ_uniforme_morceaux

Description: Field which is partly constant in space and stationary.

See also: champ_don_base (14.6) champ_uniforme_morceaux_tabule_temps (14.25) valeur_totale_sur-_volume (14.31)

Usage:
champ_uniforme_morceaux nom_dom nb_comp data
where

- **nom_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* (3.17): { Defaut val_def sous_zone_1 val_1 ... sous_zone_i val_i } By default, the value val_def is assigned to the field. It takes the sous_zone_i identifier Sous_Zone (sub_area) type object value, val_i. Sous_Zone (sub_area) type objects must have been previously defined if the operator wishes to use a Champ_Uniforme_Morceaux(partly_uniform_field) type object.

14.25 Champ_uniforme_morceaux_tabule_temps

Description: this type of field is constant in space on one or several sub_zones and tabulated as a function of time.

See also: champ_uniforme_morceaux (14.24)

Usage:
champ_uniforme_morceaux_tabule_temps nom_dom nb_comp data
where

- **nom_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* (3.17): { Defaut val_def sous_zone_1 val_1 ... sous_zone_i val_i } By default, the value val_def is assigned to the field. It takes the sous_zone_i identifier Sous_Zone (sub_area) type object value, val_i. Sous_Zone (sub_area) type objects must have been previously defined if the operator wishes to use a Champ_Uniforme_Morceaux(partly_uniform_field) type object.

14.26 Champ_fonc_txyz

Description: Field defined by analytical functions. It makes it possible the definition of a field that depends on the time and the space.

See also: champ_don_base (14.6)

Usage:
champ_fonc_txyz dom val
where

- **dom** *str*: Name of domain of calculation.
- **val** *n word1 word2 ... wordn*: List of functions on (t,x,y,z).

14.27 Champ_fonc_xyz

Description: Field defined by analytical functions. It makes it possible the definition of a field that depends on (x,y,z).

See also: champ_don_base (14.6)

Usage:
champ_fonc_xyz dom val
where

- **dom** *str*: Name of domain of calculation.
- **val** *n word1 word2 ... wordn*: List of functions on (x,y,z).

14.28 Init_par_partie

Description: ne marche que pour n_comp=1

See also: champ_don_base (14.6)

Usage:
init_par_partie n_comp val1 val2 val3
where

- **n_comp** *int into [1]*
- **val1** *float*
- **val2** *float*
- **val3** *float*

14.29 Tayl_green

Description: Class Tayl_green.

See also: champ_don_base (14.6)

Usage:
tayl_green dim
where

- **dim** *int*: Dimension.

14.30 Uniform_field

Synonymous: **champ_uniforme**

Description: Field that is constant in space and stationary.

See also: champ_don_base (14.6)

Usage:
uniform_field val
where

- **val** *n x1 x2 ... xn*: Values of field components.

14.31 Valeur_totale_sur_volume

Description: Similar as Champ_Uniforme_Morceaux with the same syntax. Used for source terms when we want to specify a source term with a value given for the volume (eg: heat in Watts) and not a value per volume unit (eg: heat in Watts/m3).

See also: champ_uniforme_morceaux (14.24)

Usage:
valeur_totale_sur_volume nom_dom nb_comp data
where

- **nom_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* (3.17): { Defaut val_def sous_zone_1 val_1 ... sous_zone_i val_i } By default, the value val_def is assigned to the field. It takes the sous_zone_i identifier Sous_Zone (sub_area) type object value, val_i. Sous_Zone (sub_area) type objects must have been previously defined if the operator wishes to use a Champ_Uniforme_Morceaux(partly_uniform_field) type object.

15 champ_front_base

15.1 Champ_front_base

Description: Basic class for fields at domain boundaries.

See also: objet_u (34) champ_front_uniforme (15.25) champ_front_fonc_xyz (15.17) champ_front_fonc-_txyz (15.16) champ_front_fonc_pois_ipsn (15.13) champ_front_fonc_pois_tube (15.14) champ_front-_tabule (15.23) champ_front_fonction (15.18) champ_front_bruite (15.8) champ_front_tangentiel_vef (15.24) champ_front_lu (15.19) boundary_field_inward (15.4) champ_front_pression_from_u (15.21) champ_front-_contact_vef (15.10) champ_front_calc (15.9) champ_front_recyclage (15.22) ch_front_input (15.5) champ-_front_normal_vef (15.20) Champ_front_debit_QC_VDF_fonc_t (15.3) Champ_front_debit_QC_VDF (15.2) champ_front_MED (15.7) champ_front_debit_massique (15.12) champ_front_debit (15.11) champ_front-_xyz_debit (15.26) champ_front_fonc_t (15.15)

Usage:

15.2 Champ_front_debit_qc_vdf

Description: This keyword is used to define a flow rate field for quasi-compressible fluids in VDF discretization. The flow rate is kept constant during a transient.

See also: champ_front_base (15.1)

Usage:
Champ_front_debit_QC_VDF dimension liste [**moyen**] **pb_name**
where

- **dimension** *int*: Problem dimension
- **liste** *bloc_lecture* (3.17): List of the mass flow rate values [kg/s/m2] with the following syntaxe: { val1 ... valdim }
- **moyen** *str*: Option to use rho mean value
- **pb_name** *str*: Problem name

15.3 Champ_front_debit_qc_vdf_fonc_t

Description: This keyword is used to define a flow rate field for quasi-compressible fluids in VDF discretization. The flow rate could be constant or time-dependent.

See also: champ_front_base (15.1)

Usage:
Champ_front_debit_QC_VDF_fonc_t dimension liste [**moyen**] **pb_name**
where

- **dimension** *int*: Problem dimension
- **liste** *bloc_lecture* (3.17): List of the mass flow rate values [kg/s/m2] with the following syntaxe: { val1 ... valdim } where val1 ... valdim are constant or function of time.
- **moyen** *str*: Option to use rho mean value
- **pb_name** *str*: Problem name

15.4 Boundary_field_inward

Description: this field is used to define the normal vector field standard at the boundary in VDF or VEF discretization.

See also: champ_front_base (15.1)

Usage:
boundary_field_inward *str*
Read *str* {

normal_value *str*

}
where

- **normal_value** *str*: normal vector value (positive value for a vector oriented outside to inside) which can depend of the time.

15.5 Ch_front_input

Description: not_set

See also: champ_front_base (15.1) ch_front_input_uniforme (15.6)

Usage:
ch_front_input *str*
Read *str* {

> **nb_comp** *int*
> **nom** *str*
> [**initial_value** *n x1 x2 ... xn*]
> **probleme** *str*
> [**sous_zone** *str*]

}
where

- **nb_comp** *int*
- **nom** *str*
- **initial_value** *n x1 x2 ... xn*
- **probleme** *str*
- **sous_zone** *str*

15.6 Ch_front_input_uniforme

Description: for coupling, you can use ch_front_input_uniforme which is a champ_front_uniforme, which use an external value. It must be used with Problem.setInputField.

See also: ch_front_input (15.5)

Usage:
ch_front_input_uniforme *str*
Read *str* {

> **nb_comp** *int*
> **nom** *str*
> [**initial_value** *n x1 x2 ... xn*]
> **probleme** *str*
> [**sous_zone** *str*]

}
where

- **nb_comp** *int* for inheritance
- **nom** *str* for inheritance
- **initial_value** *n x1 x2 ... xn* for inheritance
- **probleme** *str* for inheritance
- **sous_zone** *str* for inheritance

15.7 Champ_front_med

Description: Field allowing the loading of a boundary condition from a MED file using Champ_fonc_med

See also: champ_front_base (15.1)

Usage:
champ_front_MED champ_fonc_med
where

- **champ_fonc_med** *champ_base* (14.1): a champ_fonc_med loading the values of the unknown on a domain boundary

15.8 Champ_front_bruite

Description: Field which is variable in time and space in a random manner.

See also: champ_front_base (15.1)

Usage:
champ_front_bruite nb_comp bloc
where

- **nb_comp** *int*: Number of field components.
- **bloc** *bloc_lecture* (3.17): { [N val L val] Moyenne m_1.....[m_i] Amplitude A_1.....[A_ i]}: Random nois: If N and L are not defined, the ith component of the field varies randomly around an average value m_i with a maximum amplitude A_i.
  White noise: If N and L are defined, these two additional parameters correspond to L, the domain length and N, the number of nodes in the domain. Noise frequency will be between 2*Pi/L and 2*Pi*N/(4*L).
  For example, formula for velocity: u=U0(t) v=U1(t)Uj(t)=Mj+2*Aj*bruit_blanc where bruit_blanc (white_noise) is the formula given in the mettre_a_jour (update) method of the Champ_front_bruite (noise_boundary_field) (Refer to the Ch_fr_bruite.cpp file).

15.9 Champ_front_calc

Description: This keyword is used on a boundary to get a field from another boundary. The local and remote boundaries should have the same mesh. If not, the Champ_front_recyclage keyword could be used instead. It is used in the condition block at the limits of equation which itself refers to a problem called pb1. We are working under the supposition that pb1 is coupled to another problem.

See also: champ_front_base (15.1)

Usage:
champ_front_calc problem_name bord field_name
where

- **problem_name** *str*: Name of the other problem to which pb1 is coupled.
- **bord** *str*: Name of the side which is the boundary between the 2 domains in the domain object description associated with the problem_name object.
- **field_name** *str*: Name of the field containing the value that the user wishes to use at the boundary. The field_name object must be recognized by the problem_name object.

15.10 Champ_front_contact_vef

Description: This field is used on a boundary between a solid and fluid domain to exchange a calculated temperature at the contact face of the two domains according to the flux of the two problems.

See also: champ_front_base (15.1)

Usage:
champ_front_contact_vef local_pb local_boundary remote_pb remote_boundary
where

- **local_pb** *str*: Name of the problem.
- **local_boundary** *str*: Name of the boundary.
- **remote_pb** *str*: Name of the second problem.
- **remote_boundary** *str*: Name of the boundary in the second problem.

15.11 Champ_front_debit

Description: This field is used to define a flow rate field instead of a velocity field for a Dirichlet boundary condition on Navier-Stokes equations.

See also: champ_front_base (15.1)

Usage:
champ_front_debit ch
where

- **ch** *champ_front_base* (15.1): uniform field in space to define the flow rate. It could be, for example, champ_front_uniforme, ch_front_input_uniform or champ_front_fonc_txyz that depends only on time.

15.12 Champ_front_debit_massique

Description: This field is used to define a flow rate field using the density

See also: champ_front_base (15.1)

Usage:
champ_front_debit_massique ch
where

- **ch** *champ_front_base* (15.1): uniform field in space to define the flow rate. It could be, for example, champ_front_uniforme, ch_front_input_uniform or champ_front_fonc_txyz that depends only on time.

15.13 Champ_front_fonc_pois_ipsn

Description: Boundary field champ_front_fonc_pois_ipsn.

See also: champ_front_base (15.1)

Usage:
champ_front_fonc_pois_ipsn r_tube umoy r_loc
where

- **r_tube** *float*
- **umoy** *n x1 x2 ... xn*
- **r_loc** *x1 x2 (x3)*

15.14 Champ_front_fonc_pois_tube

Description: Boundary field champ_front_fonc_pois_tube.

See also: champ_front_base (15.1)

Usage:
champ_front_fonc_pois_tube r_tube umoy r_loc r_loc_mult
where

- **r_tube** *float*
- **umoy** *n x1 x2 ... xn*
- **r_loc** *x1 x2 (x3)*
- **r_loc_mult** *n1 n2 (n3)*

15.15 Champ_front_fonc_t

Description: Boundary field that depends only on time.

See also: champ_front_base (15.1)

Usage:
champ_front_fonc_t val
where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

15.16 Champ_front_fonc_txyz

Description: Boundary field which is not constant in space and in time.

See also: champ_front_base (15.1)

Usage:
champ_front_fonc_txyz val
where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

15.17 Champ_front_fonc_xyz

Description: Boundary field which is not constant in space.

See also: champ_front_base (15.1)

Usage:
champ_front_fonc_xyz val
where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

15.18 Champ_front_fonction

Description: boundary field that is function of another field

See also: champ_front_base (15.1)

Usage:
champ_front_fonction dim inco expression
where

- **dim** *int*: Number of field components.
- **inco** *str*: Name of the field (for example: temperature).
- **expression** *str*: keyword to use a analytical expression like 10.*EXP(-0.1*val) where val be the keyword for the field.

15.19 Champ_front_lu

Description: boundary field which is given from data issued from a read file. The format of this file has to be the same that the one generated by Ecrire_fichier_xyz_valeur
Example for K and epsilon quantities to be defined for inlet condition in a boundary named 'entree':
entree frontiere_ouverte_K_Eps_impose Champ_Front_lu dom 2pb_K_EPS_PERIO_1006.306198.dat

See also: champ_front_base (15.1)

Usage:
champ_front_lu domaine dim file
where

- **domaine** *str*: Name of domain
- **dim** *int*: number of components
- **file** *str*: path for the read file

15.20 Champ_front_normal_vef

Description: Field to define the normal vector field standard at the boundary in VEF discretization.

See also: champ_front_base (15.1)

Usage:
champ_front_normal_vef mot vit_tan
where

- **mot** *str into ['valeur_normale']*: Name of vector field.
- **vit_tan** *float*: normal vector value (positive value for a vector oriented outside to inside).

15.21 Champ_front_pression_from_u

Description: this field is used to define a pressure field depending of a velocity field.

See also: champ_front_base (15.1)

Usage:
champ_front_pression_from_u expression
where

- **expression** *str*: value depending of a velocity (like $2 * u\_moy^2$).

15.22 Champ_front_recyclage

Description: This keyword is used on a boundary to get a field from another boundary. New keyword since the 1.6.1 version which replaces and generalizes several obsolete ones:

Champ_front_calc_intern
Champ_front_calc_recycl_fluct_pbperio
Champ_front_calc_recycl_champ
Champ_front_calc_intern_2pbs
Champ_front_calc_recycl_fluct

It is to use, in a general way, on a boundary of a local_pb problem, a field calculated from a linear combination of an imposed field g(x,y,z,t) with an instantaneous f(x,y,z,t) and a spatial mean field <f>(t) or a temporal mean field <f>(x,y,z) extracted from a plane of a problem named pb (pb may be local_pb itself):
For each component i, the field F applied on the boundary will be:
F_i(x,y,z,t) = alpha_i*g_i(x,y,z,t) + xsi_i*[f_i(x,y,z,t)- beta_i*<fi>]

Usage:
Champ_front_recyclage {

pb_champ_evaluateur *problem_name field nb_comp*
[**distance_plan** *x1 x2 (x3)*]
[**moyenne_imposee** *methode_moy* [**fichier** *file [second_file]*]]
[**moyenne_recyclee** *methode_recyc* [**fichier** *file [second_file]*]]
[**direction_anisotrope** *int*]
[**ampli_moyenne_imposee** *n x1 x2 ... xn*]
[**ampli_moyenne_recyclee** *n x1 x2 ... xn*]
[**ampli_fluctuation** *n x1 x2 ... xn*]

}
where:

- **pb_champ_evaluateur** *problem_name field nb_comp*: To give the name of the problem, the name of the field of the problem and its number of components nb_comp.
- **distance_plan** *x1 x2 (x3)*: Vector which gives the distance between the boundary and the plane from where the field F will be extracted. By default, the vector is zero, that should imply the two domains have coincident boundaries.
- **ampli_moyenne_imposee** *2|3 alpha(0) alpha(1) [alpha(2)]*: alpha_i coefficients (by default =1)
- **ampli_moyenne_recyclee** *2|3 beta(0) beta(1) [beta(2)]*: beta_i coefficients (by default =1)
- **ampli_fluctuation** *2|3 gamma(0) gamma(1) [gamma(2)]*: gamma_i coefficients (by default =1)
- **direction_anisotrope** *int into [1,2,3]*: If an integer is given for direction (X:1, Y:2, Z:3, by default, direction is negative), the imposed field g will be 0 for the 2 other directions.
- **moyenne_imposee** *methode_moy*: Value of the imposed g field. The *methode_moy* option can be:

  profil *[2|3] valx(x,y,z,t) valy(x,y,z,t) [valz(x,y,z,t)]*: To specify analytic profile for the imposed g field.

  interpolation fichier *file*: To create an imposed field built by interpolation of values read from a file. The imposed field is applied on the direction given by the keyword direction_anisotrope (the field is zero for the other directions). The format of the file is:

```
pos(1) val(1)
pos(2) val(2)
...
pos(N) val(N)
```

If direction given by direction_anisotrope is 1 (or 2 or 3), then pos will be X (or Y or Z) coordinate and val will be X value (or Y value, or Z value) of the imposed field.

connexion_approchee fichier *file*: To read the imposed field from a file where positions and values are given (it is not necessary that the coordinates of points match the coordinates of the boundary faces, indeed, the nearest point of each face of the boundary will be used). The format of the file is:

```
N
x(1) y(1) [z(1)] valx(1) valy(1) [valz(1)]
x(2) y(2) [z(2)] valx(2) valy(2) [valz(2)]
...
x(N) y(N) [z(N)] valx(N) valy(N) [valz(N)]
```

connection_exacte fichier *file second_file*: To read the imposed field from two files. The first file contains the points coordinates (which should be the same as the coordinates of the boundary faces) and the second_file contains the mean values. The format of the first file is:

```
N
1 x(1) y(1) [z(1)]
2 x(2) y(2) [z(2)]
...
N x(N) y(N) [z(N)]
```

while the format of the second_file is:

```
N
1 valx(1) valy(1) [valz(1)]
2 valx(2) valy(2) [valz(2)]
...
N valx(N) valy(N) [valz(N)]
```

logarithmique diametre *float* **u_tau** *float* **visco_cin** *float* **direction** *int*: To specify the imposed field (in this case, velocity) by an analytical logarithmic law of the wall:
g(x,y,z) = u_tau * (log(0.5*diametre*u_tau/visco_cin)/Kappa + 5.1)
with g(x,y,z)=u(x,y,z) if **direction** is set to 1 (g=v(x,y,z) if **direction** is set to 2, and g=w(w,y,z) if it is set to 3)

- **moyenne_recylee** *methode_recyc*: Method used to perform a spatial or a temporal averaging of f field to specify <f>. <f> can be the surface mean of f on the plane (surface option, see below) or it can be read from several files (for example generated by the chmoy_faceperio option of the Traitement_particulier keyword to obtain a temporal mean field). The option *methode_recyc* can be:

  surfacique: Surface mean for <f> from f values on the plane

  Or one of the following *methode_moy* options applied to read a temporal mean field <f>(x,y,z):

  interpolation

  connexion_approchee

  connexion_exacte

See also: champ_front_base (15.1)

Usage:
champ_front_recyclage bloc
where

- **bloc** *str*

15.23 Champ_front_tabule

Description: Constant field on the boundary, tabulated as a function of time.

See also: champ_front_base (15.1)

Usage:
champ_front_tabule nb_comp bloc
where

- **nb_comp** *int*: Number of field components.
- **bloc** *bloc_lecture* (3.17): {nt1 t2 t3 ....tn u1 [v1 w1 ...] u2 [v2 w2 ...] u3 [v3 w3 ...] ... un [vn wn ...] }
  Values are entered into a table based on n couples (ti, ui) if nb_comp value is 1. The value of a field at a given time is calculated by linear interpolation from this table.

15.24 Champ_front_tangentiel_vef

Description: Field to define the tangential velocity vector field standard at the boundary in VEF discretization.

See also: champ_front_base (15.1)

Usage:
champ_front_tangentiel_vef mot vit_tan
where

- **mot** *str into ['vitesse_tangentielle']*: Name of vector field.
- **vit_tan** *float*: Vector field standard [m/s].

15.25 Champ_front_uniforme

Description: Boundary field which is constant in space and stationary.

See also: champ_front_base (15.1)

Usage:
champ_front_uniforme val
where

- **val** *n x1 x2 ... xn*: Values of field components.

15.26 Champ_front_xyz_debit

Description: This field is used to define a flow rate field with a velocity profil which will be normalized to match the flow rate chosen.

See also: champ_front_base (15.1)

Usage:
champ_front_xyz_debit *str*
Read *str* {

[**velocity_profil** *champ_front_base*]
flow_rate *champ_front_base*

}
where

- **velocity_profil** *champ_front_base* (15.1): velocity_profil 0 velocity field to define the profil of velocity.
- **flow_rate** *champ_front_base* (15.1): flow_rate 1 uniform field in space to define the flow rate. It could be, for example, champ_front_uniforme, ch_front_input_uniform or champ_front_fonc_t

16 interpolation_ibm_base

Description: Base class for all the interpolation methods available in the Immersed Boundary Method (IBM).

See also: objet_u (34) ibm_element_fluide (16.2) ibm_aucune (16.1) ibm_gradient_moyen (16.4)

Usage:
interpolation_ibm_base

16.1 Ibm_aucune

Synonymous: **interpolation_ibm_aucune**

Description: Immersed Boundary Method (IBM): no interpolation.

See also: interpolation_ibm_base (16)

Usage:
ibm_aucune

16.2 Ibm_element_fluide

Synonymous: **interpolation_ibm_element_fluide**

Description: Immersed Boundary Method (IBM): fluid element interpolation.

See also: interpolation_ibm_base (16) ibm_hybride (16.3)

Usage:
ibm_element_fluide *str*
Read *str* {

points_fluides *champ_base*
points_solides *champ_base*
elements_fluides *champ_base*
correspondance_elements *champ_base*

}
where

- **points_fluides** *champ_base* (14.1): Node field giving the projection of the point below (points-_solides) falling into the pure cell fluid
- **points_solides** *champ_base* (14.1): Node field giving the projection of the node on the immersed boundary
- **elements_fluides** *champ_base* (14.1): Node field giving the number of the element (cell) containing the pure fluid point
- **correspondance_elements** *champ_base* (14.1): Cell field giving the SALOME cell number

16.3 Ibm_hybride

Synonymous: **interpolation_ibm_hybride**

Description: Immersed Boundary Method (IBM): hybrid (fluid/mean gradient) interpolation.

See also: ibm_element_fluide (16.2)

Usage:
ibm_hybride *str*
Read *str* {

est_dirichlet *champ_base*
elements_solides *champ_base*
points_fluides *champ_base*
points_solides *champ_base*
elements_fluides *champ_base*
correspondance_elements *champ_base*

}
where

- **est_dirichlet** *champ_base* (14.1): Node field of booleans indicating whether the node belong to an element where the interface is
- **elements_solides** *champ_base* (14.1): Node field giving the element number containing the solid point
- **points_fluides** *champ_base* (14.1) for inheritance: Node field giving the projection of the point below (points_solides) falling into the pure cell fluid
- **points_solides** *champ_base* (14.1) for inheritance: Node field giving the projection of the node on the immersed boundary
- **elements_fluides** *champ_base* (14.1) for inheritance: Node field giving the number of the element (cell) containing the pure fluid point
- **correspondance_elements** *champ_base* (14.1) for inheritance: Cell field giving the SALOME cell number

16.4 Ibm_gradient_moyen

Synonymous: **interpolation_ibm_gradient_moyen**

Description: Immersed Boundary Method (IBM): mean gradient interpolation.

See also: interpolation_ibm_base (16)

Usage:
ibm_gradient_moyen *str*
Read *str* {

> **points_solides** *champ_base*
> **est_dirichlet** *champ_base*
> **correspondance_elements** *champ_base*
> **elements_solides** *champ_base*

}
where

- **points_solides** *champ_base* (14.1): Node field giving the projection of the node on the immersed boundary
- **est_dirichlet** *champ_base* (14.1): Node field of booleans indicating whether the node belong to an element where the interface is
- **correspondance_elements** *champ_base* (14.1): Cell field giving the SALOME cell number
- **elements_solides** *champ_base* (14.1): Node field giving the element number containing the solid point

17 loi_etat_base

Description: Basic class for state laws used with a dilatable fluid.

See also: objet_u (34) loi_etat_gaz_reel_base (17.4) loi_etat_gaz_parfait_base (17.3)

Usage:

17.1 Binaire_gaz_parfait_qc

Description: Class for perfect gas binary mixtures state law used with a quasi-compressible fluid under the iso-thermal and iso-bar assumptions.

See also: loi_etat_gaz_parfait_base (17.3)

Usage:
binaire_gaz_parfait_QC *str*
Read *str* {

> **molar_mass1** *float*
> **molar_mass2** *float*
> **mu1** *float*
> **mu2** *float*
> **temperature** *float*
> **diffusion_coeff** *float*

}
where

- **molar_mass1** *float*: Molar mass of species 1 (in kg/mol).
- **molar_mass2** *float*: Molar mass of species 2 (in kg/mol).
- **mu1** *float*: Dynamic viscosity of species 1 (in kg/m.s).
- **mu2** *float*: Dynamic viscosity of species 2 (in kg/m.s).
- **temperature** *float*: Temperature (in Kelvin) which will be constant during the simulation since this state law only works for iso-thermal conditions.
- **diffusion_coeff** *float*: Diffusion coefficient assumed the same for both species (in m2/s).

17.2 Binaire_gaz_parfait_wc

Description: Class for perfect gas binary mixtures state law used with a weakly-compressible fluid under the iso-thermal and iso-bar assumptions.

See also: loi_etat_gaz_parfait_base (17.3)

Usage:
binaire_gaz_parfait_WC *str*
Read *str* {

> **molar_mass1** *float*
> **molar_mass2** *float*
> **mu1** *float*
> **mu2** *float*
> **temperature** *float*
> **diffusion_coeff** *float*

}
where

- **molar_mass1** *float*: Molar mass of species 1 (in kg/mol).
- **molar_mass2** *float*: Molar mass of species 2 (in kg/mol).
- **mu1** *float*: Dynamic viscosity of species 1 (in kg/m.s).
- **mu2** *float*: Dynamic viscosity of species 2 (in kg/m.s).
- **temperature** *float*: Temperature (in Kelvin) which will be constant during the simulation since this state law only works for iso-thermal conditions.
- **diffusion_coeff** *float*: Diffusion coefficient assumed the same for both species (in m2/s).

17.3 Loi_etat_gaz_parfait_base

Description: Basic class for perfect gases state laws used with a dilatable fluid.

See also: loi_etat_base (17) rhoT_gaz_parfait_QC (17.9) binaire_gaz_parfait_QC (17.1) multi_gaz_parfait-_QC (17.5) gaz_parfait_QC (17.7) multi_gaz_parfait_WC (17.6) binaire_gaz_parfait_WC (17.2) gaz_parfait-_WC (17.8)

Usage:

17.4 Loi_etat_gaz_reel_base

Description: Basic class for real gases state laws used with a dilatable fluid.

See also: loi_etat_base (17) rhoT_gaz_reel_QC (17.10)

Usage:

17.5 Multi_gaz_parfait_qc

Description: Class for perfect gas multi-species mixtures state law used with a quasi-compressible fluid.

See also: loi_etat_gaz_parfait_base (17.3)

Usage:
multi_gaz_parfait_QC *str*
Read *str* {

> **sc** *float*
> **prandtl** *float*
> [**cp** *float*]
> [**dtol_fraction** *float*]
> [**correction_fraction**]
> [**ignore_check_fraction**]

}
where

- **sc** *float*: Schmidt number of the gas Sc=nu/D (D: diffusion coefficient of the mixing).
- **prandtl** *float*: Prandtl number of the gas Pr=mu*Cp/lambda
- **cp** *float*: Specific heat at constant pressure of the gas Cp.
- **dtol_fraction** *float*: Delta tolerance on mass fractions for check testing (default value 1.e-6).
- **correction_fraction** : To force mass fractions between 0. and 1.
- **ignore_check_fraction** : Not to check if mass fractions between 0. and 1.

17.6 Multi_gaz_parfait_wc

Description: Class for perfect gas multi-species mixtures state law used with a weakly-compressible fluid.

See also: loi_etat_gaz_parfait_base (17.3)

Usage:
multi_gaz_parfait_WC *str*
Read *str* {

> **species_number** *int*
> **diffusion_coeff** *champ_base*
> **molar_mass** *champ_base*
> **mu** *champ_base*
> **cp** *champ_base*
> **prandtl** *float*

}
where

- **species_number** *int*: Number of species you are considering in your problem.
- **diffusion_coeff** *champ_base* (14.1): Diffusion coefficient of each species, defined with a Champ-_uniforme of dimension equals to the species_number.
- **molar_mass** *champ_base* (14.1): Molar mass of each species, defined with a Champ_uniforme of dimension equals to the species_number.
- **mu** *champ_base* (14.1): Dynamic viscosity of each species, defined with a Champ_uniforme of dimension equals to the species_number.
- **cp** *champ_base* (14.1): Specific heat at constant pressure of the gas Cp, defined with a Champ-_uniforme of dimension equals to the species_number..
- **prandtl** *float*: Prandtl number of the gas Pr=mu*Cp/lambda.

17.7 Gaz_parfait_qc

Description: Class for perfect gas state law used with a quasi-compressible fluid.

See also: loi_etat_gaz_parfait_base (17.3)

Usage:
gaz_parfait_QC *str*
Read *str* {

> **Cp** *float*
> [**Cv** *float*]
> [**gamma** *float*]
> **Prandtl** *float*
> [**rho_constant_pour_debug** *champ_base*]

}
where

- **Cp** *float*: Specific heat at constant pressure (J/kg/K).
- **Cv** *float*: Specific heat at constant volume (J/kg/K).
- **gamma** *float*: Cp/Cv
- **Prandtl** *float*: Prandtl number of the gas Pr=mu*Cp/lambda
- **rho_constant_pour_debug** *champ_base* (14.1): For developers to debug the code with a constant rho.

17.8 Gaz_parfait_wc

Description: Class for perfect gas state law used with a weakly-compressible fluid.

See also: loi_etat_gaz_parfait_base (17.3)

Usage:
gaz_parfait_WC *str*
Read *str* {

> **Cp** *float*
> [**Cv** *float*]
> [**gamma** *float*]
> **Prandtl** *float*

}
where

- **Cp** *float*: Specific heat at constant pressure (J/kg/K).
- **Cv** *float*: Specific heat at constant volume (J/kg/K).
- **gamma** *float*: Cp/Cv
- **Prandtl** *float*: Prandtl number of the gas Pr=mu*Cp/lambda

17.9 Rhot_gaz_parfait_qc

Description: Class for perfect gas used with a aquasi-compressible fluid where the state equation is defined as rho = f(T).

See also: loi_etat_gaz_parfait_base (17.3)

Usage:
rhoT_gaz_parfait_QC *str*
Read *str* {

> **cp** *float*
> [**prandtl** *float*]
> [**rho_xyz** *champ_base*]
> [**rho_t** *str*]

}
where

- **cp** *float*: Specific heat at constant pressure of the gas Cp.
- **prandtl** *float*: Prandtl number of the gas Pr=mu*Cp/lambda
- **rho_xyz** *champ_base* (14.1): Defined with a Champ_Fonc_xyz to define a constant rho with time (space dependent)
- **rho_t** *str*: Expression of T used to calculate rho. This can lead to a variable rho, both in space and in time.

17.10 Rhot_gaz_reel_qc

Description: Class for real gas state law used with a quasi-compressible fluid.

See also: loi_etat_gaz_reel_base (17.4)

Usage:
rhoT_gaz_reel_QC **bloc**
where

- **bloc** *bloc_lecture* (3.17): Description.

18 loi_fermeture_base

Description: Class for appends fermeture to problem

Keyword Discretize should have already been used to read the object.
See also: objet_u (34) loi_fermeture_test (18.1)

Usage:

18.1 Loi_fermeture_test

Description: Loi for test only

Keyword Discretize should have already been used to read the object.
See also: loi_fermeture_base (18)

Usage:
loi_fermeture_test *str*
Read *str* {

> [**coef** *float*]

}
where

- **coef** *float*: coefficient

19 loi_horaire

Description: to define the movement with a time-dependant law for the solid interface.

See also: objet_u (34)

Usage:
loi_horaire *str*
Read *str* {

> **position** *n word1 word2 ... wordn*
> **vitesse** *n word1 word2 ... wordn*
> [**rotation** *n word1 word2 ... wordn*]
> [**derivee_rotation** *n word1 word2 ... wordn*]

}
where

- **position** *n word1 word2 ... wordn*
- **vitesse** *n word1 word2 ... wordn*
- **rotation** *n word1 word2 ... wordn*
- **derivee_rotation** *n word1 word2 ... wordn*

20 milieu_base

Description: Basic class for medium (physics properties of medium).

See also: objet_u (34) Solide (20.3) fluide_base (20.6) constituant (20.5)

Usage:

20.1 Fluide_sodium_gaz

Description: Class for Fluide_sodium_liquide

See also: fluide_reel_base (20.12)

Usage:
Fluide_sodium_gaz *str*
Read *str* {

> [**P_ref** *float*]
> [**T_ref** *float*]
> [**indice** *champ_base*]
> [**kappa** *champ_base*]

}
where

- **P_ref** *float*: Use to set the pressure value in the closure law. If not specified, the value of the pressure unknown will be used
- **T_ref** *float*: Use to set the temperature value in the closure law. If not specified, the value of the temperature unknown will be used
- **indice** *champ_base* (14.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (14.1) for inheritance: Absorptivity of fluid (m-1).

20.2 Fluide_sodium_liquide

Description: Class for Fluide_sodium_liquide

See also: fluide_reel_base (20.12)

Usage:
Fluide_sodium_liquide *str*
Read *str* {

> [**P_ref** *float*]
> [**T_ref** *float*]
> [**indice** *champ_base*]
> [**kappa** *champ_base*]

}
where

- **P_ref** *float*: Use to set the pressure value in the closure law. If not specified, the value of the pressure unknown will be used
- **T_ref** *float*: Use to set the temperature value in the closure law. If not specified, the value of the temperature unknown will be used
- **indice** *champ_base* (14.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (14.1) for inheritance: Absorptivity of fluid (m-1).

20.3 Solide

Description: Solid with cp and/or rho non-uniform.

See also: milieu_base (20)

Usage:
Solide *str*
Read *str* {

> [**rho** *champ_base*]
> [**cp** *champ_base*]
> [**lambda** *champ_base*]

}
where

- **rho** *champ_base* (14.1): Density (kg.m-3).
- **cp** *champ_base* (14.1): Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (14.1): Conductivity (W.m-1.K-1).

20.4 Stiffenedgas

Description: Class for Stiffened Gas

See also: fluide_reel_base (20.12)

Usage:
StiffenedGas *str*
Read *str* {

[**gamma** *float*]
[**pinf** *float*]
[**mu** *float*]
[**lambda** *float*]
[**indice** *champ_base*]
[**kappa** *champ_base*]

}
where

- **gamma** *float*: Heat capacity ratio (Cp/Cv)
- **pinf** *float*: Stiffened gas pressure constant (if set to zero, the state law becomes identical to that of perfect gases)
- **mu** *float*: Dynamic viscosity
- **lambda** *float*: Thermal conductivity
- **indice** *champ_base* (14.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (14.1) for inheritance: Absorptivity of fluid (m-1).

20.5 Constituant

Description: Constituent.

See also: milieu_base (20)

Usage:
constituant *str*
Read *str* {

[**rho** *champ_base*]
[**cp** *champ_base*]
[**lambda** *champ_base*]
[**coefficient_diffusion** *champ_base*]

}
where

- **rho** *champ_base* (14.1): Density (kg.m-3).
- **cp** *champ_base* (14.1): Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (14.1): Conductivity (W.m-1.K-1).
- **coefficient_diffusion** *champ_base* (14.1): Constituent diffusion coefficient value (m2.s-1). If a multi-constituent problem is being processed, the diffusivite will be a vectorial and each components will be the diffusion of the constituent.

20.6 Fluide_base

Description: Basic class for fluids.

See also: milieu_base (20) fluide_reel_base (20.12) fluide_dilatable_base (20.7) fluide_incompressible (20.8)

Usage:
fluide_base *str*
Read *str* {

[**indice** *champ_base*]
[**kappa** *champ_base*]

}
where

- **indice** *champ_base* (14.1): Refractivity of fluid.
- **kappa** *champ_base* (14.1): Absorptivity of fluid (m-1).

20.7 Fluide_dilatable_base

Description: Basic class for dilatable fluids.

See also: fluide_base (20.6) fluide_quasi_compressible (20.10) fluide_weakly_compressible (20.13)

Usage:
fluide_dilatable_base *str*
Read *str* {

[**indice** *champ_base*]
[**kappa** *champ_base*]

}
where

- **indice** *champ_base* (14.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (14.1) for inheritance: Absorptivity of fluid (m-1).

20.8 Fluide_incompressible

Description: Class for non-compressible fluids.

See also: fluide_base (20.6) fluide_ostwald (20.9)

Usage:
fluide_incompressible *str*
Read *str* {

[**beta_th** *champ_base*]
[**mu** *champ_base*]
[**beta_co** *champ_base*]
[**rho** *champ_base*]
[**cp** *champ_base*]
[**lambda** *champ_base*]

[**indice** *champ_base*]
[**kappa** *champ_base*]

}
where

- **beta_th** *champ_base* (14.1): Thermal expansion (K-1).
- **mu** *champ_base* (14.1): Dynamic viscosity (kg.m-1.s-1).
- **beta_co** *champ_base* (14.1): Volume expansion coefficient values in concentration.
- **rho** *champ_base* (14.1): Density (kg.m-3).
- **cp** *champ_base* (14.1): Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (14.1): Conductivity (W.m-1.K-1).
- **indice** *champ_base* (14.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (14.1) for inheritance: Absorptivity of fluid (m-1).

20.9 Fluide_ostwald

Description: Non-Newtonian fluids governed by Ostwald's law. The law applicable to stress tensor is:
tau=K(T)*(D:D/2)**((n-1)/2)*D Where:
D refers to the deformation tensor
K refers to fluid consistency (may be a function of the temperature T)
n refers to the fluid structure index n=1 for a Newtonian fluid, n<1 for a rheofluidifier fluid, n>1 for a rheothickening fluid.

See also: fluide_incompressible (20.8)

Usage:
fluide_ostwald *str*
Read *str* {

[**k** *champ_base*]
[**n** *champ_base*]
[**beta_th** *champ_base*]
[**mu** *champ_base*]
[**beta_co** *champ_base*]
[**rho** *champ_base*]
[**cp** *champ_base*]
[**lambda** *champ_base*]
[**indice** *champ_base*]
[**kappa** *champ_base*]

}
where

- **k** *champ_base* (14.1): Fluid consistency.
- **n** *champ_base* (14.1): Fluid structure index.
- **beta_th** *champ_base* (14.1) for inheritance: Thermal expansion (K-1).
- **mu** *champ_base* (14.1) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **beta_co** *champ_base* (14.1) for inheritance: Volume expansion coefficient values in concentration.
- **rho** *champ_base* (14.1) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (14.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (14.1) for inheritance: Conductivity (W.m-1.K-1).
- **indice** *champ_base* (14.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (14.1) for inheritance: Absorptivity of fluid (m-1).

20.10 Fluide_quasi_compressible

Description: Quasi-compressible flow with a low mach number assumption; this means that the thermo-dynamic pressure (used in state law) is uniform in space.

See also: fluide_dilatable_base (20.7)

Usage:
fluide_quasi_compressible *str*
Read *str* {

[**sutherland** *bloc_sutherland*]
[**pression** *float*]
[**loi_etat** *loi_etat_base*]
[**traitement_pth** *str into ['edo', 'constant', 'conservation_masse']*]
[**traitement_rho_gravite** *str into ['standard', 'moins_rho_moyen']*]
[**temps_debut_prise_en_compte_drho_dt** *float*]
[**omega_relaxation_drho_dt** *float*]
[**lambda** *champ_base*]
[**mu** *champ_base*]
[**indice** *champ_base*]
[**kappa** *champ_base*]

}
where

- **sutherland** *bloc_sutherland* (20.11): Sutherland law for viscosity and for conductivity.
- **pression** *float*: Initial thermo-dynamic pressure used in the assosciated state law.
- **loi_etat** *loi_etat_base* (17): The state law that will be associated to the Quasi-compressible fluid.
- **traitement_pth** *str into ['edo', 'constant', 'conservation_masse']*: Particular treatment for the thermodynamic pressure Pth ; there are three possibilities:
  1) with the keyword 'edo' the code computes Pth solving an O.D.E. ; in this case, the mass is not strictly conserved (it is the default case for quasi compressible computation):
  2) the keyword 'conservation_masse' forces the conservation of the mass (closed geometry or with periodic boundaries condition)
  3) the keyword 'constant' makes it possible to have a constant Pth ; it's the good choice when the flow is open (e.g. with pressure boundary conditions).
  It is possible to monitor the volume averaged value for temperature and density, plus Pth evolution in the .evol_glob file.
- **traitement_rho_gravite** *str into ['standard', 'moins_rho_moyen']*: It may be :1) s̀tandard̀: the gravity term is evaluted with rho*g (It is the default). 2) m̀oins_rho_moyeǹ: the gravity term is evaluated with (rho-rhomoy) *g. Unknown pressure is then P*=P+rhomoy*g*z. It is useful when you apply uniforme pressure boundary condition like P*=0.
- **temps_debut_prise_en_compte_drho_dt** *float*: While time<value, dRho/dt is set to zero (Rho, volumic mass). Useful for some calculation during the first time steps with big variation of temperature and volumic mass.
- **omega_relaxation_drho_dt** *float*: Optional option to have a relaxed algorithm to solve the mass equation. value is used (1 per default) to specify omega.
- **lambda** *champ_base* (14.1): Conductivity (W.m-1.K-1).
- **mu** *champ_base* (14.1): Dynamic viscosity (kg.m-1.s-1).
- **indice** *champ_base* (14.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (14.1) for inheritance: Absorptivity of fluid (m-1).

20.11 Bloc_sutherland

Description: Sutherland law for viscosity mu(T)=mu0*((T0+C)/(T+C))*(T/T0)**1.5 and (optional) for conductivity lambda(T)=mu0*Cp/Prandtl*((T0+Slambda)/(T+Slambda))*(T/T0)**1.5

See also: objet_lecture (33)

Usage:
problem_name mu0 mu0_val t0 t0_val [**Slambda**] [**s**] **C c_val**
where

- **problem_name** *str*: Name of problem.
- **mu0** *str into ['mu0']*
- **mu0_val** *float*
- **t0** *str into ['T0']*
- **t0_val** *float*
- **Slambda** *str into ['Slambda']*
- **s** *float*
- **C** *str into ['C']*
- **c_val** *float*

20.12 Fluide_reel_base

Description: Class for real fluids.

See also: fluide_base (20.6) Fluide_sodium_gaz (20.1) StiffenedGas (20.4) Fluide_sodium_liquide (20.2)

Usage:
fluide_reel_base *str*
Read *str* {

[**indice** *champ_base*]
[**kappa** *champ_base*]

}
where

- **indice** *champ_base* (14.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (14.1) for inheritance: Absorptivity of fluid (m-1).

20.13 Fluide_weakly_compressible

Description: Weakly-compressible flow with a low mach number assumption; this means that the thermo-dynamic pressure (used in state law) can vary in space.

See also: fluide_dilatable_base (20.7)

Usage:
fluide_weakly_compressible *str*
Read *str* {

[**loi_etat** *loi_etat_base*]
[**sutherland** *bloc_sutherland*]
[**traitement_pth** *str into ['constant']*]

[**lambda** *champ_base*]
[**mu** *champ_base*]
[**pression_thermo** *float*]
[**pression_xyz** *champ_base*]
[**use_total_pressure** *int*]
[**use_hydrostatic_pressure** *int*]
[**use_grad_pression_eos** *int*]
[**time_activate_ptot** *float*]
[**indice** *champ_base*]
[**kappa** *champ_base*]

}
where

- **loi_etat** *loi_etat_base* (17): The state law that will be associated to the Weakly-compressible fluid.
- **sutherland** *bloc_sutherland* (20.11): Sutherland law for viscosity and for conductivity.
- **traitement_pth** *str into ['constant']*: Particular treatment for the thermodynamic pressure Pth ; there is currently one possibility:
  1) the keyword 'constant' makes it possible to have a constant Pth but not uniform in space ; it's the good choice when the flow is open (e.g. with pressure boundary conditions).
- **lambda** *champ_base* (14.1): Conductivity (W.m-1.K-1).
- **mu** *champ_base* (14.1): Dynamic viscosity (kg.m-1.s-1).
- **pression_thermo** *float*: Initial thermo-dynamic pressure used in the assosciated state law.
- **pression_xyz** *champ_base* (14.1): Initial thermo-dynamic pressure used in the assosciated state law. It should be defined with as a Champ_Fonc_xyz.
- **use_total_pressure** *int*: Flag (0 or 1) used to activate and use the total pressure in the assosciated state law. The default value of this Flag is 0.
- **use_hydrostatic_pressure** *int*: Flag (0 or 1) used to activate and use the hydro-static pressure in the assosciated state law. The default value of this Flag is 0.
- **use_grad_pression_eos** *int*: Flag (0 or 1) used to specify whether or not the gradient of the thermo-dynamic pressure will be taken into account in the source term of the temperature equation (case of a non-uniform pressure). The default value of this Flag is 1 which means that the gradient is used in the source.
- **time_activate_ptot** *float*: Time (in seconds) at which the total pressure will be used in the assosciated state law.
- **indice** *champ_base* (14.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (14.1) for inheritance: Absorptivity of fluid (m-1).

21 modele_turbulence_scal_base

Description: Basic class for turbulence model for energy equation.

See also: objet_u (34)

Usage:
modele_turbulence_scal_base *str*
Read *str* {

turbulence_paroi *turbulence_paroi_scalaire_base*
[**dt_impr_nusselt** *float*]

}
where

- **turbulence_paroi** *turbulence_paroi_scalaire_base* (31): Keyword to set the wall law.

- **dt_impr_nusselt** *float*: Keyword to print local values of Nusselt number and temperature near a wall during a turbulent calculation. The values will be printed in the _Nusselt.face file each dt-_impr_nusselt time period. The local Nusselt expression is as follows : Nu = ((lambda+lambda-_t)/lambda)*d_wall/d_eq where d_wall is the distance from the first mesh to the wall and d_eq is given by the wall law. This option also gives the value of d_eq and h = (lambda+lambda_t)/d_eq and the fluid temperature of the first mesh near the wall.
  For the Neumann boundary conditions (flux_impose), the «equivalent» wall temperature given by the wall law is also printed (Tparoi equiv.) preceded for VEF calculation by the edge temperature «T face de bord».

22 nom

Description: Class to name the TRUST objects.

See also: objet_u (34) nom_anonyme (22.1)

Usage:
nom [**mot**]
where

- **mot** *str*: Chain of characters.

22.1 Nom_anonyme

Description: not_set

See also: nom (22)

Usage:
[**mot**]
where

- **mot** *str*: Chain of characters.

23 partitionneur_deriv

Description: not_set

See also: objet_u (34) metis (23.2) sous_zones (23.5) tranche (23.6) partition (23.3) fichier_decoupage (23.1) sous_domaine (23.4) union (23.7)

Usage:
partitionneur_deriv *str*
Read *str* {

[**nb_parts** *int*]

}
where

- **nb_parts** *int*: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

23.1 Fichier_decoupage

Description: This algorithm reads an array of integer values on the disc, one value for each mesh element. Each value is interpreted as the target part number n>=0 for this element. The number of parts created is the highest value in the array plus one. Empty parts can be created if some values are not present in the array.
The file format is ASCII, and contains space, tab or carriage-return separated integer values. The first value is the number nb_elem of elements in the domain, followed by nb_elem integer values (positive or zero).
This algorithm has been designed to work together with the 'ecrire_decoupage' option. You can generate a partition with any other algorithm, write it to disc, modify it, and read it again to generate the .Zone files.
Contrary to other partitioning algorithms, no correction is applied by default to the partition (eg. element 0 on processor 0 and corrections for periodic boundaries). If 'corriger_partition' is specified, these corrections are applied.

See also: partitionneur_deriv (23)

Usage:
fichier_decoupage *str*
Read *str* {

> **fichier** *str*
> [**corriger_partition**]
> [**nb_parts** *int*]

}
where

- **fichier** *str*: FILENAME
- **corriger_partition**
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

23.2 Metis

Description: Metis is an external partitionning library. It is a general algorithm that will generate a partition of the domain.

See also: partitionneur_deriv (23)

Usage:
metis *str*
Read *str* {

> [**kmetis**]
> [**use_weights**]
> [**nb_parts** *int*]

}
where

- **kmetis** : The default values are pmetis, default parameters are automatically chosen by Metis. 'kmetis' is faster than pmetis option but the last option produces better partitioning quality. In both cases, the partitioning quality may be slightly improved by increasing the nb_essais option (by default N=1). It will compute N partitions and will keep the best one (smallest edge cut number). But this option is CPU expensive, taking N=10 will multiply the CPU cost of partitioning by 10.
Experiments show that only marginal improvements can be obtained with non default parameters.

- **use_weights** : If use_weights is specified, weighting of the element-element links in the graph is used to force metis to keep opposite periodic elements on the same processor. This option can slightly improve the partitionning quality but it consumes more memory and takes more time. It is not mandatory since a correction algorithm is always applied afterwards to ensure a correct partitionning for periodic boundaries.
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

23.3 Partition

Synonymous: **decouper**

Description: This algorithm re-use the partition of the domain named DOMAINE_NAME. It is useful to partition for example a post processing domain. The partition should match with the calculation domain.

See also: partitionneur_deriv (23)

Usage:
partition *str*
Read *str* {

> **domaine** *str*
> [**nb_parts** *int*]

}
where

- **domaine** *str*: domain name
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

23.4 Sous_domaine

Description: Given a global partition of a global domain, 'sous-domaine' allows to produce a conform partition of a sub-domain generated from the bigger one using the keyword create_domain_from_sous_zone. The sub-domain will be partitionned in a conform fashion with the global domain.

See also: partitionneur_deriv (23)

Usage:
sous_domaine *str*
Read *str* {

> **fichier** *str*
> **fichier_ssz** *str*
> [**nb_parts** *int*]

}
where

- **fichier** *str*: fichier domaine
- **fichier_ssz** *str*: fichier sous zonne
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

23.5 Sous_zones

Description: This algorithm will create one part for each specified subzone/domain. All elements contained in the first subzone/domain are put in the first part, all remaining elements contained in the second subzone/domain in the second part, etc...
If all elements of the current domain are contained in the specified subzones/domain, then N parts are created, otherwise, a supplemental part is created with the remaining elements.
If no subzone is specified, all subzones defined in the domain are used to split the mesh.

See also: partitionneur_deriv (23)

Usage:
sous_zones *str*
Read *str* {

[**sous_zones** *n word1 word2 ... wordn*]
[**domaines** *n word1 word2 ... wordn*]
[**nb_parts** *int*]

}
where

- **sous_zones** *n word1 word2 ... wordn*: N SUBZONE_NAME_1 SUBZONE_NAME_2 ...
- **domaines** *n word1 word2 ... wordn*: N DOMAIN_NAME_1 DOMAIN_NAME_2 ...
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

23.6 Tranche

Description: This algorithm will create a geometrical partitionning by slicing the mesh in the two or three axis directions, based on the geometric center of each mesh element. nz must be given if dimension=3. Each slice contains the same number of elements (slices don't have the same geometrical width, and for VDF meshes, slice boundaries are generally not flat except if the number of mesh elements in each direction is an exact multiple of the number of slices). First, nx slices in the X direction are created, then each slice is split in ny slices in the Y direction, and finally, each part is split in nz slices in the Z direction. The resulting number of parts is nx*ny*nz. If one particular direction has been declared periodic, the default slicing (0, 1, 2, ..., n-1)is replaced by (0, 1, 2, ... n-1, 0), each of the two '0' slices having twice less elements than the other slices.

See also: partitionneur_deriv (23)

Usage:
tranche *str*
Read *str* {

[**tranches** *n1 n2 (n3)*]
[**nb_parts** *int*]

}
where

- **tranches** *n1 n2 (n3)*: Partitioned by nx in the X direction, ny in the Y direction, nz in the Z direction. Works only for structured meshes. No warranty for unstructured meshes.
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

23.7 Union

Description: Let several local domains be generated from a bigger one using the keyword create_domain_from_sous_zone, and let their partitions be generated in the usual way. Provided the list of partition files for each small domain, the keyword 'union' will partition the global domain in a conform fashion with the smaller domains.

See also: partitionneur_deriv (23)

Usage:
union **liste** [**nb_parts**]
where

- **liste** *bloc_lecture* (3.17): List of the partition files with the following syntaxe: {sous_zone1 decoupage1 ... sous_zoneim decoupageim } where sous_zone1 ... sous_zomeim are small domains names and decoupage1 ... decoupageim are partition files.
- **nb_parts** *int*: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

24 precond_base

Description: Basic class for preconditioning.

See also: objet_u (34) ssor (24.3) ssor_bloc (24.4) precondsolv (24.2) ilu (24.1)

Usage:

24.1 Ilu

Description: This preconditionner can be only used with the generic GEN solver.

See also: precond_base (24)

Usage:
ilu *str*
Read *str* {

  [**type** *int*]
  [**filling** *int*]

}
where

- **type** *int*: values can be 0|1|2|3 for null|left|right|left-and-right preconditionning (default value = 2)
- **filling** *int*: default value = 1.

24.2 Precondsolv

Description: not_set

See also: precond_base (24)

Usage:
precondsolv **solveur**
where

- **solveur** *solveur_sys_base* (9.13): Solver type.

24.3 Ssor

Description: Symmetric successive over-relaxation algorithm.

See also: precond_base (24)

Usage:
ssor *str*
Read *str* {

[**omega** *float*]

}
where

- **omega** *float*: Over-relaxation facteur (between 1 and 2, default value 1.6).

24.4 Ssor_bloc

Description: not_set

See also: precond_base (24)

Usage:
ssor_bloc *str*
Read *str* {

[**alpha_0** *float*]
[**precond0** *precond_base*]
[**alpha_1** *float*]
[**precond1** *precond_base*]
[**alpha_a** *float*]
[**preconda** *precond_base*]

}
where

- **alpha_0** *float*
- **precond0** *precond_base* (24)
- **alpha_1** *float*
- **precond1** *precond_base* (24)
- **alpha_a** *float*
- **preconda** *precond_base* (24)

25 saturation_base

Description: Basic class for phase change management (used in pb_multiphase)

See also: objet_u (34) saturation_sodium (25.2) saturation_constant (25.1)

Usage:

25.1 Saturation_constant

Description: Class for saturation constant

See also: saturation_base (25)

Usage:
saturation_constant *str*
Read *str* {

[**P_sat** *float*]
[**T_sat** *float*]
[**Lvap** *float*]
[**Hlsat** *float*]
[**Hvsat** *float*]

}
where

- **P_sat** *float*: Define the saturation pressure value (this is a required parameter)
- **T_sat** *float*: Define the saturation temperature value (this is a required parameter)
- **Lvap** *float*: Latent heat of vaporization
- **Hlsat** *float*: Liquid saturation enthalpy
- **Hvsat** *float*: Vapor saturation enthalpy

25.2 Saturation_sodium

Description: Class for saturation sodium

See also: saturation_base (25)

Usage:
saturation_sodium *str*
Read *str* {

[**P_ref** *float*]
[**T_ref** *float*]

}
where

- **P_ref** *float*: Use to fix the pressure value in the closure law. If not specified, the value of the pressure unknown will be used
- **T_ref** *float*: Use to fix the temperature value in the closure law. If not specified, the value of the temperature unknown will be used

26 schema_temps_base

Description: Basic class for time schemes. This scheme will be associated with a problem and the equations of this problem.

See also: objet_u (34) scheme_euler_explicit (26.3) schema_predictor_corrector (26.16) Sch_CN_iteratif (26.2) runge_kutta_ordre_3 (26.5) runge_kutta_ordre_4_d3p (26.6) leap_frog (26.4) runge_kutta_rationnel_ordre_2 (26.7) schema_implicite_base (26.15) schema_adams_bashforth_order_2 (26.8) schema_adams_bashforth_order_3 (26.9)

Usage:
schema_temps_base *str*
Read *str* {

[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **tinit** *float*: Value of initial calculation time (0 by default).
- **tmax** *float*: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float*: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float*: Minimum calculation time step (1e-16s by default).
- **dt_max** *str*: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float*: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float*: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float*: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float*: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int*

- **diffusion_implicite** *int*: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float*: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int*: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int*
- **no_conv_subiteration_diffusion_implicite** *int*
- **dt_start** *dt_start* (9.6): dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int*: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int*: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int*: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float*: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** : To disable the check of the available amount of disk space during the calculation.
- **disable_progress** : To disable the writing of the .progress file.
- **disable_dt_ev** : To disable the writing of the .dt_ev file.
- **gnuplot_header** *int*: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.1 Sch_cn_ex_iteratif

Description: This keyword also describes a Crank-Nicholson method of second order accuracy but here, for scalars, because of instablities encountered when dt>dt_CFL, the Crank Nicholson scheme is not applied to scalar quantities. Scalars are treated according to Euler-Explicite scheme at the end of the CN treatment for velocity flow fields (by doing p Euler explicite under-iterations at dt<=dt_CFL). Parameters are the sames (but default values may change) compare to the Sch_CN_iterative scheme plus a relaxation keyword: niter_min (2 by default), niter_max (6 by default), niter_avg (3 by default), facsec_max (20 by default), seuil (0.05 by default)

See also: Sch_CN_iteratif (26.2)

Usage:
Sch_CN_EX_iteratif *str*
Read *str* {

  [**omega** *float*]
  [**niter_min** *int*]
  [**niter_max** *int*]
  [**niter_avg** *int*]

[**facsec_max** *float*]
[**seuil** *float*]
[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **omega** *float*: relaxation factor (0.1 by default)
- **niter_min** *int* for inheritance: minimal number of p-iterations to satisfy convergence criteria (2 by default)
- **niter_max** *int* for inheritance: number of maximum p-iterations allowed to satisfy convergence criteria (6 by default)
- **niter_avg** *int* for inheritance: threshold of p-iterations (3 by default). If the number of p-iterations is greater than niter_avg, facsec is reduced, if lesser than niter_avg, facsec is increased (but limited by the facsec_max value).
- **facsec_max** *float* for inheritance: maximum ratio allowed between dynamical time step returned by iterative process and stability time returned by CFL condition (2 by default).
- **seuil** *float* for inheritance: criteria for ending iterative process (Max(|| u(p) - u(p-1)||/Max || u(p) ||) < seuil) (0.001 by default)
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.

- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.2 Sch_cn_iteratif

Description: The Crank-Nicholson method of second order accuracy. A mid-point rule formulation is used (Euler-centered scheme). The basic scheme is:

$$u(t+1) = u(t) + du/dt(t+1/2) * dt$$

The estimation of the time derivative du/dt at the level (t+1/2) is obtained either by iterative process. The time derivative du/dt at the level (t+1/2) is calculated iteratively with a simple under-relaxations method. Since the method is implicit, neither the cfl nor the fourier stability criteria must be respected. The time step is calculated in a way that the iterative procedure converges with the less iterations as possible. Remark : for stationary or RANS calculations, no limitation can be given for time step through high value of facsec_max parameter (for instance : facsec_max 1000). In counterpart, for LES calculations, high values of facsec_max may engender numerical instabilities.

See also: schema_temps_base (26) Sch_CN_EX_iteratif (26.1)

Usage:
Sch_CN_iteratif *str*
Read *str* {

> [**niter_min** *int*]
> [**niter_max** *int*]
> [**niter_avg** *int*]
> [**facsec_max** *float*]
> [**seuil** *float*]
> [**tinit** *float*]
> [**tmax** *float*]
> [**tcpumax** *float*]
> [**dt_min** *float*]
> [**dt_max** *str*]
> [**dt_sauv** *float*]
> [**dt_impr** *float*]
> [**facsec** *float*]
> [**seuil_statio** *float*]
> [**seuil_statio_relatif_deconseille** *int*]
> [**diffusion_implicite** *int*]
> [**seuil_diffusion_implicite** *float*]
> [**impr_diffusion_implicite** *int*]
> [**no_error_if_not_converged_diffusion_implicite** *int*]
> [**no_conv_subiteration_diffusion_implicite** *int*]
> [**dt_start** *dt_start*]
> [**nb_pas_dt_max** *int*]
> [**niter_max_diffusion_implicite** *int*]
> [**precision_impr** *int*]
> [**periode_sauvegarde_securite_en_heures** *float*]
> [**no_check_disk_space**]
> [**disable_progress**]
> [**disable_dt_ev**]
> [**gnuplot_header** *int*]

}
where

- **niter_min** *int*: minimal number of p-iterations to satisfy convergence criteria (2 by default)
- **niter_max** *int*: number of maximum p-iterations allowed to satisfy convergence criteria (6 by default)
- **niter_avg** *int*: threshold of p-iterations (3 by default). If the number of p-iterations is greater than niter_avg, facsec is reduced, if lesser than niter_avg, facsec is increased (but limited by the facsec_max value).
- **facsec_max** *float*: maximum ratio allowed between dynamical time step returned by iterative process and stability time returned by CFL condition (2 by default).
- **seuil** *float*: criteria for ending iterative process (Max(|| u(p) - u(p-1)||/Max || u(p) ||) < seuil) (0.001 by default)
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).

- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows

to use the column title instead of columns number.

26.3 Scheme_euler_explicit

Synonymous: **schema_euler_explicite**

Description: This is the Euler explicit scheme.

See also: schema_temps_base (26)

Usage:
scheme_euler_explicit *str*
Read *str* {

[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.

- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.4 Leap_frog

Description: This is the leap-frog scheme.

See also: schema_temps_base (26)

Usage:
leap_frog *str*
Read *str* {

[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened

meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.

- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.5 Runge_kutta_ordre_3

Description: This is the Runge-Kutta scheme of third order.

See also: schema_temps_base (26)

Usage:
runge_kutta_ordre_3 *str*
Read *str* {

[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]

[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.

dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.

- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.6 Runge_kutta_ordre_4_d3p

Description: not_set

See also: schema_temps_base (26)

Usage:
runge_kutta_ordre_4_d3p *str*
Read *str* {

[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.7 Runge_kutta_rationnel_ordre_2

Description: This is the Runge-Kutta rational scheme of second order. The method is described in the note: Wambeck - Rational Runge-Kutta methods for solving systems of ordinary differential equations, at the link: https://link.springer.com/article/10.1007/BF02252381. Although rational methods require more computational work than linear ones, they can have some other properties, such as a stable behaviour with explicitness, which make them preferable. The CFD application of this RRK2 scheme is described in the note: https://link.springer.com/content/pdf/10.1007%2F3-540-13917-6_112.pdf.

See also: schema_temps_base (26)

Usage:
runge_kutta_rationnel_ordre_2 *str*
Read *str* {

[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).

- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.8 Schema_adams_bashforth_order_2

Description: not_set

See also: schema_temps_base (26)

Usage:
schema_adams_bashforth_order_2 *str*
Read *str* {

[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.

- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.9 Schema_adams_bashforth_order_3

Description: not_set

See also: schema_temps_base (26)

Usage:
schema_adams_bashforth_order_3 *str*
Read *str* {

[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]

[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.

- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.10 Schema_adams_moulton_order_2

Description: not_set

See also: schema_implicite_base (26.15)

Usage:
schema_adams_moulton_order_2 *str*
Read *str* {

[**facsec_max** *float*]
[**max_iter_implicite** *int*]
solveur *solveur_implicite_base*
[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]

[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.
  Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.
  Advice:
  The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:
  -Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
  -Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
  -Thermohydralic with natural convection, facsec around 300
  -Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable
  These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.
- **max_iter_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (27) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. solver is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps, and ICE (for PB_multiphase). But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
  Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not

entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).

- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.11 Schema_adams_moulton_order_3

Description: not_set

See also: schema_implicite_base (26.15)

Usage:
schema_adams_moulton_order_3 *str*
Read *str* {

> [**facsec_max** *float*]
> [**max_iter_implicite** *int*]
> **solveur** *solveur_implicite_base*
> [**tinit** *float*]
> [**tmax** *float*]
> [**tcpumax** *float*]
> [**dt_min** *float*]
> [**dt_max** *str*]
> [**dt_sauv** *float*]
> [**dt_impr** *float*]
> [**facsec** *float*]
> [**seuil_statio** *float*]
> [**seuil_statio_relatif_deconseille** *int*]
> [**diffusion_implicite** *int*]
> [**seuil_diffusion_implicite** *float*]
> [**impr_diffusion_implicite** *int*]
> [**no_error_if_not_converged_diffusion_implicite** *int*]
> [**no_conv_subiteration_diffusion_implicite** *int*]
> [**dt_start** *dt_start*]
> [**nb_pas_dt_max** *int*]
> [**niter_max_diffusion_implicite** *int*]
> [**precision_impr** *int*]
> [**periode_sauvegarde_securite_en_heures** *float*]
> [**no_check_disk_space**]
> [**disable_progress**]
> [**disable_dt_ev**]
> [**gnuplot_header** *int*]

}
where

- **facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.
  Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.
  Advice:
  The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:
  -Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
  -Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
  -Thermohydralic with natural convection, facsec around 300
  -Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable
  These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.

- **max_iter_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (27) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. solver is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps, and ICE (for PB_multiphase). But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
  Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.

dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.

- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.12 Schema_backward_differentiation_order_2

Description: not_set

See also: schema_implicite_base (26.15)

Usage:
schema_backward_differentiation_order_2 *str*
Read *str* {

[**facsec_max** *float*]
[**max_iter_implicite** *int*]
solveur *solveur_implicite_base*
[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]

[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.
Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton-_order_3 needs facsec=facsec_max=1.
Advice:
The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:
-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
-Thermohydralic with natural convection, facsec around 300
-Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable
These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.
- **max_iter_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (27) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. solver is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps, and ICE (for PB_multiphase). But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.

Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.

- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.13 Schema_backward_differentiation_order_3

Description: not_set

See also: schema_implicite_base (26.15)

Usage:
schema_backward_differentiation_order_3 *str*
Read *str* {

[**facsec_max** *float*]
[**max_iter_implicite** *int*]

solveur *solveur_implicite_base*
[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.
Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton-_order_3 needs facsec=facsec_max=1.
Advice:
The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:
-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
-Thermohydralic with natural convection, facsec around 300
-Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable
These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.
- **max_iter_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (27) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. solver is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps, and ICE (for PB_multiphase). But it may run faster because the pressure matrix is not re-assembled and thus

provides CPU gains.
Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).

- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.14 Scheme_euler_implicit

Synonymous: **schema_euler_implicite**

Description: This is the Euler implicit scheme.

See also: schema_implicite_base (26.15)

Usage:
scheme_euler_implicit *str*
Read *str* {

> [**facsec_max** *float*]
> [**thermique_monolithique** *int*]
> [**max_iter_implicite** *int*]
> **solveur** *solveur_implicite_base*
> [**tinit** *float*]
> [**tmax** *float*]
> [**tcpumax** *float*]
> [**dt_min** *float*]
> [**dt_max** *str*]
> [**dt_sauv** *float*]
> [**dt_impr** *float*]
> [**facsec** *float*]
> [**seuil_statio** *float*]
> [**seuil_statio_relatif_deconseille** *int*]
> [**diffusion_implicite** *int*]
> [**seuil_diffusion_implicite** *float*]
> [**impr_diffusion_implicite** *int*]
> [**no_error_if_not_converged_diffusion_implicite** *int*]
> [**no_conv_subiteration_diffusion_implicite** *int*]
> [**dt_start** *dt_start*]
> [**nb_pas_dt_max** *int*]
> [**niter_max_diffusion_implicite** *int*]
> [**precision_impr** *int*]
> [**periode_sauvegarde_securite_en_heures** *float*]
> [**no_check_disk_space**]
> [**disable_progress**]
> [**disable_dt_ev**]
> [**gnuplot_header** *int*]

}
where

- **facsec_max** *float*: 1 Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the im-

plicit scheme but it couldn't be higher than facsec_max value.
Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton-_order_3 needs facsec=facsec_max=1.
Advice:
The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:
-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
-Thermohydralic with natural convection, facsec around 300
-Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable
These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.

- **thermique_monolithique** *int*: Activate monolithic thermal coupling of equations for coupled problems. 0 = no, 1 = yes, 2 = yes and test convergence
- **max_iter_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (27) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. solver is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps, and ICE (for PB_multiphase). But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.

- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.15 Schema_implicite_base

Description: Basic class for implicite time scheme.

See also: schema_temps_base (26) schema_adams_moulton_order_2 (26.10) schema_adams_moulton_order_3 (26.11) schema_backward_differentiation_order_2 (26.12) schema_backward_differentiation_order_3 (26.13) scheme_euler_implicit (26.14)

Usage:
schema_implicite_base *str*
Read *str* {

[**max_iter_implicite** *int*]
solveur *solveur_implicite_base*
[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]
[**dt_min** *float*]

[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **max_iter_implicite** *int*: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (27): This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. solver is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps, and ICE (for PB_multiphase). But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
  Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-_Adams_Bashforth_order_3.

- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

26.16 Schema_predictor_corrector

Description: This is the predictor-corrector scheme (second order). It is more accurate and economic than MacCormack scheme. It gives best results with a second ordre convective scheme like quick, centre (VDF).

See also: schema_temps_base (26)

Usage:
schema_predictor_corrector *str*
Read *str* {

[**tinit** *float*]
[**tmax** *float*]
[**tcpumax** *float*]

[**dt_min** *float*]
[**dt_max** *str*]
[**dt_sauv** *float*]
[**dt_impr** *float*]
[**facsec** *float*]
[**seuil_statio** *float*]
[**seuil_statio_relatif_deconseille** *int*]
[**diffusion_implicite** *int*]
[**seuil_diffusion_implicite** *float*]
[**impr_diffusion_implicite** *int*]
[**no_error_if_not_converged_diffusion_implicite** *int*]
[**no_conv_subiteration_diffusion_implicite** *int*]
[**dt_start** *dt_start*]
[**nb_pas_dt_max** *int*]
[**niter_max_diffusion_implicite** *int*]
[**precision_impr** *int*]
[**periode_sauvegarde_securite_en_heures** *float*]
[**no_check_disk_space**]
[**disable_progress**]
[**disable_dt_ev**]
[**gnuplot_header** *int*]

}
where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
  Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually

if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.

- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (9.6) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
  dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
  dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
  By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

27 solveur_implicite_base

Description: Class for solver in the situation where the time scheme is the implicit scheme. Solver allows equation diffusion and convection operators to be set as implicit terms.

See also: objet_u (34) solveur_lineaire_std (27.7) simpler (27.6)

Usage:

27.1 Ice

Description: Implicit Continuous-fluid Eulerian solver which is useful for a multiphase problem. Robust pressure reduction resolution.

See also: sets (27.4)

Usage:
ice *str*
Read *str* {

[**criteres_convergence** *criteres_convergence*]
[**seuil_convergence_implicite** *float*]
[**nb_corrections_max** *int*]
[**seuil_convergence_solveur** *float*]
[**seuil_generation_solveur** *float*]

[**seuil_verification_solveur** *float*]
[**seuil_test_preliminaire_solveur** *float*]
[**solveur** *solveur_sys_base*]
[**no_qdm**]
[**nb_it_max** *int*]
[**controle_residu**]

}
where

- **criteres_convergence** *criteres_convergence* (3.22) for inheritance: Set the convergence thresholds for each unknown (i.e: alpha, temperature, velocity and pressure). The default values are respectively 0.01, 0.1, 0.01 and 100
- **seuil_convergence_implicite** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections then nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections-_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system Ax=B will be solved if residual error ||Ax-B|| is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error ||Ax-B|| is lesser than vrel after the implicit linear system Ax=B has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system Ax=B should be solved by checking if the residual error ||Ax-B|| is bigger than vrel.
- **solveur** *solveur_sys_base* (9.13) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

27.2 Implicite

Description: similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.

See also: piso (27.3)

Usage:
implicite *str*
Read *str* {

[**seuil_convergence_implicite** *float*]
[**nb_corrections_max** *int*]
[**seuil_convergence_solveur** *float*]
[**seuil_generation_solveur** *float*]
[**seuil_verification_solveur** *float*]
[**seuil_test_preliminaire_solveur** *float*]

[**solveur** *solveur_sys_base*]
[**no_qdm**]
[**nb_it_max** *int*]
[**controle_residu**]

}
where

- **seuil_convergence_implicite** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections then nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections-_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system Ax=B will be solved if residual error ||Ax-B|| is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error ||Ax-B|| is lesser than vrel after the implicit linear system Ax=B has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system Ax=B should be solved by checking if the residual error ||Ax-B|| is bigger than vrel.
- **solveur** *solveur_sys_base* (9.13) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

27.3 Piso

Description: Piso (Pressure Implicit with Split Operator) - method to solve N_S.

See also: simpler (27.6) sets (27.4) implicite (27.2) simple (27.5)

Usage:
piso *str*
Read *str* {

[**seuil_convergence_implicite** *float*]
[**nb_corrections_max** *int*]
[**seuil_convergence_solveur** *float*]
[**seuil_generation_solveur** *float*]
[**seuil_verification_solveur** *float*]
[**seuil_test_preliminaire_solveur** *float*]
[**solveur** *solveur_sys_base*]
[**no_qdm**]
[**nb_it_max** *int*]
[**controle_residu**]

}
where

- **seuil_convergence_implicite** *float*: Convergence criteria.
- **nb_corrections_max** *int*: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections then nb-_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system Ax=B will be solved if residual error ||Ax-B|| is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error ||Ax-B|| is lesser than vrel after the implicit linear system Ax=B has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system Ax=B should be solved by checking if the residual error ||Ax-B|| is bigger than vrel.
- **solveur** *solveur_sys_base* (9.13) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

27.4 Sets

Description: Stability-Enhancing Two-Step solver which is useful for a multiphase problem.

See also: piso (27.3) ice (27.1)

Usage:
sets *str*
Read *str* {

> [**criteres_convergence** *criteres_convergence*]
> [**seuil_convergence_implicite** *float*]
> [**nb_corrections_max** *int*]
> [**seuil_convergence_solveur** *float*]
> [**seuil_generation_solveur** *float*]
> [**seuil_verification_solveur** *float*]
> [**seuil_test_preliminaire_solveur** *float*]
> [**solveur** *solveur_sys_base*]
> [**no_qdm**]
> [**nb_it_max** *int*]
> [**controle_residu**]

}
where

- **criteres_convergence** *criteres_convergence* (3.22): Set the convergence thresholds for each unknown (i.e: alpha, temperature, velocity and pressure). The default values are respectively 0.01, 0.1, 0.01 and 100
- **seuil_convergence_implicite** *float* for inheritance: Convergence criteria.

- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections then nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections-_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system Ax=B will be solved if residual error ||Ax-B|| is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error ||Ax-B|| is lesser than vrel after the implicit linear system Ax=B has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system Ax=B should be solved by checking if the residual error ||Ax-B|| is bigger than vrel.
- **solveur** *solveur_sys_base* (9.13) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

27.5 Simple

Description: SIMPLE type algorithm

See also: piso (27.3) solveur_u_p (27.8)

Usage:
simple *str*
Read *str* {

[**relax_pression** *float*]
[**seuil_convergence_implicite** *float*]
[**nb_corrections_max** *int*]
[**seuil_convergence_solveur** *float*]
[**seuil_generation_solveur** *float*]
[**seuil_verification_solveur** *float*]
[**seuil_test_preliminaire_solveur** *float*]
[**solveur** *solveur_sys_base*]
[**no_qdm**]
[**nb_it_max** *int*]
[**controle_residu**]

}
where

- **relax_pression** *float*: Value between 0 and 1 (by default 1), this keyword is used only by the SIMPLE algorithm for relaxing the increment of pressure.
- **seuil_convergence_implicite** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections then nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections-_max is set to 21).

- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system Ax=B will be solved if residual error ||Ax-B|| is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error ||Ax-B|| is lesser than vrel after the implicit linear system Ax=B has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system Ax=B should be solved by checking if the residual error ||Ax-B|| is bigger than vrel.
- **solveur** *solveur_sys_base* (9.13) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

27.6 Simpler

Description: Simpler method for incompressible systems.

See also: solveur_implicite_base (27) piso (27.3)

Usage:
simpler *str*
Read *str* {

> **seuil_convergence_implicite** *float*
> [**seuil_convergence_solveur** *float*]
> [**seuil_generation_solveur** *float*]
> [**seuil_verification_solveur** *float*]
> [**seuil_test_preliminaire_solveur** *float*]
> [**solveur** *solveur_sys_base*]
> [**no_qdm**]
> [**nb_it_max** *int*]
> [**controle_residu**]

}
where

- **seuil_convergence_implicite** *float*: Keyword to set the value of the convergence criteria for the resolution of the implicit system build to solve either the Navier_Stokes equation (only for Simple and Simpler algorithms) or a scalar equation. It is adviced to use the default value (1e6) to solve the implicit system only once by time step. This value must be decreased when a coupling between problems is considered.
- **seuil_convergence_solveur** *float*: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float*: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system Ax=B will be solved if residual error ||Ax-B|| is lesser than vrel).

- **seuil_verification_solveur** *float*: Option to check if residual error ||Ax-B|| is lesser than vrel after the implicit linear system Ax=B has been solved.
- **seuil_test_preliminaire_solveur** *float*: Option to decide if the implicit linear system Ax=B should be solved by checking if the residual error ||Ax-B|| is bigger than vrel.
- **solveur** *solveur_sys_base* (9.13): Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** : Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** : Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

27.7 Solveur_lineaire_std

Description: not_set

See also: solveur_implicite_base (27)

Usage:
solveur_lineaire_std *str*
Read *str* {

  [**solveur** *solveur_sys_base*]

}
where

- **solveur** *solveur_sys_base* (9.13)

27.8 Solveur_u_p

Description: similar to simple.

See also: simple (27.5)

Usage:
solveur_u_p *str*
Read *str* {

  [**relax_pression** *float*]
  [**seuil_convergence_implicite** *float*]
  [**nb_corrections_max** *int*]
  [**seuil_convergence_solveur** *float*]
  [**seuil_generation_solveur** *float*]
  [**seuil_verification_solveur** *float*]
  [**seuil_test_preliminaire_solveur** *float*]
  [**solveur** *solveur_sys_base*]
  [**no_qdm**]
  [**nb_it_max** *int*]
  [**controle_residu**]

}
where

- **relax_pression** *float* for inheritance: Value between 0 and 1 (by default 1), this keyword is used only by the SIMPLE algorithm for relaxing the increment of pressure.

- **seuil_convergence_implicite** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections then nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections-_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system Ax=B will be solved if residual error ||Ax-B|| is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error ||Ax-B|| is lesser than vrel after the implicit linear system Ax=B has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system Ax=B should be solved by checking if the residual error ||Ax-B|| is bigger than vrel.
- **solveur** *solveur_sys_base* (9.13) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

28 source_base

Description: Basic class of source terms introduced in the equation.

See also: objet_u (34) source_generique (28.22) boussinesq_temperature (28.4) boussinesq_concentration (28.3) dirac (28.8) puissance_thermique (28.19) source_qdm_lambdaup (28.27) source_th_tdivu (28.31) source_robin (28.28) source_robin_scalaire (28.29) canal_perio (28.5) source_constituant (28.21) radioactive-_decay (28.20) acceleration (28.2) coriolis (28.6) source_qdm (28.26) perte_charge_singuliere (28.18) DP-_Impose (28.1) terme_puissance_thermique_echange_impose (28.32) perte_charge_directionnelle (28.14) perte_charge_isotrope (28.15) perte_charge_anisotrope (28.12) perte_charge_circulaire (28.13) darcy (28.7) forchheimer (28.10) perte_charge_reguliere (28.16) flux_interfacial (28.9) frottement_interfacial (28.11) travail_pression (28.33) source_pdf_base (28.25)

Usage:

28.1 Dp_impose

Description: Source term to impose a pressure difference according to the formula : DP = A + B * (Q - Q0)

See also: source_base (28)

Usage:
DP_Impose *str*
Read *str* {

> **dp** *champ_base*
> **surface** *bloc_lecture*

}

where

- **dp** *champ_base* (14.1): the parameters of the previous formula champ_uniforme 3 A B Q0 where Q0 is a volume flow (m3/s).
- **surface** *bloc_lecture* (3.17): Three syntaxes are possible for the surface definition block:
  For VDF and VEF: { X|Y|Z = location subzone_name }
  Only for VEF: { Surface surface_name }.
  For polymac { Surface surface_name Orientation champ_uniforme }.

28.2 Acceleration

Description: Momentum source term to take in account the forces due to rotation or translation of a non Galilean referential R' (centre 0') into the Galilean referential R (centre 0).

See also: source_base (28)

Usage:
acceleration *str*
Read *str* {

> [**vitesse** *champ_base*]
> [**acceleration** *champ_base*]
> [**omega** *champ_base*]
> [**domegadt** *champ_base*]
> [**centre_rotation** *champ_base*]
> [**option** *str into ['terme_complet', 'coriolis_seul', 'entrainement_seul']*]

}
where

- **vitesse** *champ_base* (14.1): Keyword for the velocity of the referential R' into the R referential (dOO'/dt term [m.s-1]). The velocity is mandatory when you want to print the total cinetic energy into the non-mobile Galilean referential R (see Ec_dans_repere_fixe keyword).
- **acceleration** *champ_base* (14.1): Keyword for the acceleration of the referential R' into the R referential (d2OO'/dt2 term [m.s-2]). field_base is a time dependant field (eg: Champ_Fonc_t).
- **omega** *champ_base* (14.1): Keyword for a rotation of the referential R' into the R referential [rad.s-1]. field_base is a 3D time dependant field specified for example by a Champ_Fonc_t keyword. The time_field field should have 3 components even in 2D (In 2D: 0 0 omega).
- **domegadt** *champ_base* (14.1): Keyword to define the time derivative of the previous rotation [rad.s-2]. Should be zero if the rotation is constant. The time_field field should have 3 components even in 2D (In 2D: 0 0 domegadt).
- **centre_rotation** *champ_base* (14.1): Keyword to specify the centre of rotation (expressed in R' coordinates) of R' into R (if the domain rotates with the R' referential, the centre of rotation is 0'=(0,0,0)). The time_field should have 2 or 3 components according the dimension 2 or 3.
- **option** *str into ['terme_complet', 'coriolis_seul', 'entrainement_seul']*: Keyword to specify the kind of calculation: terme_complet (default option) will calculate both the Coriolis and centrifugal forces, coriolis_seul will calculate the first one only, entrainement_seul will calculate the second one only.

28.3 Boussinesq_concentration

Description: Class to describe a source term that couples the movement quantity equation and constituent transport equation with the Boussinesq hypothesis.

See also: source_base (28)

Usage:
boussinesq_concentration *str*
Read *str* {

> **c0** *n x1 x2 ... xn*
> [**verif_boussinesq** *int*]

}
where

- **c0** *n x1 x2 ... xn*: Reference concentration field type. The only field type currently available is Champ_Uniforme (Uniform field).
- **verif_boussinesq** *int*: Keyword to check (1) or not (0) the reference concentration in comparison with the mean concentration value in the domain. It is set to 1 by default.

28.4 Boussinesq_temperature

Description: Class to describe a source term that couples the movement quantity equation and energy equation with the Boussinesq hypothesis.

See also: source_base (28)

Usage:
boussinesq_temperature *str*
Read *str* {

> **t0** *str*
> [**verif_boussinesq** *int*]

}
where

- **t0** *str*: Reference temperature value (oC or K). It can also be a time dependant function since the 1.6.6 version.
- **verif_boussinesq** *int*: Keyword to check (1) or not (0) the reference temperature in comparison with the mean temperature value in the domain. It is set to 1 by default.

28.5 Canal_perio

Description: Momentum source term to maintain flow rate. The expression of the source term is:
S(t) = (2*(Q(0) - Q(t))-(Q(0)-Q(t-dt))/(coeff*dt*area)

Where:
coeff=damping coefficient
area=area of the periodic boundary
Q(t)=flow rate at time t
dt=time step

Three files will be created during calculation on a datafile named DataFile.data. The first file contains the flow rate evolution. The second file is useful for resuming a calculation with the flow rate of the previous stopped calculation, and the last one contains the pressure gradient evolution:

-DataFile_Channel_Flow_Rate_ProblemName_BoundaryName
-DataFile_Channel_Flow_Rate_repr_ProblemName_BoundaryName
-DataFile_Pressure_Gradient_ProblemName_BoundaryName

See also: source_base (28)

Usage:
canal_perio *str*
Read *str* {

> **bord** *str*
> [**h** *float*]
> [**coeff** *float*]
> [**debit_impose** *float*]

}
where

- **bord** *str*: The name of the (periodic) boundary normal to the flow direction.
- **h** *float*: Half heigth of the channel.
- **coeff** *float*: Damping coefficient (optional, default value is 10).
- **debit_impose** *float*: Optional option to specify the aimed flow rate Q(0). If not used, Q(0) is computed by the code after the projection phase, where velocity initial conditions are slighlty changed to verify incompressibility.

28.6 Coriolis

Description: Keyword for a Coriolis term in hydraulic equation. Warning: Only available in VDF.

See also: source_base (28)

Usage:
coriolis **omega**
where

- **omega** *str*: Value of omega.

28.7 Darcy

Description: Class for calculation in a porous media with source term of Darcy -nu/K*V. This keyword must be used with a permeability model. For the moment there are two models : permeability constant or Ergun's law. Darcy source term is available for quasi compressible calculation. A new keyword is aded for porosity (porosite).

See also: source_base (28)

Usage:
darcy **bloc**
where

- **bloc** *bloc_lecture* (3.17): Description.

28.8 Dirac

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: source_base (28)

Usage:
dirac position ch
where

- **position** *n x1 x2 ... xn*
- **ch** *champ_base* (14.1): Thermal power field type. To impose a volume power on a domain sub-area, the Champ_Uniforme_Morceaux (partly_uniform_field) type must be used.
  Warning : The volume thermal power is expressed in W.m-3.

28.9 Flux_interfacial

Description: Source term of mass transfer between phases connected by the saturation object defined in saturation_xxxx

See also: source_base (28)

Usage:
flux_interfacial

28.10 Forchheimer

Description: Class to add the source term of Forchheimer -Cf/sqrt(K)*V2 in the Navier-Stokes equations. We must precise a permeability model : constant or Ergun's law. Moreover we can give the constant Cf : by default its value is 1. Forchheimer source term is available also for quasi compressible calculation. A new keyword is aded for porosity (porosite).

See also: source_base (28)

Usage:
forchheimer bloc
where

- **bloc** *bloc_lecture* (3.17): Description.

28.11 Frottement_interfacial

Description: Source term which corresponds to the phases friction at the interface

See also: source_base (28)

Usage:
frottement_interfacial [**model**] [**bloc_bulles**]
where

- **model** *str into ['bulles', 'wallis', 'sonnenburg']*: Correlation for friction in bubbly flows if bulles, Correlation for drift flux of Sonnenburg if sonnenburg or Correlation for friction in annular flows if wallis
- **bloc_bulles** *bloc_b* (3.15): not set

28.12 Perte_charge_anisotrope

Description: Anisotropic pressure loss.

See also: source_base (28)

Usage:
perte_charge_anisotrope *str*
Read *str* {

> **lambda** *str*
> **lambda_ortho** *str*
> **diam_hydr** *champ_don_base*
> **direction** *champ_don_base*
> [**sous_zone** *str*]

}
where

- **lambda** *str*: Function for loss coefficient which may be Reynolds dependant (Ex: 64/Re).
- **lambda_ortho** *str*: Function for loss coefficient in transverse direction which may be Reynolds dependant (Ex: 64/Re).
- **diam_hydr** *champ_don_base* (14.6): Hydraulic diameter value.
- **direction** *champ_don_base* (14.6): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

28.13 Perte_charge_circulaire

Description: New pressure loss.

See also: source_base (28)

Usage:
perte_charge_circulaire *str*
Read *str* {

> **lambda** *str*
> **lambda_ortho** *str*
> **diam_hydr** *champ_don_base*
> **diam_hydr_ortho** *champ_don_base*
> **direction** *champ_don_base*
> [**sous_zone** *str*]

}
where

- **lambda** *str*: Function f(Re_tot, Re_long, t, x, y, z) for loss coefficient in the longitudinal direction
- **lambda_ortho** *str*: function: Function f(Re_tot, Re_ortho, t, x, y, z) for loss coefficient in transverse direction
- **diam_hydr** *champ_don_base* (14.6): Hydraulic diameter value.
- **diam_hydr_ortho** *champ_don_base* (14.6): Transverse hydraulic diameter value.
- **direction** *champ_don_base* (14.6): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

28.14 Perte_charge_directionnelle

Description: Directional pressure loss.

See also: source_base (28)

Usage:
perte_charge_directionnelle *str*
Read *str* {

> **lambda** *str*
> **diam_hydr** *champ_don_base*
> **direction** *champ_don_base*
> [**sous_zone** *str*]

}
where

- **lambda** *str*: Function for loss coefficient which may be Reynolds dependant (Ex: 64/Re).
- **diam_hydr** *champ_don_base* (14.6): Hydraulic diameter value.
- **direction** *champ_don_base* (14.6): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

28.15 Perte_charge_isotrope

Description: Isotropic pressure loss.

See also: source_base (28)

Usage:
perte_charge_isotrope *str*
Read *str* {

> **lambda** *str*
> **diam_hydr** *champ_don_base*
> [**sous_zone** *str*]

}
where

- **lambda** *str*: Function for loss coefficient which may be Reynolds dependant (Ex: 64/Re).
- **diam_hydr** *champ_don_base* (14.6): Hydraulic diameter value.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

28.16 Perte_charge_reguliere

Description: Source term modelling the presence of a bundle of tubes in a flow.

See also: source_base (28)

Usage:
perte_charge_reguliere **spec** **zone_name**
where

- **spec** *spec_pdcr_base* (28.17): Description of longitudinale or transversale type.
- **zone_name** *str*: Name of the sub-area occupied by the tube bundle. A Sous_Zone (Sub-area) type object called zone_name should have been previously created.

28.17 Spec_pdcr_base

Description: Class to read the source term modelling the presence of a bundle of tubes in a flow. Cf=A Re-B.

See also: objet_lecture (33) longitudinale (28.17.1) transversale (28.17.2)

Usage:
spec_pdcr_base ch_a a [**ch_b**] [**b**]
where

- **ch_a** *str into ['a', 'cf']*: Keyword to be used to set law coefficient values for the coefficient of regular pressure losses.
- **a** *float*: Value of a law coefficient for regular pressure losses.
- **ch_b** *str into ['b']*: Keyword to be used to set law coefficient values for regular pressure losses.
- **b** *float*: Value of a law coefficient for regular pressure losses.

28.17.1 Longitudinale

Description: Class to define the pressure loss in the direction of the tube bundle.

See also: spec_pdcr_base (28.17)

Usage:
longitudinale dir dd ch_a a [**ch_b**] [**b**]
where

- **dir** *str into ['x', 'y', 'z']*: Direction.
- **dd** *float*: Tube bundle hydraulic diameter value. This value is expressed in m.
- **ch_a** *str into ['a', 'cf']*: Keyword to be used to set law coefficient values for the coefficient of regular pressure losses.
- **a** *float*: Value of a law coefficient for regular pressure losses.
- **ch_b** *str into ['b']*: Keyword to be used to set law coefficient values for regular pressure losses.
- **b** *float*: Value of a law coefficient for regular pressure losses.

28.17.2 Transversale

Description: Class to define the pressure loss in the direction perpendicular to the tube bundle.

See also: spec_pdcr_base (28.17)

Usage:
transversale dir dd chaine_d d ch_a a [**ch_b**] [**b**]
where

- **dir** *str into ['x', 'y', 'z']*: Direction.
- **dd** *float*: Value of the tube bundle step.
- **chaine_d** *str into ['d']*: Keyword to be used to set the value of the tube external diameter.
- **d** *float*: Value of the tube external diameter.
- **ch_a** *str into ['a', 'cf']*: Keyword to be used to set law coefficient values for the coefficient of regular pressure losses.
- **a** *float*: Value of a law coefficient for regular pressure losses.
- **ch_b** *str into ['b']*: Keyword to be used to set law coefficient values for regular pressure losses.
- **b** *float*: Value of a law coefficient for regular pressure losses.

28.18 Perte_charge_singuliere

Description: Source term that is used to model a pressure loss over a surface area (transition through a grid, sudden enlargement) defined by the faces of elements located on the intersection of a subzone named subzone_name and a X,Y, or Z plane located at X,Y or Z = location.

See also: source_base (28)

Usage:
perte_charge_singuliere *str*
Read *str* {

> **dir** *str into ['kx', 'ky', 'kz', 'K']*
> [**coeff** *float*]
> [**regul** *bloc_lecture*]
> **surface** *bloc_lecture*

}
where

- **dir** *str into ['kx', 'ky', 'kz', 'K']*: KX, KY or KZ designate directional pressure loss coefficients for respectively X, Y or Z direction. Or in the case where you chose a target flow rate with regul. Use K for isotropic pressure loss coefficient
- **coeff** *float*: Value (float) of friction coefficient (KX, KY, KZ).
- **regul** *bloc_lecture* (3.17): option to have adjustable K with flowrate target
  { K0 valeur_initiale_de_k deb debit_cible eps intervalle_variation_mutiplicatif}.
- **surface** *bloc_lecture* (3.17): Three syntaxes are possible for the surface definition block:
  For VDF and VEF: { X|Y|Z = location subzone_name }
  Only for VEF: { Surface surface_name }.
  For polymac { Surface surface_name Orientation champ_uniforme }

28.19 Puissance_thermique

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: source_base (28)

Usage:
puissance_thermique ch
where

- **ch** *champ_base* (14.1): Thermal power field type. To impose a volume power on a domain sub-area, the Champ_Uniforme_Morceaux (partly_uniform_field) type must be used.
  Warning : The volume thermal power is expressed in W.m-3 in 3D (in W.m-2 in 2D). It is a power per volume unit (in a porous media, it is a power per fluid volume unit).

28.20 Radioactive_decay

Description: Radioactive decay source term of the form $-\lambda\_ic\_i$, where $0 \leq i \leq N$, N is the number of component of the constituent, c_i and λ_i are the concentration and the decay constant of the i-th component of the constituant.

See also: source_base (28)

Usage:
radioactive_decay **val**
where

- **val** *n x1 x2 ... xn*: n is the number of decay constants to read (int), and val1, val2... are the decay constants (double)

28.21 Source_constituant

Description: Keyword to specify source rates, in [[C]/s], for each one of the nb constituents. [C] is the concentration unit.

See also: source_base (28)

Usage:
source_constituant **ch**
where

- **ch** *champ_base* (14.1): Field type.

28.22 Source_generique

Description: to define a source term depending on some discrete fields of the problem and (or) analytic expression. It is expressed by the way of a generic field usually used for post-processing.

See also: source_base (28)

Usage:
source_generique **champ**
where

- **champ** *champ_generique_base* (7): the source field

28.23 Source_pdf

Description: Source term for Penalised Direct Forcing (PDF) method.

See also: source_pdf_base (28.25)

Usage:
source_pdf *str*
Read *str* {

> **aire** *champ_base*
> **rotation** *champ_base*
> [**transpose_rotation**]
> **modele** *bloc_pdf_model*
> [**interpolation** *interpolation_ibm_base*]

}
where

- **aire** *champ_base* (14.1) for inheritance: volumic field: a boolean for the cell (0 or 1) indicating if the obstacle is in the cell
- **rotation** *champ_base* (14.1) for inheritance: volumic field with 9 components representing the change of basis on cells (local to global). Used for rotating cases for example.
- **transpose_rotation** for inheritance: whether to transpose the basis change matrix.
- **modele** *bloc_pdf_model* (28.24) for inheritance: model used for the Penalized Direct Forcing
- **interpolation** *interpolation_ibm_base* (16) for inheritance: interpolation method

28.24 Bloc_pdf_model

Description: not_set

See also: objet_lecture (33)

Usage:
{

> **eta** *float*
> [**temps_relaxation_coefficient_PDF** *float*]
> [**echelle_relaxation_coefficient_PDF** *float*]
> [**local**]
> [**vitesse_imposee_data** *champ_base*]
> [**vitesse_imposee_fonction** *troismots*]

}
where

- **eta** *float*: penalization coefficient
- **temps_relaxation_coefficient_PDF** *float*: time relaxation on the forcing term to help
- **echelle_relaxation_coefficient_PDF** *float*: time relaxation on the forcing term to help convergence

- **local** : rien whether the prescribed velocity is expressed in the global or local basis
- **vitesse_imposee_data** *champ_base* (14.1): Prescribed velocity as a field
- **vitesse_imposee_fonction** *troismots* (28.24.1): Prescribed velocity as a set of ananlytical component

28.24.1 Troismots

Description: Three words.

See also: objet_lecture (33)

Usage:
mot_1 **mot_2** **mot_3**
where

- **mot_1** *str*: First word.
- **mot_2** *str*: Snd word.
- **mot_3** *str*: Third word.

28.25 Source_pdf_base

Description: Base class of the source term for the Immersed Boundary Penalized Direct Forcing method (PDF)

See also: source_base (28) source_pdf (28.23)

Usage:
source_pdf_base *str*
Read *str* {

> **aire** *champ_base*
> **rotation** *champ_base*
> [**transpose_rotation**]
> **modele** *bloc_pdf_model*
> [**interpolation** *interpolation_ibm_base*]

}
where

- **aire** *champ_base* (14.1): volumic field: a boolean for the cell (0 or 1) indicating if the obstacle is in the cell
- **rotation** *champ_base* (14.1): volumic field with 9 components representing the change of basis on cells (local to global). Used for rotating cases for example.
- **transpose_rotation** : whether to transpose the basis change matrix.
- **modele** *bloc_pdf_model* (28.24): model used for the Penalized Direct Forcing
- **interpolation** *interpolation_ibm_base* (16): interpolation method

28.26 Source_qdm

Description: Momentum source term in the Navier-Stokes equations.

See also: source_base (28)

Usage:
source_qdm ch
where

- **ch** *champ_base* (14.1): Field type.

28.27 Source_qdm_lambdaup

Description: This source term is a dissipative term which is intended to minimise the energy associated to non-conformscales u' (responsible for spurious oscillations in some cases). The equation for these scales can be seen as: du'/dt= -lambda. u' + grad P' where -lambda. u' represents the dissipative term, with lambda = a/Delta t For Crank-Nicholson temporal scheme, recommended value for a is 2.
Remark : This method requires to define a filtering operator.

See also: source_base (28)

Usage:
source_qdm_lambdaup *str*
Read *str* {

> **lambda** *float*
> [**lambda_min** *float*]
> [**lambda_max** *float*]
> [**ubar_umprim_cible** *float*]

}
where

- **lambda** *float*: value of lambda
- **lambda_min** *float*: value of lambda_min
- **lambda_max** *float*: value of lambda_max
- **ubar_umprim_cible** *float*: value of ubar_umprim_cible

28.28 Source_robin

Description: This source term should be used when a Paroi_decalee_Robin boundary condition is set in a hydraulic equation. The source term will be applied on the N specified boundaries. To post-process the values of tauw, u_tau and Reynolds_tau into the files tauw_robin.dat, reynolds_tau_robin.dat and u_tau-_robin.dat, you must add a block Traitement_particulier { canal { } }

See also: source_base (28)

Usage:
source_robin bords
where

- **bords** *vect_nom* (3.116)

28.29 Source_robin_scalaire

Description: This source term should be used when a Paroi_decalee_Robin boundary condition is set in a an energy equation. The source term will be applied on the N specified boundaries. The values temp_wall-_valueI are the temperature specified on the Ith boundary. The last value dt_impr is a printing period which is mandatory to specify in the data file but has no effect yet.

See also: source_base (28)

Usage:
source_robin_scalaire bords
where

- **bords** *listdeuxmots_sacc* (28.30)

28.30 Listdeuxmots_sacc

Description: List of groups of two words (without curly brackets).

See also: listobj (32.5)

Usage:
n object1 object2 ....
list of *deuxmots* (5.23)

28.31 Source_th_tdivu

Description: This term source is dedicated for any scalar (called T) transport. Coupled with upwind (amont) or muscl scheme, this term gives for final expression of convection : div(U.T)-T.div (U)=U.grad(T) This

ensures, in incompressible flow when divergence free is badly resolved, to stay in a better way in the physical boundaries.
Warning: Only available in VEF discretization.

See also: source_base (28)

Usage:
source_th_tdivu

28.32 Terme_puissance_thermique_echange_impose

Description: Source term to impose thermal power according to formula : P = himp * (T - Text). Where T is the Trust temperature, Text is the outside temperature with which energy is exchanged via an exchange coefficient himp

See also: source_base (28)

Usage:
terme_puissance_thermique_echange_impose *str*
Read *str* {

> **himp** *champ_base*
> **Text** *champ_base*

}
where

- **himp** *champ_base* (14.1): the exchange coefficient
- **Text** *champ_base* (14.1): the outside temperature

28.33 Travail_pression

Description: Source term which corresponds to the additional pressure work term that appears when dealing with compressible multiphase fluids

See also: source_base (28)

Usage:
travail_pression

29 sous_zone

Description: It is an object type describing a domain sub-set.
A Sous_Zone (Sub-area) type object must be associated with a Domaine type object. The Read (Lire) interpretor is used to define the items comprising the sub-area.
Caution: The Domain type object nom_domaine must have been meshed (and triangulated or tetrahedralised in VEF) prior to carrying out the Associate (Associer) nom_sous_zone nom_domaine instruction; this instruction must always be preceded by the read instruction.

See also: objet_u (34)

Usage:
sous_zone *str*
Read *str* {

[**restriction** *str*]
[**rectangle** *bloc_origine_cotes*]
[**segment** *bloc_origine_cotes*]
[**boite** *bloc_origine_cotes*]
[**liste** *n n1 n2 ... nn*]
[**fichier** *str*]
[**intervalle** *deuxentiers*]
[**polynomes** *bloc_lecture*]
[**couronne** *bloc_couronne*]
[**tube** *bloc_tube*]
[**fonction_sous_zone** *str*]
[**union** *str*]

}
where

- **restriction** *str*: The elements of the sub-area nom_sous_zone must be included into the other sub-area named nom_sous_zone2. This keyword should be used first in the Read keyword.
- **rectangle** *bloc_origine_cotes* (29.1): The sub-area will include all the domain elements whose centre of gravity is within the Rectangle (in dimension 2).
- **segment** *bloc_origine_cotes* (29.1)
- **boite** *bloc_origine_cotes* (29.1): The sub-area will include all the domain elements whose centre of gravity is within the Box (in dimension 3).
- **liste** *n n1 n2 ... nn*: The sub-area will include n domain items, numbers No. 1 No. i No. n.
- **fichier** *str*: The sub-area is read into the file filename.
- **intervalle** *deuxentiers* (29.2): The sub-area will include domain items whose number is between n1 and n2 (where n1<=n2).
- **polynomes** *bloc_lecture* (3.17): A REPRENDRE
- **couronne** *bloc_couronne* (29.3): In 2D case, to create a couronne.
- **tube** *bloc_tube* (29.4): In 3D case, to create a tube.
- **fonction_sous_zone** *str*: Keyword to build a sub-area with the the elements included into the area defined by fonction>0.
- **union** *str*: The elements of the sub-area nom_sous_zone3 will be added to the sub-area nom_sous-_zone. This keyword should be used last in the Read keyword.

29.1 Bloc_origine_cotes

Description: Class to create a rectangle (or a box).

See also: objet_lecture (33)

Usage:
name origin name2 cotes
where

- **name** *str into ['Origine']*: Keyword to define the origin of the rectangle (or the box).
- **origin** *x1 x2 (x3)*: Coordinates of the origin of the rectangle (or the box).
- **name2** *str into ['Cotes']*: Keyword to define the length along the axes.
- **cotes** *x1 x2 (x3)*: Length along the axes.

29.2 Deuxentiers

Description: Two integers.

See also: objet_lecture (33)

Usage:
int1 int2
where

- **int1** *int*: First integer.
- **int2** *int*: Second integer.

29.3 Bloc_couronne

Description: Class to create a couronne (2D).

See also: objet_lecture (33)

Usage:
name origin name3 ri name4 re
where

- **name** *str into ['Origine']*: Keyword to define the center of the circle.
- **origin** *x1 x2 (x3)*: Center of the circle.
- **name3** *str into ['ri']*: Keyword to define the interior radius.
- **ri** *float*: Interior radius.
- **name4** *str into ['re']*: Keyword to define the exterior radius.
- **re** *float*: Exterior radius.

29.4 Bloc_tube

Description: Class to create a tube (3D).

See also: objet_lecture (33)

Usage:
name origin name2 direction name3 ri name4 re name5 h
where

- **name** *str into ['Origine']*: Keyword to define the center of the tube.
- **origin** *x1 x2 (x3)*: Center of the tube.
- **name2** *str into ['dir']*: Keyword to define the direction of the main axis.
- **direction** *str into ['X', 'Y', 'Z']*: direction of the main axis X, Y or Z
- **name3** *str into ['ri']*: Keyword to define the interior radius.
- **ri** *float*: Interior radius.
- **name4** *str into ['re']*: Keyword to define the exterior radius.
- **re** *float*: Exterior radius.
- **name5** *str into ['hauteur']*: Keyword to define the heigth of the tube.
- **h** *float*: Heigth of the tube.

30 turbulence_paroi_base

Description: Basic class for wall laws for Navier-Stokes equations.

See also: objet_u (34)

Usage:

31 turbulence_paroi_scalaire_base

Description: Basic class for wall laws for energy equation.

See also: objet_u (34)

Usage:

32 listobj_impl

Description: not_set

See also: objet_u (34) listobj (32.5)

Usage:

32.1 List_un_pb

Description: pour les groupes

See also: listobj (32.5)

Usage:
{ object1 , object2 .... }
list of *un_pb* (32.2) separeted with ,

32.2 Un_pb

Description: pour les groupes

See also: objet_lecture (33)

Usage:
mot
where

- **mot** *str*: the string

32.3 Liste_sonde_tble

Description: not_set

See also: listobj (32.5)

Usage:
n object1 object2 ....
list of *sonde_tble* (32.4)

32.4 Sonde_tble

Description: not_set

See also: objet_lecture (33)

Usage:
name point
where

- **name** *str*
- **point** *un_point* (3.19.3)

32.5 Listobj

Description: List of objects.

See also: listobj_impl (32) champs_a_post (4.2.21) list_stat_post (4.2.24) listpoints (4.2.7) sondes (4.2.3) listchamp_generique (7.3) list_nom_virgule (7.2) definition_champs (4.2.1) post_processings (4.3) liste_post (4.5) liste_post_ok (4.4) condinits (5.4) condlims (5.5) sources (5.6) vect_nom (3.116) list_nom (3.101) list_bord (3.61.4) list_bloc_mailler (3.61) list_un_pb (32.1) list_list_nom (4.10) ecrire_fichier_xyz_valeur_param (5.7) pp (5.20) listdeuxmots_sacc (28.30) liste_sonde_tble (32.3) list_info_med (4.28) listsous_zone_valeur (5.2.12) reactions (8.1) listeqn (4.12)

Usage:

33 objet_lecture

Description: Auxiliary class for reading.

See also: objet_u (34) bloc_lecture (3.17) deuxmots (5.23) troismots (28.24.1) format_file (4.6) deuxentiers (29.2) floatfloat (5.24) entierfloat (33.1) champ_a_post (4.2.22) champs_posts (4.2.20) stat_post_deriv (4.2.25) stats_posts (4.2.23) stats_serie_posts (4.2.31) sonde_base (4.2.5) un_point (3.19.3) sonde (4.2.4) definition_champ (4.2.2) postraitement_base (4.4.2) un_postraitement (4.3.1) type_un_post (4.5.2) type_postraitement_ft_lata (4.5.3) un_postraitement_spec (4.5.1) nom_postraitement (4.4.1) condinit (5.4.1) condlimlu (5.5.1) mailler_base (3.61.1) defbord (3.61.7) bord_base (3.61.5) bloc_pave (3.61.3) un_pb (32.2) bords_ecrire (5.7.2) ecrire_fichier_xyz_valeur_item (5.7.1) convection_deriv (5.2.1) bloc_convection (5.2) diffusion_deriv (5.3.1) op_implicite (5.3.9) bloc_diffusion (5.3) parametre_equation_base (5.8) traitement_particulier_base (5.25.1) traitement_particulier (5.25) penalisation_l2_ftd_lec (5.20.1) dt_impr_ustar_mean_only (33.2) modele_turbulence_hyd_deriv (33.3) paroi_ft_disc_deriv (33.4) form_a_nb_points (33.5) fourfloat (33.6) twofloat (33.7) sonde_tble (32.4) remove_elem_bloc (3.91) lecture_bloc_moment_base (3.19) bloc_origine_cotes (29.1) bloc_couronne (29.3) bloc_tube (29.4) verifiercoin_bloc (3.119) bloc_lecture_poro (3.75) bloc_lec_champ_init_canal_sinal (14.17) fonction_champ_reprise (14.13) troisf (3.47) spec_pdcr_base (28.17) info_med (4.28.1) methode_transport_deriv (33.8) decoup (14.3) bloc_ef (5.2.9) sous_zone_valeur (5.2.13) bloc_diffusion_standard (5.3.7) reaction (8.1.1) bloc_pdf_model (28.24) bloc_sutherland (20.11) format_lata_to_med (3.57) bloc_decouper (3.69)

Usage:

33.1 Entierfloat

Description: An integer and a real.

See also: objet_lecture (33)

Usage:
the_int the_float
where

- **the_int** *int*: Integer.
- **the_float** *float*: Real.

33.2 Dt_impr_ustar_mean_only

Description: not_set

See also: objet_lecture (33)

Usage:
{

dt_impr *float*
[**boundaries** *n word1 word2 ... wordn*]

}
where

- **dt_impr** *float*
- **boundaries** *n word1 word2 ... wordn*

33.3 Modele_turbulence_hyd_deriv

Description: Basic class for turbulence model for Navier-Stokes equations.

See also: objet_lecture (33)

Usage:
modele_turbulence_hyd_deriv {

[**correction_visco_turb_pour_controle_pas_de_temps**]
[**correction_visco_turb_pour_controle_pas_de_temps_parametre** *float*]
[**turbulence_paroi** *turbulence_paroi_base*]
[**dt_impr_ustar** *float*]
[**dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only*]
[**nut_max** *float*]

}
where

- **correction_visco_turb_pour_controle_pas_de_temps** : Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float*: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paroi** *turbulence_paroi_base* (30): Keyword to set the wall law.
- **dt_impr_ustar** *float*: This keyword is used to print the values (U +, d+, u$\star$) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.

- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (33.2): This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile-_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float*: Upper limitation of turbulent viscosity (default value 1.e8).

33.4 Paroi_ft_disc_deriv

Description: not_set

See also: objet_lecture (33) symetrie (33.4.1)

Usage:
paroi_ft_disc_deriv

33.4.1 Symetrie

Description: Symetrie condition in the case of two-phase flows

See also: paroi_ft_disc_deriv (33.4)

Usage:
symetrie

33.5 Form_a_nb_points

Description: The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homegeneity planes. Example for channel flows, planes parallel to the walls.

See also: objet_lecture (33)

Usage:
nb dir1 dir2
where

- **nb** *int into [4]*: Number of points.
- **dir1** *int*: First direction.
- **dir2** *int*: Second direction.

33.6 Fourfloat

Description: Four reals.

See also: objet_lecture (33)

Usage:
a b c d
where

- **a** *float*: First real.
- **b** *float*: Second real.
- **c** *float*: Third real.
- **d** *float*: Fourth real.

33.7 Twofloat

Description: two reals.

See also: objet_lecture (33)

Usage:
a b
where

- **a** *float*: First real.
- **b** *float*: Second real.

33.8 Methode_transport_deriv

Description: Basic class for method of transport of interface.

See also: objet_lecture (33) loi_horaire (33.8.1)

Usage:
methode_transport_deriv

33.8.1 Loi_horaire

Description: not_set

See also: methode_transport_deriv (33.8)

Usage:
loi_horaire nom_loi
where

- **nom_loi** *str*

34 index

Index