

TRUST Reference Manual V1.7.4

Support team: triou@cea.fr

Link to: **[TRUST Generic Guide](#)**

December 9, 2016

Contents

1	Syntax to define a mathematical function	12
2	interprete	13
2.1	Raffiner_isotrope_parallele	14
2.2	analyse_angle	14
2.3	associate	14
2.4	axi	15
2.5	bidim_axi	15
2.6	calculer_moments	15
2.7	lecture_bloc_moment_base	15
2.7.1	calcul	16
2.7.2	centre_de_gravite	16
2.7.3	un_point	16
2.8	corriger_frontiere_periodique	16
2.9	create_domain_from_sous_zone	17
2.10	debog	17
2.11	{	18
2.12	decoupebord_pour_rayonnement	18
2.13	decouper_bord_coincident	19
2.14	dilate	19
2.15	dimension	19
2.16	discretiser_domaine	19
2.17	discretize	20
2.18	distance_pari	20
2.19	ecrire_champ_med	20
2.20	ecrire_fichier_formatte	21
2.21	ecriturelecturespecial	21
2.22	execute_parallel	21
2.23	export	22
2.24	extract_2d_from_3d	22
2.25	extract_2daxi_from_3d	22
2.26	extraire_domaine	22
2.27	extraire_plan	23
2.28	extraire_surface	24
2.29	extrudebord	24
2.30	extrudeparoi	25
2.31	extruder	25
2.32	troisf	26
2.33	extruder_en20	26
2.34	extruder_en3	27
2.35	end	27
2.36	}	27
2.37	imprimer_flux	27
2.38	bloc_lecture	28
2.39	imprimer_flux_sum	28
2.40	integrer_champ_med	28
2.41	lata_to_med	29
2.42	format_lata_to_med	29
2.43	lata_to_other	29
2.44	lire_ideas	30
2.45	mailler	30
2.46	list_bloc_mailler	30

2.46.1	mailler_base	30
2.46.2	pave	31
2.46.3	bloc_pave	31
2.46.4	list_bord	32
2.46.5	bord_base	32
2.46.6	bord	32
2.46.7	defbord	32
2.46.8	defbord_2	32
2.46.9	defbord_3	33
2.46.10	raccord	33
2.46.11	internes	34
2.46.12	epsilon	34
2.46.13	domain	34
2.47	maillerparallel	34
2.48	modif_bord_to_raccord	36
2.49	moyenne_volumique	36
2.50	option_vdf	37
2.51	orientefacesbord	37
2.52	partition	37
2.53	bloc_decouper	38
2.54	pilote_icoco	39
2.55	porosites	39
2.56	bloc_lecture_poro	40
2.57	porosites_champ	40
2.58	postraiter_domaine	40
2.59	precisiongeom	41
2.60	raffiner_anisotrope	41
2.61	raffiner_isotrope	41
2.62	read	42
2.63	read_file	42
2.64	read_file_binary	42
2.65	lire_tgrid	42
2.66	read_unsupported_ascii_file_from_icem	43
2.67	read_med	43
2.68	orienter_simplexes	44
2.69	redresser_hexaedres_vdf	44
2.70	regroupebord	44
2.71	remove_elem	44
2.72	remove_elem_bloc	45
2.73	remove_invalid_internal_boundaries	45
2.74	reordonner_faces_periodiques	46
2.75	reorienter_tetraedres	46
2.76	reorienter_triangles	46
2.77	reordonner	46
2.78	rotation	47
2.79	scatter	47
2.80	scattermed	47
2.81	solve	48
2.82	supprime_bord	48
2.83	list_nom	48
2.84	system	48
2.85	test_solveur	49
2.86	testeur	49
2.87	testeur_medcoupling	49

2.88	tetraedriser	50
2.89	tetraedriser_homogene	50
2.90	tetraedriser_homogene_compact	50
2.91	tetraedriser_homogene_fin	51
2.92	tetraedriser_par_prisme	51
2.93	transformer	51
2.94	triangler	52
2.95	triangler_fin	52
2.96	triangler_h	52
2.97	verifier_qualite_raffinements	52
2.98	vect_nom	53
2.99	verifier_simplexes	53
2.100	verifiercoin	53
2.101	ecrire	53
2.102	ecrire_fichier_bin	54
2.103	ecrire_med	54
3	pb_gen_base	54
3.1	Pb_base	54
3.2	corps_postraitement	55
3.2.1	definition_champs	56
3.2.2	definition_champ	56
3.2.3	sondes	56
3.2.4	sonde	57
3.2.5	sonde_base	57
3.2.6	points	57
3.2.7	listpoints	58
3.2.8	point	58
3.2.9	segmentpoints	58
3.2.10	numero_elem_sur_maitre	58
3.2.11	position_like	58
3.2.12	segment	59
3.2.13	plan	59
3.2.14	volume	59
3.2.15	circle	60
3.2.16	circle_3	60
3.2.17	champs_posts	60
3.2.18	champs_a_post	61
3.2.19	champ_a_post	61
3.2.20	stats_posts	61
3.2.21	list_stat_post	62
3.2.22	stat_post_deriv	62
3.2.23	t_deb	62
3.2.24	t_fin	63
3.2.25	moyenne	63
3.2.26	ecart_type	63
3.2.27	correlation	63
3.2.28	stats_serie_posts	64
3.3	post_processings	64
3.3.1	un_postraitement	65
3.4	liste_post_ok	65
3.4.1	nom_postraitement	65
3.4.2	postraitement_base	65
3.4.3	post_processing	65

3.5	liste_post	66
3.5.1	un_postraitement_spec	66
3.5.2	type_un_post	67
3.5.3	type_postraitement_ft_lata	67
3.6	format_file	67
3.7	probleme_couple	67
3.8	list_list_nom	68
3.9	pb_avec_passif	68
3.10	listeqn	69
3.11	pb_conduction	69
3.12	pb_hydraulique	70
3.13	pb_hydraulique_concentration	71
3.14	pb_hydraulique_concentration_scalaires_passifs	72
3.15	pb_hydraulique_concentration_turbulent	73
3.16	pb_hydraulique_concentration_turbulent_scalaires_passifs	74
3.17	pb_hydraulique_turbulent	76
3.18	pb_post	77
3.19	pb_thermohydraulique	77
3.20	pb_thermohydraulique_concentration	78
3.21	pb_thermohydraulique_concentration_scalaires_passifs	80
3.22	pb_thermohydraulique_concentration_turbulent	81
3.23	pb_thermohydraulique_concentration_turbulent_scalaires_passifs	82
3.24	pb_thermohydraulique_qc	83
3.25	pb_thermohydraulique_qc_fraction_massique	84
3.26	pb_thermohydraulique_scalaires_passifs	85
3.27	pb_thermohydraulique_turbulent	86
3.28	pb_thermohydraulique_turbulent_qc	88
3.29	pb_thermohydraulique_turbulent_qc_fraction_massique	89
3.30	pb_thermohydraulique_turbulent_scalaires_passifs	90
3.31	pb_med	91
3.32	list_info_med	91
3.32.1	info_med	91
3.33	problem_read_generic	92
4	mor_eqn	93
4.1	conduction	93
4.2	bloc_diffusion	94
4.2.1	diffusion_deriv	94
4.2.2	negligeable	94
4.2.3	p1b	94
4.2.4	p1ncp1b	94
4.2.5	stab	95
4.2.6	standard	95
4.2.7	bloc_diffusion_standard	96
4.2.8	option	96
4.2.9	op_implicite	96
4.3	condinits	97
4.3.1	condinit	97
4.4	condlims	97
4.4.1	condlimlu	97
4.5	sources	98
4.6	ecrire_fichier_xyz_valeur_param	98
4.6.1	ecrire_fichier_xyz_valeur_item	98
4.6.2	bords_ecrire	98

4.7	parametre_equation_base	99
4.7.1	parametre_diffusion_implicite	99
4.7.2	parametre_implicite	99
4.8	convection_diffusion_chaleur_qc	100
4.9	bloc_convection	101
4.9.1	convection_deriv	101
4.9.2	amont	101
4.9.3	amont_old	102
4.9.4	centre	102
4.9.5	centre4	102
4.9.6	centre_old	102
4.9.7	di_l2	102
4.9.8	ef	102
4.9.9	bloc_ef	103
4.9.10	muscl3	103
4.9.11	ef_stab	104
4.9.12	listsous_zone_valeur	104
4.9.13	sous_zone_valeur	104
4.9.14	generic	105
4.9.15	kquick	105
4.9.16	muscl	105
4.9.17	muscl_old	105
4.9.18	muscl_new	106
4.9.19	negligeable	106
4.9.20	quick	106
4.9.21	supg	106
4.9.22	btd	106
4.10	convection_diffusion_chaleur_turbulent_qc	107
4.11	convection_diffusion_concentration	108
4.12	convection_diffusion_concentration_turbulent	109
4.13	convection_diffusion_fraction_massique_qc	110
4.14	convection_diffusion_fraction_massique_turbulent_qc	111
4.15	convection_diffusion_temperature	112
4.16	pp	113
4.16.1	penalisation_l2_ftd_lec	113
4.17	convection_diffusion_temperature_turbulent	114
4.18	eqn_base	115
4.19	navier_stokes_qc	116
4.20	deuxmots	117
4.21	floatfloat	118
4.22	traitement_particulier	118
4.22.1	traitement_particulier_base	118
4.22.2	temperature	118
4.22.3	canal	119
4.22.4	ec	119
4.22.5	thi	120
4.22.6	chmoy_faceperio	121
4.23	navier_stokes_standard	121
4.24	navier_stokes_turbulent	122
4.25	modele_turbulence_hyd_deriv	124
4.25.1	dt_impr_ustar_mean_only	125
4.25.2	NUL	126
4.25.3	mod_turb_hyd_ss_maille	126
4.25.4	form_a_nb_points	127

4.25.5	sous_maille_wale	128
4.25.6	sous_maille_smago	129
4.25.7	combinaison	130
4.25.8	longueur_melange	132
4.25.9	sous_maille	133
4.25.10	k_epsilon	134
4.25.11	modele_fonction_bas_reynolds_base	136
4.26	navier_stokes_turbulent_qc	136
4.27	transport_k_epsilon	138
5	/*	139
5.1	/*	139
6	champ_generique_base	139
6.1	champ_post_de_champs_post	139
6.2	list_nom_virgule	140
6.3	listchamp_generique	140
6.4	champ_post_operateur_base	140
6.5	champ_post_operateur_eqn	140
6.6	champ_post_statistiques_base	141
6.7	correlation	142
6.8	champ_post_operateur_divergence	142
6.9	ecart_type	143
6.10	champ_post_extraction	143
6.11	champ_post_operateur_gradient	144
6.12	champ_post_interpolation	144
6.13	champ_post_morceau_equation	145
6.14	moyenne	145
6.15	predefini	146
6.16	champ_post_reduction_0d	146
6.17	champ_post_refchamp	147
6.18	champ_post_tparoi_vef	147
6.19	champ_post_transformation	148
7	chimie	149
7.1	reactions	149
7.1.1	reaction	149
8	class_generic	150
8.1	cholesky	150
8.2	dt_calc	150
8.3	dt_fixe	151
8.4	dt_min	151
8.5	dt_start	151
8.6	gcp_ns	151
8.7	gen	152
8.8	gmres	152
8.9	optimal	153
8.10	petsc	153
8.11	gcp	157
8.12	solveur_sys_base	158
9	#	158
9.1	#	158

10	condlim_base	159
10.1	Paroi	159
10.2	dirichlet	159
10.3	entree_temperature_imposee_h	159
10.4	frontiere_ouverte	159
10.5	frontiere_ouverte_concentration_imposee	160
10.6	frontiere_ouverte_fraction_massique_imposee	160
10.7	frontiere_ouverte_gradient_pression_impose	160
10.8	frontiere_ouverte_gradient_pression_impose_vef	161
10.9	frontiere_ouverte_gradient_pression_impose_vefprep1b	161
10.10	frontiere_ouverte_gradient_pression_libre_vef	161
10.11	frontiere_ouverte_gradient_pression_libre_vefprep1b	161
10.12	frontiere_ouverte_k_eps_impose	161
10.13	frontiere_ouverte_pression_imposee	162
10.14	frontiere_ouverte_pression_imposee_orlansky	162
10.15	frontiere_ouverte_pression_moyenne_imposee	162
10.16	frontiere_ouverte_rho_u_impose	162
10.17	frontiere_ouverte_temperature_imposee	163
10.18	frontiere_ouverte_vitesse_imposee	163
10.19	frontiere_ouverte_vitesse_imposee_sortie	163
10.20	neumann	163
10.21	paroi_adiabatique	164
10.22	paroi_contact	164
10.23	paroi_contact_fictif	165
10.24	paroi_couple	165
10.25	paroi_decalee_robin	165
10.26	paroi_defilante	166
10.27	paroi_echange_contact_correlation_vdf	166
10.28	paroi_echange_contact_correlation_vef	167
10.29	paroi_echange_contact_vdf	168
10.30	paroi_echange_externes_impose	168
10.31	paroi_echange_externes_impose_h	168
10.32	paroi_echange_global_impose	169
10.33	paroi_fixe	169
10.34	paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets	169
10.35	paroi_flux_impose	169
10.36	paroi_knudsen_non_negligeable	170
10.37	paroi_rugueuse	170
10.38	paroi_temperature_imposee	170
10.39	periodique	171
10.40	sortie_libre_temperature_imposee_h	171
10.41	symetrie	171
10.42	temperature_imposee_parois	171
11	discretisation_base	171
11.1	ef	172
11.2	vdf	172
11.3	vef	172
11.4	vefprep1b	172
12	domaine	173
13	espece	173

14 champ_base	173
14.1 champ_base	173
14.2 champ_don_base	174
14.3 champ_don_lu	174
14.4 champ_fonc_fonction	174
14.5 champ_fonc_fonction_txyz	174
14.6 champ_fonc_med	175
14.7 champ_fonc_reprise	175
14.8 fonction_champ_reprise	176
14.9 champ_fonc_t	176
14.10 champ_fonc_tabule	176
14.11 champ_init_canal_sinal	176
14.12 bloc_lec_champ_init_canal_sinal	177
14.13 champ_input_base	177
14.14 champ_input_p0	178
14.15 champ_ostwald	178
14.16 champ_som_lu_vdf	179
14.17 champ_som_lu_vef	179
14.18 champ_tabule_temps	179
14.19 champ_uniforme_morceaux	180
14.20 champ_uniforme_morceaux_tabule_temps	180
14.21 champ_fonc_txyz	180
14.22 champ_fonc_xyz	181
14.23 field_uniform_keps_from_ud	181
14.24 init_par_partie	181
14.25 tayl_green	181
14.26 uniform_field	182
14.27 valeur_totale_sur_volume	182
15 champ_front_base	182
15.1 champ_front_base	182
15.2 boundary_field_inward	183
15.3 boundary_field_uniform_keps_from_ud	183
15.4 ch_front_input	183
15.5 ch_front_input_uniforme	184
15.6 champ_front_bruite	184
15.7 champ_front_calc	185
15.8 champ_front_contact_vef	185
15.9 champ_front_debit	185
15.10 champ_front_fonc_pois_ipsn	185
15.11 champ_front_fonc_pois_tube	186
15.12 champ_front_fonc_txyz	186
15.13 champ_front_fonc_xyz	186
15.14 champ_front_fonction	187
15.15 champ_front_lu	187
15.16 champ_front_normal_vef	187
15.17 champ_front_pression_from_u	187
15.18 champ_front_recyclage	188
15.19 champ_front_tabule	189
15.20 champ_front_tangentiel_vef	190
15.21 champ_front_uniforme	190

16	loi_etat_base	190
16.1	gaz_reel_rhot	190
16.2	melange_gaz_parfait	191
16.3	gaz_parfait	191
17	loi_horaire	191
18	milieu_base	192
18.1	constituant	192
18.2	fluide_incompressible	193
18.3	fluide_ostwald	193
18.4	fluide_quasi_compressible	194
18.5	bloc_sutherland	195
18.6	solide	195
19	modele_turbulence_scal_base	196
19.1	prandtl	196
19.2	schmidt	197
20	nom	197
20.1	nom_anonyme	198
21	partitionneur_deriv	198
21.1	fichier_decoupage	198
21.2	metis	199
21.3	partition	199
21.4	sous_zones	200
21.5	tranche	200
22	precond_base	201
22.1	precond_local	201
22.2	precondsolv	201
22.3	ssor	201
22.4	ssor_bloc	202
23	schema_temps_base	202
23.1	Sch_CN_EX_iteratif	204
23.2	Sch_CN_iteratif	206
23.3	scheme_euler_explicit	208
23.4	leap_frog	209
23.5	runge_kutta_ordre_3	211
23.6	runge_kutta_ordre_4_d3p	213
23.7	runge_kutta_rationnel_ordre_2	214
23.8	schema_adams_bashforth_order_2	216
23.9	schema_adams_bashforth_order_3	217
23.10	schema_adams_moulton_order_2	219
23.11	schema_adams_moulton_order_3	221
23.12	schema_backward_differentiation_order_2	223
23.13	schema_backward_differentiation_order_3	226
23.14	scheme_euler_implicit	228
23.15	schema_implicite_base	230
23.16	schema_predictor_corrector	232

24	solveur_implicite_base	234
24.1	implicite	234
24.2	piso	235
24.3	simple	236
24.4	simpler	237
24.5	solveur_lineaire_std	238
25	source_base	238
25.1	Source_Transport_K_Eps_anisotherme	238
25.2	acceleration	239
25.3	boussinesq_concentration	239
25.4	boussinesq_temperature	240
25.5	canal_perio	240
25.6	coriolis	241
25.7	darcy	241
25.8	dirac	241
25.9	forchheimer	242
25.10	perte_charge_anisotrope	242
25.11	perte_charge_circulaire	242
25.12	perte_charge_directionnelle	243
25.13	perte_charge_isotrope	243
25.14	perte_charge_reguliere	244
25.15	spec_pdc_base	244
25.15.1	longitudinale	244
25.15.2	transversale	245
25.16	perte_charge_singuliere	245
25.17	puissance_thermique	246
25.18	source_constituant	246
25.19	source_generique	246
25.20	source_qdm	246
25.21	source_qdm_lambdaup	247
25.22	source_robin	247
25.23	source_robin_scalaire	247
25.24	listdeuxmots_sacc	248
25.25	source_th_tdivu	248
25.26	source_transport_k_eps	248
25.27	source_transport_k_eps_aniso_concen	248
25.28	source_transport_k_eps_aniso_therm_concen	249
26	sous_zone	249
26.1	bloc_origine_cotes	250
26.2	deuxentiers	250
26.3	bloc_couronne	251
26.4	bloc_tube	251
27	turbulence_paro_base	251
27.1	loi_expert_hydr	251
27.2	loi_standard_hydr	252
27.3	loi_standard_hydr_old	252
27.4	negligeable	252
27.5	paroi_tble	253
27.6	twofloat	253
27.7	liste_sonde_tble	254
27.7.1	sonde_tble	254

27.8 entierfloat	254
27.9 utau_imp	254
28 turbulence_paro_scalaire_base	255
28.1 loi_analytique_scalaire	255
28.2 loi_expert_scalaire	255
28.3 loi_paro_nu_impose	256
28.4 loi_standard_hydr_scalaire	256
28.5 negligeable_scalaire	256
28.6 paroi_tble_scal	256
28.7 fourfloat	257
29 listobj_impl	257
29.1 list_un_pb	257
29.2 un_pb	258
29.3 listobj	258
30 objet_lecture	258
30.1 paroi_ft_disc_deriv	258
30.1.1 symetrie	259
30.2 methode_transport_deriv	259
30.2.1 loi_horaire	259
31 index	259

1 Syntax to define a mathematical function

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

ABS : absolute value function
COS : cosinus function
SIN : sinus function
TAN : tan function
ATAN : arctan function
EXP : exponential function
LN : neperian logaithm function
SQRT : root mean square function
INT : integer function
ERF : erf function
RND(x) : random function (values between 0 and x)
COSH : hyperbolic cosinus function
SINH : hyperbolic sinus function
TANH : hyperbolic tangent function
ACOS : inverse cosinus function
ATANH : inverse hyperbolic tangent function
NOT(x) : not equal to x
x_AND_y : and function (returns 1 if x and y true else 0)
x_OR_y : or function (returns 1 if x or y true else 0)
x_GT_y : greater to (returns 1 if x>y else 0)
x_GE_y : greater or equal to (returns 1 if x>=y else 0)
x_LT_y : lesser to (returns 1 if x<y else 0)
x_LE_y : lesser or equal to (returns 1 if x<=y else 0)
x_MIN_y : minimum of x and y
x_MAX_y : maximum 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
`^` : power
`<` : lesser than
`>` : greater than
`[` : less or equal to
`]` : greater of 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+x^2) t+ln(y)`
`Champ_fonc_xyz dom 2 tanh(4*y)*(0.95+0.1*rnd(1)) 0.`

Possible error:

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

2 interpret

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

See also: `objet_u` (31) `read` (2.61) `associate` (2.2) `discretize` (2.16) `mailler` (2.44) `maillerparallel` (2.46.13) `ecrire_fichier_bin` (2.101) `ecrire` (2.100) `read_file` (2.62) `lire_tgrid` (2.64) `solve` (2.80) `execute_parallel` (2.21) `end` (2.34) `dimension` (2.14) `bidim_axi` (2.4) `axi` (2.3) `transformer` (2.92) `rotation` (2.77) `dilate` (2.13) `testeur` (2.85) `test_solveur` (2.84) `postraiter_domaine` (2.57) `modif_bord_to_raccord` (2.47) `remove_elem` (2.70) `regroupebord` (2.69) `supprime_bord` (2.81) `calculer_moments` (2.5) `imprimer_flux` (2.36) `decouper_bord_coincident` (2.12) `raffiner_anisotrope` (2.59) `raffiner_isotrope` (2.60) `triangler` (2.93) `tetraedriser` (2.87) `orientefacesbord` (2.50) `reorienter_tetraedres` (2.74) `reorienter_triangles` (2.75) `verifiercoin` (2.99) `porosites` (2.54) `porosites_champ` (2.56) `discretiser_domaine` (2.15) `{` (2.10) `}` (2.35) `export` (2.22) `debog` (2.9) `pilote_icoco` (2.53) `moyenne_volumique` (2.48) `ecrire_champ_med` (2.18) `read_med` (2.66) `lire_ideas` (2.43) `ecrire_med` (2.102) `system` (2.83) `redresser_hexaedres_vdf` (2.68) `analyse_angle` (2.1) `remove_invalid_internal_boundaries` (2.72) `reordonner` (2.76) `option_vdf` (2.49) `precisiongeom` (2.58) `scatter` (2.78) `partition` (2.51) `reordonner_faces_periodiques` (2.73) `corriger_frontiere_periodique` (2.7.3) `distance_parois` (2.17) `extrudebord` (2.28) `extruder` (2.30) `extract_2d_from_3d` (2.23) `extruder_en20` (2.32) `extrudeparois` (2.29) `ecriturelecturespecial` (2.20) `lata_to_med` (2.40) `lata_to_other` (2.42) `decoupebord_pour_rayonnement` (2.11) `extraire_plan` (2.26) `create_domain_from_sous_zone` (2.8) `extraire_domaine` (2.25) `extraire_surface` (2.27)

integrer_champ_med (2.39) orienter_simplexes (2.67) verifier_simplexes (2.98) verifier_qualite_raffinements (2.96) testeur_medcoupling (2.86) Raffiner_isotrope_parallele (2)

Usage:

interprete

2.1 Raffiner_isotrope_parallele

Description: Refine parallel mesh in parallel

See also: interprete (2)

Usage:

```
Raffiner_isotrope_parallele {  
    name_of_initial_zones str  
    [ ascii ]  
    name_of_new_zones str  
}
```

where

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

2.2 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 (2)

Usage:

```
analyse_angle domain_name nb_histo  
where
```

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

2.3 associate

Description: This interpreter 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 in error because it cannot execute the Associer (Associate) 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 Schema_euler_explicite type object for time discretisation, a discretisation 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 (2)

Usage:

associate objet_1 objet_2

where

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

2.4 axi

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

See also: [interpret \(2\)](#)

Usage:

axi

2.5 bidim_axi

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

See also: [interpret \(2\)](#)

Usage:

bidim_axi

2.6 calculer_moments

Description: Calculate and print the torque (moment of force) exerted by the fluid on each boundaries in output files (.out) of the domain **nom_dom**.

See also: [interpret \(2\)](#)

Usage:

calculer_moments nom_dom mot

where

- **nom_dom** *str*: Name of domain.
- **mot** *lecture_bloc_moment_base (2.6)*: Keyword.

2.7 lecture_bloc_moment_base

Description: Auxiliary class for calcul and print of the moments.

See also: [objet_lecture \(30\)](#) [calcul \(2.7\)](#) [centre_de_gravite \(2.7.1\)](#)

Usage:

2.7.1 calcul

Description: The centre of gravity will be calculated.

See also: (2.6)

Usage:

calcul

2.7.2 centre_de_gravite

Description: To specify a specific centre of gravity.

See also: (2.6)

Usage:

centre_de_gravite point

where

- **point** *un_point* (2.7.2): A centre of gravity.

2.7.3 un_point

Description: A point.

See also: objet_lecture (30)

Usage:

pos

where

- **pos** *x1 x2 (x3)*: Point co-ordinates.

2.8 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: `interpret` (2)

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*: see `corriger_coordonnees`

2.9 create_domain_from_sous_zone

Description: These 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 \(2\)](#)

Usage:

```
create_domain_from_sous_zone {
    domaine_final str
    par_sous_zone str
    domaine_init str
```

```
}
```

where

- **domaine_final** *str*: domaine dans lequel stocke les faces
- **par_sous_zone** *str*: sous zone permettant de choisir les elements
- **domaine_init** *str*: domaine d origine

2.10 debug

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: `Debug::verifier(msg,value);` Where `msg` is a string and `value` may be a double, integer or array.

During the second run (mode=1), it prints into a file `Err_Debug.dbg` the same messages than in the `DEBOG` file and checks if the differences between results from the two codes are less than error. If not, it prints `Ok` else show the differences and the lines where it occurred.

See also: [interprete \(2\)](#)

Usage:

```
debug 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 run with the first code, and 1 for the run with the second code.

2.11 {

Description: Block's beginning.

See also: [interpret \(2\)](#)

Usage:

{

2.12 decoupebord_pour_rayonnement

Description: To subdivide the external boundary of a domain in 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: [interpret \(2\)](#)

Usage:

```
decoupebord_pour_rayonnement {
    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*

2.13 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: [interpret \(2\)](#)

Usage:

decouper_bord_coincident domain_name bord

where

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

2.14 dilate

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

See also: [interpret \(2\)](#)

Usage:

dilate domain_name alpha

where

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

2.15 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: [interpret \(2\)](#)

Usage:

dimension dim

where

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

2.16 discretiser_domaine

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

See also: [interpret \(2\)](#)

Usage:

discretiser_domaine domain_name

where

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

2.17 discretize

Description: Keyword to discretise a problem `problem_name` according to the discretisation `dis`.

IMPORTANT: A number of objects must be already associated (a domain, time scheme, central object) prior to invoking the Discretiser (Discretise) keyword. The physical properties of this central object must also have been read.

See also: [interpret](#) (2)

Usage:

discretize `problem_name` **dis**

where

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

2.18 distance_pari

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 which are associated to walls). A field `Distance_pari` is available to post process the distance to the wall.

See also: [interpret](#) (2)

Usage:

distance_pari `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).

2.19 ecrire_champ_med

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

See also: [interpret](#) (2)

Usage:

ecrire_champ_med `nom_dom` `nom_chp` **file**

where

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

2.20 `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` ([2.101](#))

Usage:

`ecrire_fichier_formatte name_obj filename`

where

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

2.21 `ecriturelecturespecial`

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

See also: `interpret` ([2](#))

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`).

2.22 `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 usually written to `stderr` and `stdout` are redirected to .log files (journaling must be activated).

See also: `interpret` ([2](#))

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.

2.23 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 \(2\)](#)

Usage:

export

2.24 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 \(2\)](#) [extract_2daxi_from_3d \(2.24\)](#)

Usage:

extract_2d_from_3d dom3D bord dom2D

where

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

2.25 extract_2daxi_from_3d

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

See also: [extract_2d_from_3d \(2.23\)](#)

Usage:

extract_2daxi_from_3d dom3D bord dom2D

where

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

2.26 extraire_domaine

Description: Keyword to create a new 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 Discretiser should have already be used to read the object.

See also: [interprete \(2\)](#)

Usage:

extraire_domaine {

domaine *str*

```

    probleme str
    [ condition_elements str]
    [ sous_zone str]
}

```

where

- **domaine** *str*: domaine dans lequel stocke les faces
- **probleme** *str*: Probleme duquel il faut extraire les faces
- **condition_elements** *str*
- **sous_zone** *str*

2.27 extraire_plan

Description: This keyword extract a plan mesh named domain_name (this domain should have be declared before) from the mesh of the pb_name problem. The plan 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 plan 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 plan is a boundary not well oriented, and avec_certain_bords_pour_extraire_surface is the option related to the Extraire_surface option named avec_certain_bords.

Keyword Discretiser should have already be used to read the object.

See also: [interprete \(2\)](#)

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_certain_bords_pour_extraire_surface n word1 word2 ... wordn]
}

```

where

- **domaine** *str*: domain_name
- **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_certain_bords_pour_extraire_surface** *n word1 word2 ... wordn*

2.28 extraire_surface

Description: This keyword extract a surface mesh named `domain_name` (this domain should have be 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 conditions `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_certaines_bords` is used to add only some boundaries.

Keyword Discretiser should have already be used to read the object.

See also: [interprete \(2\)](#)

Usage:

```
extraire_surface {  
    domaine str  
    probleme str  
    [ condition_elements str]  
    [ condition_faces str]  
    [ avec_les_bords ]  
    [ avec_certaines_bords n word1 word2 ... wordn]  
}  
where
```

- **domaine** *str*: domaine dans lequel stocke les faces
- **probleme** *str*: Probleme duquel il faut extraire les faces
- **condition_elements** *str*
- **condition_faces** *str*
- **avec_les_bords**
- **avec_certaines_bords** *n word1 word2 ... wordn*

2.29 extrudebord

Description: Class to generate an extruded mesh from a boundary of a tetrahedral or an hexahedral mesh.

Warning: If the initial domain is an tetrahedral mesh, the boundary will be moved in the XY plan then extrusion will be applied (you should may be 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 \(2\)](#)

Usage:

```
extrudebord {  
    domaine_init str  
    [ direction x1 x2 (x3)]  
    [ nb_tranches int]
```



```

[ domaine_final str]
[ nom_bord str]
[ non_perio ]
[ 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.
- **non_perio** : Extruded domain will not have periodic boundaries. So, the boundaries will be named DEVANT and DERRIERE instead of PERIO.
- **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

2.30 extrudeparoi

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

See also: [interprete \(2\)](#)

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 slide) boundary for creation of prismatic layers.
- **epaisseur** *n x1 x2 ... xn*: n r_1 r_2 r_n : (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. default values are : epaisseur_relative 1 0.5 projection_normale_bord 1

2.31 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 \(2\)](#) [extruder_en3 \(2.33\)](#)

Usage:

```
extruder {  
    domaine str  
    direction troisf  
    nb_tranches int  
}
```

where

- **domaine** *str*: Name of the domain.
- **direction** *troisf* ([2.31](#)): Direction of the extrude operation.
- **nb_tranches** *int*: Number of elements in the extrusion direction.

2.32 troisf

Description: Auxiliary class to extrude.

See also: [objet_lecture \(30\)](#)

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.

2.33 extruder_en20

Description: It does the same task as Extruder except a prism is cut in 20 instead of 3. The nem of the boundaries will be devant and derriere. But you can change this name with the keyword RegroupeBord.

See also: [interprete \(2\)](#)

Usage:

```
extruder_en20 {  
    domaine str  
    [ direction troisf ]  
    nb_tranches int  
}
```

where

- **domaine** *str*: Name of the domain.
- **direction** *troisf* ([2.31](#)): 0 Direction of the extrude operation.
- **nb_tranches** *int*: Number of elements in the extrusion direction.

2.34 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 (by default, *devant* and *derriere*) may be renamed 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* ([2.30](#))

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* ([2.31](#)) for inheritance: Direction of the extrude operation.
- **nb_tranches** *int* for inheritance: Number of elements in the extrusion direction.

2.35 end

Description: Keyword which must complete the data file.

See also: *interpret* ([2](#))

Usage:

```
end
```

2.36 }

Description: Block's end.

See also: *interpret* ([2](#))

Usage:

```
}
```

2.37 imprimer_flux

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

See also: *interpret* ([2](#)) *imprimer_flux_sum* ([2.38](#))

Usage:

imprimer_flux domain_name noms_bord

where

- **domain_name** *str*: Name of the domain.
- **noms_bord** *bloc_lecture* (2.37): Liste des noms des bords ex: { Bord1 Bord2 }

2.38 bloc_lecture

Description: pour lire entre deux accolades

See also: [objet_lecture \(30\)](#)

Usage:

bloc_lecture

where

- **bloc_lecture** *str*

2.39 imprimer_flux_sum

Description: This keyword allows the sum of the flux per face at the boundaries of a domain defined by the user in the data set to be printed. The flux 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 \(2.36\)](#)

Usage:

imprimer_flux_sum domain_name noms_bord

where

- **domain_name** *str*: Name of the domain.
- **noms_bord** *bloc_lecture* (2.37): Liste des noms des bords ex: { Bord1 Bord2 }

2.40 integrer_champ_med

Description: this 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 $z=z_{min}$ and $z=z_{max}$ 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 case, only one tranche is considered.

file :z Sum(u.dS)/Sum(dS) Sum(dS) Sum(u.dS)

See also: [interprete \(2\)](#)

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'*]: permet de choisir si l on veut l integrale suivant z ou sur toute la hauteur (debit_total correspond a zmin=-DMAXFLOAT, ZMax=DMAXFLOAT, nb_tranche=1)
- **zmin** *float*
- **zmax** *float*
- **nb_tranche** *int*
- **fichier_sortie** *str*: nom du fichier de sortie par default : integrale.

2.41 lata_to_med

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

See also: [interpret \(2\)](#)

Usage:

```
lata_to_med [ format ] file file_med
```

where

- **format** *format_lata_to_med* ([2.41](#)): generated file post_med.data use format (MED or MESHTV or LML keyword).
- **file** *str*: LATA file to convert to the new format.
- **file_med** *str*: Name of file med.

2.42 format_lata_to_med

Description: not_set

See also: [objet_lecture \(30\)](#)

Usage:

```
mot [ format ]
```

where

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

2.43 lata_to_other

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

See also: [interpret \(2\)](#)

Usage:

lata_to_other [**format**] **file** **file_post**

where

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

2.44 lire_ideas

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

See also: [interpret \(2\)](#)

Usage:

lire_ideas **nom_dom** **file**

where

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

2.45 mailler

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

See also: [interpret \(2\)](#)

Usage:

mailler **domaine** **bloc**

where

- **domaine** *str*: Name of domain.
- **bloc** *list_bloc_mailler* ([2.45](#)): Instructions to mesh.

2.46 list_bloc_mailler

Description: List of block mesh.

See also: [listobj \(29.2\)](#)

Usage:

{ object1 , object2 }

list of *mailler_base* ([2.46](#)) separated with ,

2.46.1 mailler_base

Description: Basic class to mesh.

See also: [objet_lecture \(30\)](#) [pave \(2.46.1\)](#) [epsilon \(2.46.11\)](#) [domain \(2.46.12\)](#)

Usage:

2.46.2 pave

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

See also: `mailler_base` (2.46)

Usage:

pave name bloc list_bord

where

- **name** *str*: Name of the pave (block).
- **bloc** *bloc_pave* (2.46.2): Definition of the pave (block).
- **list_bord** *list_bord* (2.46.3): Definition of boundaries of domain.

2.46.3 bloc_pave

Description: Class to create a pave.

See also: `objet_lecture` (30)

Usage:

```
{  
    [ Origine x1 x2 (x3)]  
    [ longueurs x1 x2 (x3)]  
    [ nombre_de_noeuds n1 n2 (n3)]  
    [ facteurs x1 x2 (x3)]  
    [ symx ]  
    [ symy ]  
    [ symz ]  
    [ tanh float]  
    [ tanh_dilatation int into [-1, 0, 1]]  
    [ tanh_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 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 discretisation 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 straight Y in 2D) passing through the block centre.
- **symy**: Keyword to define a block mesh that is symmetrical with respect to the XZ plane (respectively straight X 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.
- **tanh** *float*: Keyword to generate mesh with tanh (hyperbolic tangent) variation.

- **tanh_dilatation** *int into [-1, 0, 1]*: Keyword to generate mesh with tanh (hyperbolic tangent) variation. **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 bottom of the channel and smaller near the top -1: coarse mesh at the top of the channel and smaller near the bottom.
- **tanh_taille_premiere_maille** *float*: Size of the first cell of the mesh with tanh (hyperbolic tangent) variation in the Y direction.

2.46.4 list_bord

Description: The block sides.

See also: listobj ([29.2](#))

Usage:

```
{ object1 object2 .... }
```

list of *bord_base* ([2.46.4](#))

2.46.5 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 recognised and deleted.

See also: objet_lecture ([30](#)) bord ([2.46.5](#)) raccord ([2.46.9](#)) internes ([2.46.10](#))

Usage:

2.46.6 bord

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

See also: bord_base ([2.46.4](#))

Usage:

bord nom defbord

where

- **nom** *str*: Name of block side.
- **defbord** *defbord* ([2.46.6](#)): Definition of block side.

2.46.7 defbord

Description: Class to define an edge.

See also: objet_lecture ([30](#)) defbord_2 ([2.46.7](#)) defbord_3 ([2.46.8](#))

Usage:

2.46.8 defbord_2

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

See also: ([2.46.6](#))

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*: Value minimal.
- **inf1** *str* into ['<=']: Less or equal sign.
- **dir2** *str* into ['X', 'Y']: Edge is parallel to this direction.
- **inf2** *str* into ['<=']: Less or equal sign.
- **pos2_max** *float*: Value maximal.

2.46.9 defbord_3

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

See also: (2.46.6)

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*: Value minimal.
- **inf1** *str* into ['<=']: Less or equal sign.
- **dir2** *str* into ['X', 'Y']: Edge is parallel to this direction.
- **inf2** *str* into ['<=']: Less or equal sign.
- **pos2_max** *float*: Value maximal.
- **pos3_min** *float*: Value minimal.
- **inf3** *str* into ['<=']: Less or equal sign.
- **dir3** *str* into ['Y', 'Z']: Edge is parallel to this direction.
- **inf4** *str* into ['<=']: Less or equal sign.
- **pos3_max** *float*: Value maximal.

2.46.10 raccord

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

See also: bord_base (2.46.4)

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* (2.46.6): Definition of block side.

2.46.11 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 limitation conditions may be given 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` ([2.46.4](#))

Usage:

internes nom defbord

where

- **nom** *str*: Name of block side.
- **defbord** *defbord* ([2.46.6](#)): Definition of block side.

2.46.12 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` ([2.46](#))

Usage:

epsilon eps

where

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

2.46.13 domain

Description: Class to reuse a domain.

See also: `mailler_base` ([2.46](#))

Usage:

domain domain_name

where

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

2.47 mailerparallel

Description: creates a parallel distributed hexaedral mesh of a parallelepipedic 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: `interpret` ([2](#))

Usage:

mailerparallel {

```

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*: the 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*

2.48 **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 \(2\)](#)

Usage:

modif_bord_to_raccord **domaine** **nom_bord**

where

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

2.49 **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 \(2\)](#)

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 postraitements) 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 (2.37)*: 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 σ 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 .

2.50 option_vdf

Description: Class of VDF options.

See also: [interpret \(2\)](#)

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).
- **p_imposee_aux_faces** *str* into ['oui', 'non']: Pressure imposed at the faces (yes or no).

2.51 orientefacesbord

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

See also: [interpret \(2\)](#)

Usage:

```
orientefacesbord domain_name  
where
```

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

2.52 partition

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

See also: [interpret \(2\)](#)

Usage:

partition domaine bloc_decouper
where

- **domaine** *str*: Name of the domain to be cut.
- **bloc_decouper** *bloc_decouper* ([2.52](#)): Description how to cut a domain.

2.53 bloc_decouper

Description: Auxiliary class to cut a domain.

See also: [objet_lecture \(30\)](#)

Usage:

```
{  
  
    [ Partition_toolpartitionneur partitionneur_deriv]  
    [ larg_joint int]  
    [ zones_namelnom_zones str]  
    [ ecrire_decoupage str]  
    [ ecrire_lata str]  
    [ nb_parts_tot int]  
    [ formatte ]  
    [ periodique n word1 word2 ... wordn]  
    [ reorder int]  
  
}
```

where

- **Partition_tool**partitionneur *partitionneur_deriv* ([21](#)): 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_namelnom_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 disc (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 disc in the specified filename. See also partitionneur Fichier_Decoupage. This keyword is useful to change the partition numbers (for example, to do manually the task of the keyword Echange_domcut): 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 partitioning 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.
- **formatte** : Optional keyword to have formatted format for .Zones files. By default, it is binary format.
- **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 partitioning 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 slightly improves parallel performance.

2.54 pilote_icoco

Description: not_set

See also: [interpret \(2\)](#)

Usage:

```
pilote_icoco {
    pb_name str
    main str
}
```

where

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

2.55 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: [interpret \(2\)](#)

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* ([2.55](#)): Surface and volume porosity values.

2.56 bloc_lecture_poro

Description: Surface and volume porosity values.

See also: `objet_lecture` (30)

Usage:

```
{  
    volumique float  
    surfactive n x1 x2 ... xn  
}  
where
```

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

2.57 porosites_champ

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

Keyword Discretiser should have already be used to read the object.

See also: `interpret` (2)

Usage:

```
porosites_champ pb ch  
where
```

- **pb** *str*: Name of the problem to which the sub-area is attached.
- **ch** *champ_base* (14): field used to define the porosity field

2.58 postraiter_domaine

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

See also: `interpret` (2)

Usage:

```
postraiter_domaine {  
    format str into ['lml', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med']  
    [ filefichier 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', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med']: File format.
- **filefichier** *str*: The file name can be changed with the `fichier` option.

- **domaine** *str*: Name of domain
- **domaines** *bloc_lecture* (2.37): 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)

2.59 precisiongeom

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

See also: [interpret \(2\)](#)

Usage:

precisiongeom precision
where

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

2.60 raffiner_anisotrope

Description: To allows to cut triangle or tetrahedra elements respectively in 3 or 4 new ones by defining a new summit located at the center of the element. Note that such a cut creates flat elements (anisotropic).

See also: [interpret \(2\)](#)

Usage:

raffiner_anisotrope domain_name
where

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

2.61 raffiner_isotrope

Description: To 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).

See also: [interpret \(2\)](#)

Usage:

raffiner_isotrope domain_name
where

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

2.62 read

Description: Interpreter to read the object objet defined between the braces.

See also: [interpret](#) (2)

Usage:

read a_object bloc

where

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

2.63 read_file

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 lire_fichier 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: [interpret](#) (2) [read_unsupported_ascii_file_from_icem](#) (2.65) [read_file_binary](#) (2.63)

Usage:

read_file name_obj filename

where

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

2.64 read_file_binary

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

See also: [read_file](#) (2.62)

Usage:

read_file_binary name_obj filename

where

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

2.65 lire_tgrid

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

See also: [interpret](#) (2)

Usage:

lire_tgrid dom filename

where

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

2.66 read_unsupported_ascii_file_from_icem

Description: not_set

See also: read_file ([2.62](#))

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.

2.67 read_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 Lire_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 Lire_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 Lire_Med keyword) something like:

Lire_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: interpret ([2](#))

Usage:

read_med [**vef**] [**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.
- **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 shorty_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

2.68 orienter_simplexes

Description: Keyword to raffine a mesh

See also: [interpret \(2\)](#)

Usage:

orienter_simplexes domain_name

where

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

2.69 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: [interpret \(2\)](#)

Usage:

redresser_hexaedres_vdf domain_name

where

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

2.70 regroupebord

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

See also: [interpret \(2\)](#)

Usage:

regroupebord domaine new_bord bords

where

- **domaine** *str*: Name of domain
- **new_bord** *str*: Name of the new boundary
- **bords** *bloc_lecture (2.37)*: { Bound1 Bound2 }

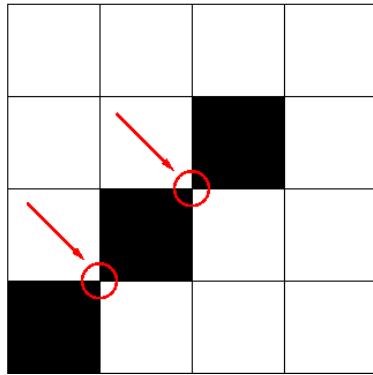
2.71 remove_elem

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):

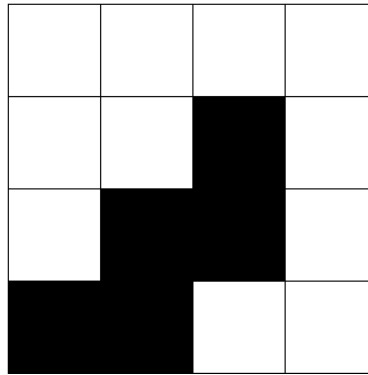
Remove_elem dom { fonction $0.2 * 0.2 - (x - 0.5)^2 - (y - 0.5)^2 > 0$ }

Warning : the thickness of removed zone has to be large enough to avoid singular nodes as decribed below

UNCORRECT – 2 SINGULAR NODES



CORRECT



See also: [interpret \(2\)](#)

Usage:

remove_elem **domaine** **bloc**

where

- **domaine** *str*: Name of domain
- **bloc** *remove_elem_bloc* ([2.71](#))

2.72 remove_elem_bloc

Description: not_set

See also: [objet_lecture \(30\)](#)

Usage:

```
{
    [ liste  n n1 n2 ... nn]
    [ fonction  str]
}
```

where

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

2.73 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: [interpret \(2\)](#)

Usage:

remove_invalid_internal_boundaries **domain_name**

where

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

2.74 reordonner_faces_periodiques

Description: The Reordonner_faces_periodiques keyword is mandatory to first define the periodic boundaries and also to reorder the faces of these boundaries.

See also: [interpret \(2\)](#)

Usage:

reordonner_faces_periodiques **domaine** **nom_bord_perio**

where

- **domaine** *str*: Name of domain.
- **nom_bord_perio** *str*: boundary_name.

2.75 reorienter_tetraedres

Description: This keyword is mandatory for front-tracking computations with the VEF discretisation. 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: [interpret \(2\)](#)

Usage:

reorienter_tetraedres **domain_name**

where

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

2.76 reorienter_triangles

Description: not_set

See also: [interpret \(2\)](#)

Usage:

reorienter_triangles **domain_name**

where

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

2.77 reordonner

Description: The Reordonner interpreter is required sometimes for a VDF mesh which is not produced by the internal mesher. Example where this is used:

Lire_Fichier 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: [interpret \(2\)](#)

Usage:

reordonner **domain_name**

where

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

2.78 rotation

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

See also: [interpret](#) (2)

Usage:

rotation **domain_name** **dir** **coord1** **coord2** **angle**

where

- **domain_name** *str*: Name of domain to which 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)

2.79 scatter

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

See also: [interpret](#) (2) [scattermed](#) (2.79)

Usage:

scatter **file** **domaine**

where

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

2.80 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](#) (2.78)

Usage:

scattermed **file** **domaine**

where

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

2.81 solve

Description: Interpreter to start calculation with TRUST.

Keyword Discretiser should have already be used to read the object.

See also: [interpret \(2\)](#)

Usage:

solve pb

where

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

2.82 supprime_bord

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

See also: [interpret \(2\)](#)

Usage:

supprime_bord domaine bords

where

- **domaine** *str*: Name of domain
- **bords** *list_nom* ([2.82](#)): { Boundary_name1 Boundaray_name2 }

2.83 list_nom

Description: List of name.

See also: [listobj \(29.2\)](#)

Usage:

{ object1 object2 }

list of *nom_anonyme* ([20](#))

2.84 system

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

See also: [interpret \(2\)](#)

Usage:

system cmd

where

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

2.85 test_solveur

Description: To test several solvers

See also: [interprete \(2\)](#)

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* (8.11): 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 $\|Ax-B\| < \text{precision}$
- **pas_de_solution_initiale** : Resolution isn't initialized with the solution x
- **ascii** : Ascii files

2.86 testeur

Description: not_set

See also: [interprete \(2\)](#)

Usage:

```
testeur data  
where
```

- **data** *bloc_lecture* (2.37)

2.87 testeur_medcoupling

Description: not_set

See also: [interprete \(2\)](#)

Usage:

testeur_medcoupling pb_name field_name

where

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

2.88 tetraedriser

Description: To achieve a tetrahedral mesh based on a mesh comprising blocks, the Tetraedriser (Tetrahe-dralise) interpreter is used in VEF discretisation.

See also: [interpret \(2\)](#) [tetraedriser_homogene \(2.88\)](#) [tetraedriser_homogene_fin \(2.90\)](#) [tetraedriser_homogene_compact \(2.89\)](#) [tetraedriser_par_prisme \(2.91\)](#)

Usage:

tetraedriser domain_name

where

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

2.89 tetraedriser_homogene

Description: Use the Tetraedriser_homogene (Homogeneous_Tetrahedralisation) interpreter in VEF discretisation to mesh a block in tetrahedrals. Each block hexahedral is no longer divided into 5 tetrahedrals (keyword Tetraedriser (Tetrahdralise)), 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 discretisation items to be avoided.

See also: [tetraedriser \(2.87\)](#)

Usage:

tetraedriser_homogene domain_name

where

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

2.90 tetraedriser_homogene_compact

Description: This new discretisation 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.

See also: [tetraedriser \(2.87\)](#)

Usage:

tetraedriser_homogene_compact domain_name

where

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

2.91 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).

See also: tetraedriser ([2.87](#))

Usage:

tetraedriser_homogene_fin domain_name

where

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

2.92 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).

See also: tetraedriser ([2.87](#))

Usage:

tetraedriser_par_prisme domain_name

where

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

2.93 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: interpret ([2](#))

Usage:

transformer domain_name formule

where

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

Function_forz

2.94 **triangler**

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

See also: interpret (2) [triangler_h \(2.95\)](#) [triangler_fin \(2.94\)](#)

Usage:

triangler domain_name

where

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

2.95 **triangler_fin**

Description: Triangler_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 discretisation PreP1B). - a better isotropy of elements than with Triangler_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 realise statistical analysis in plan channel configuration for instance).

See also: triangler (2.93)

Usage:

triangler_fin domain_name

where

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

2.96 **triangler_h**

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

See also: triangler (2.93)

Usage:

triangler_h domain_name

where

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

2.97 **verifier_qualite_raffinements**

Description: not_set

See also: interpret (2)

Usage:

verifier_qualite_raffinements domain_names

where

- **domain_names** *vect_nom* (2.97)

2.98 vect_nom

Description: Vect of name.

See also: listobj ([29.2](#))

Usage:

n object1 object2

list of *nom_anonyme* ([20](#))

2.99 verifier_simplexes

Description: Keyword to raffine a simplexes

See also: interpret ([2](#))

Usage:

verifier_simplexes domain_name

where

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

2.100 verifiercoin

Description: This keyword subdivides inconsistent 2D/3D cells used with VEFPreP1B discretization. Must be used before the mesh is discretized. NL he lire_fichier 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: interpret ([2](#))

Usage:

verifiercoin dom

where

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

2.101 ecrire

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

See also: interpret ([2](#))

Usage:

ecrire name_obj

where

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

2.102 ecrire_fichier_bin

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: [interpret](#) (2) [ecrire_fichier_formatte](#) (2.19)

Usage:

ecrire_fichier_bin **name_obj** **filename**

where

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

2.103 ecrire_med

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

See also: [interpret](#) (2)

Usage:

ecrire_med **nom_dom** **file**

where

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

3 pb_gen_base

Description: Basic class for problems.

See also: [objet_u](#) (31) [Pb_base](#) (3) [probleme_couple](#) (3.6) [pbc_med](#) (3.30)

Usage:

3.1 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) interpreter is used with a data block.

Keyword Discretiser should have already be used to read the object.

See also: [pb_gen_base](#) (3) [pb_thermohydraulique](#) (3.18) [pb_hydraulique](#) (3.11) [pb_hydraulique_turbulent](#) (3.16) [pb_thermohydraulique_turbulent](#) (3.26) [pb_conduction](#) (3.10) [pb_thermohydraulique_qc](#) (3.23) [pb_thermohydraulique_turbulent_qc](#) (3.27) [pb_hydraulique_concentration](#) (3.12) [pb_hydraulique_concentration_turbulent](#) (3.14) [pb_thermohydraulique_concentration](#) (3.19) [pb_thermohydraulique_concentration_turbulent](#) (3.21) [pb_avec_passif](#) (3.8) [pb_post](#) (3.17) [problem_read_generic](#) (3.32.1)

Usage:

Pb_base **obj** Lire **obj** {

```
[ 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* (3.1): One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28): List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1): This
- **liste_postraitements** *liste_post* (3.4.3): 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* (3.5.3): 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3): The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3): Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, 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* (3.5.3): Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.2 corps_postraitement

Description: not_set

See also: post_processing (3.4.2)

Usage:

```

{
  [ definition_champs definition_champs]
  [ Probes|sondes sondes]
  [ domaine str]
  [ format str into ['lml', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med']]
  [ fields|champs champs_posts]
  [ statistiques stats_posts]
  [ fichier str]
  [ statistiques_en_serie stats_serie_posts]
  [ interfaces champs_posts]
}
where

```

- **definition_champs** *definition_champs* (3.2) for inheritance: Keyword to create new or more complex field for advanced postprocessing.
- **Probes/sondes** *sondes* (3.2.2) 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', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med'] for inheritance: This optional parameter specifies the format of the output file. The basename used for the output file is the base-name of the data file. For the fmt parameter, choices are lml, lata, or meshtv. A short description of each format can be found below. The default value is lml.
- **fields/champs** *champs_posts* (3.2.16) for inheritance: Field's write mode.
- **statistiques** *stats_posts* (3.2.19) 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* (3.2.27) for inheritance: Statistics between two points not fixed : on period of integration.
- **interfaces** *champs_posts* (3.2.16) for inheritance: Keyword to read all the characteristics of the interfaces. Different kind of interfaces exist as well as different interface initialisations.

3.2.1 definition_champs

Description: List of definition champ

See also: listobj (29.2)

Usage:

```
{ object1 object2 .... }
```

list of *definition_champ* (3.2.1)

3.2.2 definition_champ

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

See also: objet_lecture (30)

Usage:

name **champ_generique**

where

- **name** *str*: The name of the new created field.
- **champ_generique** *champ_generique_base* (6)

3.2.3 sondes

Description: List of probes.

See also: listobj (29.2)

Usage:

```
{ object1 object2 .... }
```

list of *sonde* (3.2.3)

3.2.4 sonde

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

See also: [objet_lecture \(30\)](#)

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* [*'chsom'*, *'nodes'*, *'grav'*, *'som'*]: 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.
- **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* (3.2.4): Type of probe.

3.2.5 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 \(30\)](#) [points \(3.2.5\)](#) [numero_elem_sur_maitre \(3.2.9\)](#) [position_like \(3.2.10\)](#) [segment \(3.2.11\)](#) [plan \(3.2.12\)](#) [volume \(3.2.13\)](#) [circle \(3.2.14\)](#) [circle_3 \(3.2.15\)](#)

Usage:

sonde_base

3.2.6 points

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

See also: [sonde_base \(3.2.4\)](#) [point \(3.2.7\)](#) [segmentpoints \(3.2.8\)](#)

Usage:

points points
where

- **points** *listpoints* (3.2.6): Probe points.

3.2.7 listpoints

Description: Points.

See also: listobj ([29.2](#))

Usage:

n object1 object2

list of *un_point* ([2.7.2](#))

3.2.8 point

Description: Point as class-daughter of Points.

See also: points ([3.2.5](#))

Usage:

point points

where

- **points** *listpoints* ([3.2.6](#)): Probe points.

3.2.9 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 ([3.2.5](#))

Usage:

segmentpoints points

where

- **points** *listpoints* ([3.2.6](#)): Probe points.

3.2.10 numero_elem_sur_maitre

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

See also: sonde_base ([3.2.4](#))

Usage:

numero_elem_sur_maitre numero

where

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

3.2.11 position_like

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

See also: sonde_base ([3.2.4](#))

Usage:

position_like **autre_sonde**

where

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

3.2.12 segment

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

See also: **sonde_base** ([3.2.4](#))

Usage:

segment **nbr** **point_deb** **point_fin**

where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* ([2.7.2](#)): First outer probe segment point.
- **point_fin** *un_point* ([2.7.2](#)): Second outer probe segment point.

3.2.13 plan

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

See also: **sonde_base** ([3.2.4](#))

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* ([2.7.2](#)): First point defining the angle. This angle should be positive.
- **point_fin** *un_point* ([2.7.2](#)): Second point defining the angle. This angle should be positive.
- **point_fin_2** *un_point* ([2.7.2](#)): Third point defining the angle. This angle should be positive.

3.2.14 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** ([3.2.4](#))

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* ([2.7.2](#)): Point of origin.
- **point_fin** *un_point* ([2.7.2](#)): Point defining the first direction (from point of origin).
- **point_fin_2** *un_point* ([2.7.2](#)): Point defining the second direction (from point of origin).
- **point_fin_3** *un_point* ([2.7.2](#)): Point defining the third direction (from point of origin).

3.2.15 circle

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

See also: `sonde_base` ([3.2.4](#))

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* ([2.7.2](#)): 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.

3.2.16 circle_3

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

See also: `sonde_base` ([3.2.4](#))

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* ([2.7.2](#)): 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.

3.2.17 champs_posts

Description: Field's write mode.

See also: `objet_lecture` ([30](#))

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.
- **fields|champs** *champs_a_post* ([3.2.17](#)): Post-processed fields.

3.2.18 champs_a_post

Description: Fields to be post-processed.

See also: listobj ([29.2](#))

Usage:

```
{ object1 object2 .... }  
list of champ_a_post (3.2.18)
```

3.2.19 champ_a_post

Description: Field to be post-processed.

See also: objet_lecture ([30](#))

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 (CHAMPPPOINT type field in the lml file). If no selection is made, localisation is set to som by default.

3.2.20 stats_posts

Description: Field's write mode.

Dt_post: This keyword is used to set the calculated statistics write period.

dt: 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 (speed)**, **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:
```

$$\begin{aligned}
& t \leq t_{\text{deb}} : \\
& \text{average: } \overline{P(t)} = 0 \\
& \text{std_deviation: } \langle P(t) \rangle = 0 \\
& \text{correlation: } \langle U(t).V(t) \rangle = 0 \\
\\
& t > t_{\text{deb}} : \\
& \text{average: } \overline{P(t)} = \frac{1}{t - t_{\text{deb}}} \int_{t_{\text{deb}}}^t P(t) dt \\
& \text{std_deviation: } \langle P(t) \rangle = \sqrt{\frac{1}{t - t_{\text{deb}}} \int_{t_{\text{deb}}}^t [P(t) - \overline{P(t)}]^2 dt} \\
& \text{correlation: } \langle U(t).V(t) \rangle = \frac{1}{t - t_{\text{deb}}} \int_{t_{\text{deb}}}^t [U(t) - \overline{U(t)}] \cdot [V(t) - \overline{V(t)}] dt
\end{aligned}$$

See also: [objet_lecture \(30\)](#)

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.
- **fields|champs** *list_stat_post* (3.2.20): Post-processed fields.

3.2.21 list_stat_post

Description: Post-processing for statistics

See also: [listobj \(29.2\)](#)

Usage:

{ object1 object2 }

list of *stat_post_deriv* (3.2.21)

3.2.22 stat_post_deriv

Description: not_set

See also: [objet_lecture \(30\)](#) [t_deb \(3.2.22\)](#) [t_fin \(3.2.23\)](#) [moyenne \(3.2.24\)](#) [ecart_type \(3.2.25\)](#) [correlation \(3.2.26\)](#)

Usage:

stat_post_deriv

3.2.23 t_deb

Description: not_set

See also: [stat_post_deriv \(3.2.21\)](#)

Usage:

t_deb **val**

where

- **val** *float*

3.2.24 t_fin

Description: not_set

See also: stat_post_deriv ([3.2.21](#))

Usage:

t_fin **val**

where

- **val** *float*

3.2.25 moyenne

Description: not_set

See also: stat_post_deriv ([3.2.21](#))

Usage:

moyenne **field** [**localisation**]

where

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

3.2.26 ecart_type

Description: not_set

See also: stat_post_deriv ([3.2.21](#))

Usage:

ecart_type **field** [**localisation**]

where

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

3.2.27 correlation

Description: not_set

See also: stat_post_deriv ([3.2.21](#))

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

3.2.28 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 (speed)**, **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:

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

See also: [objet_lecture \(30\)](#)

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* ([3.2.20](#))

3.3 post_processings

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

See also: [listobj \(29.2\)](#)

Usage:

```
{ object1 object2 .... }
```

list of *un_postraitement* ([3.3](#))

3.3.1 un_postraitement

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

See also: objet_lecture (30)

Usage:

nom post

where

- **nom** *str*: Name of the post-processing.
- **post** *corps_postraitement* (3.1): Definition of the post-processing.

3.4 liste_post_ok

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

See also: listobj (29.2)

Usage:

{ object1 object2 }

list of *nom_postraitement* (3.4)

3.4.1 nom_postraitement

Description:

See also: objet_lecture (30)

Usage:

nom post

where

- **nom** *str*: Name of the post-processing.
- **post** *postraitement_base* (3.4.1): the post

3.4.2 postraitement_base

Description: not_set

See also: objet_lecture (30) post_processing (3.4.2)

Usage:

3.4.3 post_processing

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

See also: postraitement_base (3.4.1) corps_postraitement (3.1)

Usage:

post_processing {

[**definition_champs** *definition_champs*]

```

[ Probes|sondes sondes]
[ domaine str]
[ format str into ['lml', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med']]
[ fields|champs champs_posts]
[ statistiques stats_posts]
[ fichier str]
[ statistiques_en_serie stats_serie_posts]
[ interfaces champs_posts]
}
where

```

- **definition_champs** *definition_champs* (3.2): Keyword to create new or more complex field for advanced postprocessing.
- **Probes|sondes** *sondes* (3.2.2): 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', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med']: 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, lata, or meshtv. A short description of each format can be found below. The default value is lml.
- **fields|champs** *champs_posts* (3.2.16): Field's write mode.
- **statistiques** *stats_posts* (3.2.19): 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* (3.2.27): Statistics between two points not fixed : on period of integration.
- **interfaces** *champs_posts* (3.2.16): Keyword to read all the characteristics of the interfaces. Different kind of interfaces exist as well as different interface initialisations.

3.5 liste_post

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

See also: listobj (29.2)

Usage:

{ object1 object2 }

list of *un_postraitement_spec* (3.5)

3.5.1 un_postraitement_spec

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

See also: objet_lecture (30)

Usage:

[**type_un_post**] [**type_postraitement_ft_lata**]

where

- **type_un_post** *type_un_post* (3.5.1)
- **type_postraitement_ft_lata** *type_postraitement_ft_lata* (3.5.2)

3.5.2 type_un_post

Description: not_set

See also: objet_lecture (30)

Usage:

type post

where

- **type** *str* into ['postraitement', 'post_processing']
- **post** *un_postraitement* (3.3)

3.5.3 type_postraitement_ft_lata

Description: not_set

See also: objet_lecture (30)

Usage:

type nom bloc

where

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

3.6 format_file

Description: File formatted.

See also: objet_lecture (30)

Usage:

[**format**] **name_file**

where

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

3.7 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 Associer keyword or with the Lire/groupe 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

Lire 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 (3)

Usage:

probleme_couple obj Lire obj {

 [**groupes** *list_list_nom*]

}

where

- **groupes** *list_list_nom* (3.7): { groupes { { pb1 , pb2 } , { pb3 , pb4 } } }

3.8 list_list_nom

Description: pour les groupes

See also: listobj (29.2)

Usage:

{ object1 , object2 }

list of *list_un_pb* (29) separated with ,

3.9 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 Discretiser should have already be used to read the object.

See also: Pb_base (3) pb_thermohydraulique_concentration_turbulent_scalaires_passifs (3.22) pb_thermohydraulique_concentration_scalaires_passifs (3.20) pb_thermohydraulique_turbulent_scalaires_passifs (3.29) pb_thermohydraulique_scalaires_passifs (3.25) pb_hydraulique_concentration_turbulent_scalaires_passifs (3.15) pb_hydraulique_concentration_scalaires_passifs (3.13) pb_thermohydraulique_qc_fraction_massique (3.24) pb_thermohydraulique_turbulent_qc_fraction_massique (3.28)

Usage:

pb_avec_passif obj Lire obj {

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* (3.9): 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* (3.1) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.10 listeqn

Description: List of equations.

See also: listobj (29.2)

Usage:

```
{ object1 object2 .... }
```

list of *eqn_base* (4.17)

3.11 pb_conduction

Description: Resolution of the heat equation.

Keyword Discretiser should have already be used to read the object.

See also: Pb_base (3)

Usage:

```
pb_conduction obj Lire obj {
```

```
    [ 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* (4): Heat equation.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.12 pb_hydraulique

Description: Resolution of the NAVIER STOKES equations.

Keyword Discretiser should have already be used to read the object.

See also: Pb_base (3)

Usage:

```

pb_hydraulique obj Lire obj {

    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* (4.22.6): NAVIER STOKES equations.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.13 pb_hydraulique_concentration

Description: Resolution of NAVIER STOKES/multiple constituent transportation equations.

Keyword Discretiser should have already be used to read the object.

See also: Pb_base (3)

Usage:

```

pb_hydraulique_concentration obj Lire obj {
    [ 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* (4.22.6): NAVIER STOKES equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (4.10): Constituent transportation vectorial equation (concentration diffusion convection).
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.14 pb_hydraulique_concentration_scalaires_passifs

Description: Resolution of NAVIER STOKES/multiple constituent transportation equations with the additional passive scalar equations.

Keyword Discretiser should have already be used to read the object.

See also: pb_avec_passif (3.8)

Usage:

```

pb_hydraulique_concentration_scalaires_passifs obj Lire obj {
    [ 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* (4.22.6): NAVIER STOKES equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (4.10): Constituent transportation equations (concentration diffusion convection).
- **equations_scalaires_passifs** *listeqn* (3.9) 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* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \neq P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.15 pb_hydraulique_concentration_turbulent

Description: Resolution of NAVIER STOKES/multiple constituent transportation equations, with turbulence modelling.

Keyword Discretiser should have already be used to read the object.

See also: Pb_base (3)

Usage:

pb_hydraulique_concentration_turbulent obj Lire obj {

```

[ navier_stokes_turbulent navier_stokes_turbulent]
[ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent]
[ 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_turbulent** *navier_stokes_turbulent* (4.23): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (4.11): Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.16 pb_hydraulique_concentration_turbulent_scalaires_passifs

Description: Resolution of NAVIER STOKES/multiple constituent transportation equations, with turbulence modelling and with the additional passive scalar equations.

Keyword Discretiser should have already be used to read the object.

See also: `pb_avec_passif` (3.8)

Usage:

```
pb_hydraulique_concentration_turbulent_scalaires_passifs obj Lire obj {  
    [ navier_stokes_turbulent navier_stokes_turbulent ]  
    [ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent ]  
    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_turbulent** *navier_stokes_turbulent* (4.23): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (4.11): Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (3.9) 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* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved

files).

3.17 pb_hydraulique_turbulent

Description: Resolution of NAVIER STOKES equations with turbulence modelling.

Keyword Discretiser should have already be used to read the object.

See also: Pb_base (3)

Usage:

```
pb_hydraulique_turbulent obj Lire obj {  
    navier_stokes_turbulent navier_stokes_turbulent  
    [ 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_turbulent** *navier_stokes_turbulent* (4.23): NAVIER STOKES equations as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.18 pb_post

Description: not_set

Keyword Discretiser should have already be used to read the object.

See also: Pb_base (3)

Usage:

```
pb_post obj Lire obj {  
    [ 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* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.19 pb_thermohydraulique

Description: Resolution of thermohydraulic problem.

Keyword Discretiser should have already be used to read the object.

See also: Pb_base (3)

Usage:

```
pb_thermohydraulique obj Lire obj {  
    [ 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* (4.22.6): NAVIER STOKES equations.
- **convection_diffusion_temperature** *convection_diffusion_temperature* (4.14): Energy equation (temperature diffusion convection).
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.20 pb_thermohydraulique_concentration

Description: Resolution of NAVIER STOKES/energy/multiple constituent transportation equations.

Keyword Discretiser should have already be used to read the object.

See also: Pb_base (3)

Usage:

```
pb_thermohydraulique_concentration obj Lire obj {  
    [ 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* (4.22.6): NAVIER STOKES equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (4.10): Constituent transportation equations (concentration diffusion convection).
- **convection_diffusion_temperature** *convection_diffusion_temperature* (4.14): Energy equation (temperature diffusion convection).
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.21 pb_thermohydraulique_concentration_scalaires_passifs

Description: Resolution of NAVIER STOKES/energy/multiple constituent transportation equations, with the additional passive scalar equations.

Keyword Discretiser should have already be used to read the object.

See also: pb_avec_passif (3.8)

Usage:

```
pb_thermohydraulique_concentration_scalaires_passifs obj Lire obj {  
    [ 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* (4.22.6): NAVIER STOKES equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (4.10): Constituent transportation equations (concentration diffusion convection).
- **convection_diffusion_temperature** *convection_diffusion_temperature* (4.14): Energy equations (temperature diffusion convection).
- **equations_scalaires_passifs** *listeqn* (3.9) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_masseN. 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* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.22 pb_thermohydraulique_concentration_turbulent

Description: Resolution of NAVIER STOKES/energy/multiple constituent transportation equations, with turbulence modelling.

Keyword Discretiser should have already be used to read the object.

See also: Pb_base (3)

Usage:

```
pb_thermohydraulique_concentration_turbulent obj Lire obj {
    [ navier_stokes_turbulent navier_stokes_turbulent ]
    [ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent ]
    [ convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent ]
    [ 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_turbulent** *navier_stokes_turbulent* (4.23): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (4.11): Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (4.16.1): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.23 pb_thermohydraulique_concentration_turbulent_scalaires_passifs

Description: Resolution of NAVIER STOKES/energy/multiple constituent transportation equations, with turbulence modelling and with the additional passive scalar equations.

Keyword Discretiser should have already be used to read the object.

See also: pb_avec_passif (3.8)

Usage:

```
pb_thermohydraulique_concentration_turbulent_scalaires_passifs obj Lire obj {
    [ navier_stokes_turbulent navier_stokes_turbulent]
    [ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent]
    [ convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent]
    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_turbulent** *navier_stokes_turbulent* (4.23): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (4.11): Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (4.16.1): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (3.9) 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* (3.1) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.24 pb_thermohydraulique_qc

Description: Resolution of thermohydraulic problem under small Mach number.

Keywords for the unknowns other than pressure, velocity, temperature are :

masse_volumique : density

enthalpie : enthalpy

pression : reduced pressure

pression_tot : total pressure.

Keyword Discretiser should have already be used to read the object.

See also: Pb_base (3)

Usage:

```
pb_thermohydraulique_qc obj Lire obj {
    navier_stokes_qc navier_stokes_qc
    convection_diffusion_chaleur_qc convection_diffusion_chaleur_qc
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processing|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* (4.18): NAVIER STOKES equations under small Mach number.
- **convection_diffusion_chaleur_qc** *convection_diffusion_chaleur_qc* (4.7.2): Energy equation under small Mach number.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.25 pb_thermohydraulique_qc_fraction_massique

Description: Resolution of thermohydraulic problem under small Mach number with passive scalar equations.

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

See also: pb_avec_passif (3.8)

Usage:

```
pb_thermohydraulique_qc_fraction_massique obj Lire obj {
    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* (4.18): NAVIER STOKES equations under small Mach number.
- **convection_diffusion_chaleur_qc** *convection_diffusion_chaleur_qc* (4.7.2): Energy equation under small Mach number.
- **equations_scalaires_passifs** *listeqn* (3.9) 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* (3.1) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.26 pb_thermohydraulique_scalaires_passifs

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

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

See also: pb_avec_passif (3.8)

Usage:

```
pb_thermohydraulique_scalaires_passifs obj Lire obj {
```

```

[ 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* (4.22.6): NAVIER STOKES equations.
- **convection_diffusion_temperature** *convection_diffusion_temperature* (4.14): Energy equations (temperature diffusion convection).
- **equations_scalaires_passifs** *listeqn* (3.9) 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* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.27 pb_thermohydraulique_turbulent

Description: Resolution of thermohydraulic problem, with turbulence modelling.

Keyword Discretiser should have already be used to read the object.

See also: Pb_base (3)

Usage:

```
pb_thermohydraulique_turbulent obj Lire obj {  
  
    navier_stokes_turbulent navier_stokes_turbulent  
    convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent  
    [ 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_turbulent** *navier_stokes_turbulent* (4.23): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (4.16.1): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N <> P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.28 pb_thermohydraulique_turbulent_qc

Description: Resolution of turbulent thermohydraulic problem under small Mach number.

Warning : Available for VDF and VEF P0/P1NC discretization only.

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

See also: Pb_base (3)

Usage:

```
pb_thermohydraulique_turbulent_qc obj Lire obj {  
    navier_stokes_turbulent_qc navier_stokes_turbulent_qc  
    convection_diffusion_chaleur_turbulent_qc convection_diffusion_chaleur_turbulent_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_turbulent_qc** *navier_stokes_turbulent_qc* (4.25.11): NAVIER STOKES equations under small Mach number as well as the associated turbulence model equations.
- **convection_diffusion_chaleur_turbulent_qc** *convection_diffusion_chaleur_turbulent_qc* (4.9.22): Energy equation under small Mach number as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.29 pb_thermohydraulique_turbulent_qc_fraction_massique

Description: Resolution of turbulent thermohydraulic problem under small Mach number with passive scalar equations.

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

See also: pb_avec_passif (3.8)

Usage:

```
pb_thermohydraulique_turbulent_qc_fraction_massique obj Lire obj {  
    navier_stokes_turbulent_qc navier_stokes_turbulent_qc  
    convection_diffusion_chaleur_turbulent_qc convection_diffusion_chaleur_turbulent_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_turbulent_qc** *navier_stokes_turbulent_qc* (4.25.11): NAVIER STOKES equations under small Mach number as well as the associated turbulence model equations.
- **convection_diffusion_chaleur_turbulent_qc** *convection_diffusion_chaleur_turbulent_qc* (4.9.22): Energy equation under small Mach number as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (3.9) 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* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the

calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.30 pb_thermohydraulique_turbulent_scalaires_passifs

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

Keyword Discretiser should have already be used to read the object.

See also: pb_avec_passif (3.8)

Usage:

```
pb_thermohydraulique_turbulent_scalaires_passifs obj Lire obj {
    [ navier_stokes_turbulent navier_stokes_turbulent]
    [ convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent]
    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_turbulent** *navier_stokes_turbulent* (4.23): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (4.16.1): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (3.9) 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* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.31 pbc_med

Description: Permet de relire des fichiers meds et de les postraiter.

See also: pb_gen_base (3)

Usage:

pbc_med list_info_med

where

- **list_info_med** *list_info_med* (3.31)

3.32 list_info_med

Description: not_set

See also: listobj (29.2)

Usage:

{ object1 , object2 }

list of *info_med* (3.32) separated with ,

3.32.1 info_med

Description: not_set

See also: objet_lecture (30)

Usage:

file_med domaine pb_post

where

- **file_med** *str*: Name of file med.
- **domaine** *str*: Name of domain.
- **pb_post** *pb_post* (3.17)

3.33 problem_read_generic

Description: The `probleme_read_generic` differs from 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 `Associer` keyword.

Keyword `Discretiser` should have already been used to read the object.

See also: `Pb_base` (3)

Usage:

```
problem_read_generic obj Lire obj {  
    [ 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* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.3) 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* (3.5.3) 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 restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than `Sauvegarde` except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart 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 restart a parallel calculation on `P` processors, whereas the previous calculation has been run on `N` ($N < > P$) processors. Should the calculation be restarted, 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* (3.5.3) for inheritance: Keyword to restart a calculation based on the `name_file` file, restart the calculation at the last time found in the file (`tinit` is set to last time of saved files).

4 mor_eqn

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

See also: `objet_u` (31) `eqn_base` (4.17)

Usage:

4.1 conduction

Description: Heat equation.

Keyword Discretiser should have already be used to read the object.

See also: `eqn_base` (4.17)

Usage:

```
conduction obj Lire obj {  
    [ 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
```

- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.6.2) 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 is not solved between time t0 and t1.

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

4.2 bloc_diffusion

Description: not_set

See also: objet_lecture (30)

Usage:

aco [**opérateur**] [**op_implicite**] **acof**

where

- **aco** *str into* ['']: Open accodance sign.
- **opérateur** *diffusion_deriv* (4.2): if none is specified, the diffusive scheme used is an order 2 scheme.
- **op_implicite** *op_implicite* (4.2.8): To have diffusive implicitation, it use Uzawa algorithm. Very useful when viscosity has large variations.
- **acof** *str into* ['']: Closed accodance sign.

4.2.1 diffusion_deriv

Description: not_set

See also: objet_lecture (30) negligeable (4.2.1) p1b (4.2.2) p1ncp1b (4.2.3) stab (4.2.4) standard (4.2.5) option (4.2.7)

Usage:

diffusion_deriv

4.2.2 negligeable

Description: the diffusivity will not taken in count

See also: diffusion_deriv (4.2)

Usage:

negligeable

4.2.3 p1b

Description: not_set

See also: diffusion_deriv (4.2)

Usage:

p1b

4.2.4 p1ncp1b

Description: not_set

See also: diffusion_deriv (4.2)

Usage:

4.2.5 stab

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

See also: [diffusion_deriv \(4.2\)](#)

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*

4.2.6 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 \(4.2\)](#)

Usage:

```
standard [ mot1 ] [ bloc_diffusion_standard ]  
where
```

- **mot1** *str into ['default_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 (4.2.6)*

4.2.7 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` (30)

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]

4.2.8 option

Description: `not_set`

See also: `diffusion_deriv` (4.2)

Usage:

option bloc_lecture

where

- **bloc_lecture** *bloc_lecture* (2.37)

4.2.9 op_implicite

Description: `not_set`

See also: `objet_lecture` (30)

Usage:

implicite mot solveur

where

- **implicite** *str* into ['implicite']
- **mot** *str* into ['solveur']
- **solveur** *solveur_sys_base* (8.11)

4.3 condinits

Description: Initial conditions.

See also: `objet_lecture` (30)

Usage:

aco condinit acof

where

- **aco** *str* into [' ']: Open accodance sign.
- **condinit** *condinit* (4.3): CI
- **acof** *str* into [' ']: Closed accodance sign.

4.3.1 condinit

Description: Initial condition.

See also: `objet_lecture` (30)

Usage:

nom ch

where

- **nom** *str*: Name of initial condition field.
- **ch** *champ_base* (14): Type field and the initial values.

4.4 condlims

Description: Boundary conditions.

See also: `listobj` (29.2)

Usage:

{ object1 object2 }

list of *condlimlu* (4.4)

4.4.1 condlimlu

Description: Boundary condition specified.

See also: `objet_lecture` (30)

Usage:

bord cl

where

- **bord** *str*: Name of the edge where the boundary condition applies.
- **cl** *condlim_base* (10): Boundary condition at the boundary called bord (edge).

4.5 sources

Description: The sources.

See also: `listobj` (29.2)

Usage:

{ object1 , object2 }

list of `source_base` (25) separated with ,

4.6 ecrire_fichier_xyz_valeur_param

Description: `not_set`

Keyword `Discretiser` should have already be used to read the object.

See also: `listobj` (29.2)

Usage:

n object1 , object2

list of `ecrire_fichier_xyz_valeur_item` (4.6) separated with ,

4.6.1 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` (30)

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* (4.6.1): to post-process only on some boundaries

4.6.2 bords_ecrire

Description: `not_set`

See also: `objet_lecture` (30)

Usage:

chaîne **bords**

where

- **chaîne** *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.

4.7 parametre_equation_base

Description: Basic class for parametre_equation

See also: objet_lecture (30) parametre_diffusion_implicite (4.7) parametre_implicite (4.7.1)

Usage:

4.7.1 parametre_diffusion_implicite

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

See also: parametre_equation_base (4.6.2)

Usage:

```
parametre_diffusion_implicite {  
    [ crank int into [0, 1]]  
    [ preconditionnement_diag int into [0, 1]]  
    [ niter_max_diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
}  
where
```

- **crank** *int into [0, 1]*: Use (1) or not (0, default) a Crank Nicholson method for the diffusion implication 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 implication of the equation diffusion operator is not preconditioned by default. If this option is set to 1, a diagonal preconditioning 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 implication of the equation.
- **seuil_diffusion_implicite** *float*: Change the threshold convergence value used by default for the CG resolution for the diffusion implication of this equation.

4.7.2 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 (4.6.2)

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_implicit** *float*: Keyword to change for this equation only the value of `seuil_convergence_implicit` 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* (8.11): 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)*).

4.8 convection_diffusion_chaleur_qc

Description: Energy equation under small Mach number.

Keyword `Discretiser` should have already been used to read the object.

See also: `eqn_base` (4.17) `convection_diffusion_chaleur_turbulent_qc` (4.9.22)

Usage:

convection_diffusion_chaleur_qc *obj Lire obj* {

```
[ mode_calcul_convection str into ['ancien', 'divuT_moins_Tdivu', 'divrhout_moins_Tdivrhout']]
[ 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_Tdivrhout']:
Option to set the form of the convective operator
 $\text{divrhout_moins_Tdivrhout}$ (the default since 1.6.8): $\rho \cdot u \cdot \text{grad} T = \text{div}(\rho \cdot u \cdot T) - T \cdot \text{div}(\rho \cdot u)$
ancien: $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \cdot \text{div}(u)$
 divuT_moins_Tdivu : $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \cdot \text{div}(u)$
- **convection** *bloc_convection* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.6.2) 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. Example: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.9 bloc_convection

Description: not_set

See also: objet_lecture (30)

Usage:

aco operateur acof
where

- **aco** *str into ['']*: Open accodance sign.
- **operateur** *convection_deriv* (4.9)
- **acof** *str into ['']*: Closed accodance sign.

4.9.1 convection_deriv

Description: not_set

See also: objet_lecture (30) *amount* (4.9.1) *amount_old* (4.9.2) *centre* (4.9.3) *centre4* (4.9.4) *centre_old* (4.9.5) *di_l2* (4.9.6) *ef* (4.9.7) *muscl3* (4.9.9) *ef_stab* (4.9.10) *generic* (4.9.13) *kquick* (4.9.14) *muscl* (4.9.15) *muscl_old* (4.9.16) *muscl_new* (4.9.17) *negligeable* (4.9.18) *quick* (4.9.19) *supg* (4.9.20) *btd* (4.9.21)

Usage:

convection_deriv

4.9.2 amount

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

See also: convection_deriv (4.9)

Usage:

amount

4.9.3 **amont_old**

Description: not_set

See also: convection_deriv (4.9)

Usage:

amont_old

4.9.4 **centre**

Description: not_set

See also: convection_deriv (4.9)

Usage:

centre

4.9.5 **centre4**

Description: not_set

See also: convection_deriv (4.9)

Usage:

centre4

4.9.6 **centre_old**

Description: not_set

See also: convection_deriv (4.9)

Usage:

centre_old

4.9.7 **di_l2**

Description: not_set

See also: convection_deriv (4.9)

Usage:

di_l2

4.9.8 **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 : $\text{div}((u \cdot \text{grad } ub, vb) - (u \cdot \text{grad } vb, ub))$, 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` (4.9)

Usage:

ef [**mot1**] [**bloc_ef**]

where

- **mot1** *str* into ['default_bar']: equivalent to `transportant_bar 0 transporte_bar 1 filtrer_resu 1 antisym 1`
- **bloc_ef** *bloc_ef* (4.9.8)

4.9.9 bloc_ef

Description: `not_set`

See also: `objet_lecture` (30)

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]

4.9.10 muscl3

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

See also: `convection_deriv` (4.9)

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).

4.9.11 ef_stab

Description: Keyword for a VEF convective scheme.

See also: [convection_deriv \(4.9\)](#)

Usage:

```
ef_stab {  
    [ alpha float]  
    [ test int]  
    [ tdivu ]  
    [ old ]  
    [ volumes_etendus ]  
    [ volumes_non_etendus ]  
    [ montant_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 advised to use alpha=1 and for the momentum equation, alpha=0.2 is advised.
- **test int**: Developer option to compare old and new version of EF_stab
- **tdivu** : To have the convective operator calculated as $\text{div}(\mathbf{TU}) - \mathbf{T} \text{div} \mathbf{U} (= \mathbf{U} \text{grad} \mathbf{T})$.
- **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).
- **montant_sous_zone str**: Option to degenerate EF_stab scheme into Amount (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 (4.9.11)**: 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.

4.9.12 listsous_zone_valeur

Description: List of groups of two words.

See also: [listobj \(29.2\)](#)

Usage:

n object1 object2

list of *sous_zone_valeur* ([4.9.12](#))

4.9.13 sous_zone_valeur

Description: Two words.

See also: [objet_lecture \(30\)](#)

Usage:

sous_zone valeur

where

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

4.9.14 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 amount }
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 (4.9)

Usage:

generic **type** [**limiteur**] [**ordre**] [**alpha**]

where

- **type** *str* into ['amount', '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

4.9.15 kquick

Description: not_set

See also: convection_deriv (4.9)

Usage:

kquick

4.9.16 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 (4.9)

Usage:

muscl

4.9.17 muscl_old

Description: not_set

See also: `convection_deriv` ([4.9](#))

Usage:

muscl_old

4.9.18 **muscl_new**

Description: `not_set`

See also: `convection_deriv` ([4.9](#))

Usage:

muscl_new

4.9.19 **negligeable**

Description: suppresses the convection operator.

See also: `convection_deriv` ([4.9](#))

Usage:

negligeable

4.9.20 **quick**

Description: `not_set`

See also: `convection_deriv` ([4.9](#))

Usage:

quick

4.9.21 **supg**

Description: `not_set`

See also: `convection_deriv` ([4.9](#))

Usage:

supg {
 [**facteur** *float*]

}

where

- **facteur** *float*

4.9.22 **btd**

Description: `not_set`

See also: `convection_deriv` ([4.9](#))

Usage:

btd {

```

    [ facteur float]
    btd float

```

}
where

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

4.10 convection_diffusion_chaleur_turbulent_qc

Description: Energy equation under small Mach number as well as the associated turbulence model equations.

Keyword Discretiser should have already been used to read the object.
See also: convection_diffusion_chaleur_qc (4.7.2)

Usage:

```

convection_diffusion_chaleur_turbulent_qc obj Lire obj {
    [ modele_turbulence modele_turbulence_scal_base]
    [ mode_calcul_convection str into ['ancien', 'divuT_moins_Tdivu', 'divrhout_moins_Tdivrhout']]
    [ 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

```

- **modele_turbulence** *modele_turbulence_scal_base* (19): Turbulence model for the energy equation.
- **mode_calcul_convection** *str* into ['ancien', 'divuT_moins_Tdivu', 'divrhout_moins_Tdivrhout']
for inheritance: Option to set the form of the convective operator
divrhout_moins_Tdivrhout (the default since 1.6.8): $\rho \cdot u \cdot \text{grad} T = \text{div}(\rho \cdot u \cdot T) - T \cdot \text{div}(\rho \cdot u)$
ancien: $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \cdot \text{div}(u)$
divuT_moins_Tdivu : $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \cdot \text{div}(u)$
- **convection** *bloc_convection* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.6.2) 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 is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.11 convection_diffusion_concentration

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

Keyword Discretiser should have already be used to read the object.

See also: eqn_base (4.17) convection_diffusion_concentration_turbulent (4.11)

Usage:

```
convection_diffusion_concentration obj Lire obj {
    [ 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* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.6.2) 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 is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.12 convection_diffusion_concentration_turbulent

Description: Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.

Keyword Discretiser should have already be used to read the object.

See also: convection_diffusion_concentration (4.10)

Usage:

```
convection_diffusion_concentration_turbulent obj Lire obj {
    [ modele_turbulence modele_turbulence_scal_base]
    [ 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

- **modele_turbulence** *modele_turbulence_scal_base* (19): Turbulence model to be used in the constituent transportation equations. The only model currently available is Schmidt.
- **nom_inconnue** *str* for inheritance: 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 useful 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* for inheritance
- **alias** *str* for inheritance
- **convection** *bloc_convection* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.6.2) 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 is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.13 convection_diffusion_fraction_massique_qc

Description: not_set

Keyword Discretiser should have already be used to read the object.

See also: eqn_base (4.17)

Usage:

convection_diffusion_fraction_massique_qc obj Lire obj {

```

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* (13)
- **convection** *bloc_convection* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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 : pbyname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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 : pbyname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.6.2) 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 is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.14 convection_diffusion_fraction_massique_turbulent_qc

Description: not_set

Keyword Discretiser should have already be used to read the object.
See also: eqn_base (4.17)

Usage:

convection_diffusion_fraction_massique_turbulent_qc obj Lire obj {

```
[ modele_turbulence modele_turbulence_scal_base]
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

- **modele_turbulence** *modele_turbulence_scal_base* (19): Turbulence model to be used.
- **espece** *espece* (13)
- **convection** *bloc_convection* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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 : pbnome_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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 : pbnome_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.6.2) 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 is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.15 convection_diffusion_temperature

Description: Energy equation (temperature diffusion convection).

Keyword Discretiser should have already be used to read the object.

See also: eqn_base (4.17)

Usage:

```
convection_diffusion_temperature obj Lire obj {
    [ 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* (4.15): 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* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.6.2) 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 is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.16 pp

Description: not_set

See also: listobj (29.2)

Usage:

```
{ object1 object2 .... }  
list of penalisation_l2_ftd_lec (4.16)
```

4.16.1 penalisation_l2_ftd_lec

Description: not_set

See also: objet_lecture (30)

Usage:

```
bord val  
where
```

- **bord** *str*
- **val** *n x1 x2 ... xn*

4.17 convection_diffusion_temperature_turbulent

Description: Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.

Keyword Discretiser should have already be used to read the object.

See also: eqn_base (4.17)

Usage:

```
convection_diffusion_temperature_turbulent obj Lire obj {
    [ modele_turbulence modele_turbulence_scal_base]
    [ 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

- **modele_turbulence** *modele_turbulence_scal_base* (19): Turbulence model for the energy equation.
- **convection** *bloc_convection* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.6.2) 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. Example: The Navier Stokes is not solved between time t0 and t1.

Navier_Sokes_Standard

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

4.18 eqn_base

Description: Basic class for equations.

Keyword Discretiser should have already be used to read the object.

See also: `mor_eqn` (4) `navier_stokes_standard` (4.22.6) `convection_diffusion_temperature` (4.14) `convection_diffusion_temperature_turbulent` (4.16.1) `conduction` (4) `convection_diffusion_chaleur_qc` (4.7.2) `transport_k_epsilon` (4.26) `convection_diffusion_concentration` (4.10) `convection_diffusion_fraction_massique_qc` (4.12) `convection_diffusion_fraction_massique_turbulent_qc` (4.13)

Usage:

eqn_base obj Lire obj {

```
[ 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* (4.8): Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1): Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9): Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1): Boundary conditions.
- **sources** *sources* (4.4.1): 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* (4.5): This keyword is used to write the values of a field for the whole domain or 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 : `pdbname_fieldname_[boundaryname]_time.dat`
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.5): This keyword is used to write the values of a field for the whole domain or 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 : `pdbname_fieldname_[boundaryname]_time.dat`

- **parametre_equation** *parametre_equation_base* (4.6.2): 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. Example: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.19 navier_stokes_qc

Description: NAVIER STOKES equations under smal Mach number.

Keyword Discretiser should have already be used to read the object.

See also: navier_stokes_standard (4.22.6)

Usage:

```
navier_stokes_qc obj Lire obj {
    [ 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]
    [ 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 first 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 equation) 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 equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.
- **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* (8.11) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (8.11) 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* (4.19) 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* (4.20) 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 $\|Ax-B\| < \text{seuil}(tn)$. For tn+1, the threshold value $\text{seuil}(tn+1)$ will be evaluated as:

```
If ( lmax(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* (4.21) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.6.2) 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. Example: The Navier Stokes is not solved between time t0 and t1.

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

4.20 deuxmots

Description: Two words.

See also: objet_lecture (30)

Usage:

mot_1 mot_2

where

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

4.21 floatfloat

Description: Two reals.

See also: [objet_lecture \(30\)](#)

Usage:

a b

where

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

4.22 traitement_particulier

Description: Auxiliary class to post-process particular values.

See also: [objet_lecture \(30\)](#)

Usage:

aco trait_part acof

where

- **aco** *str* into [' ']: Open accodance sign.
- **trait_part** *traitement_particulier_base* ([4.22](#)): Type of traitement_particulier.
- **acof** *str* into [' ']: Closed accodance sign.

4.22.1 traitement_particulier_base

Description: Basic class to post-process particular values.

See also: [objet_lecture \(30\)](#) [temperature \(4.22.1\)](#) [canal \(4.22.2\)](#) [ec \(4.22.3\)](#) [thi \(4.22.4\)](#) [chmoy_faceperio \(4.22.5\)](#)

Usage:

4.22.2 temperature

Description: not_set

See also: [traitement_particulier_base \(4.22\)](#)

Usage:

temperature {

bord *str*

direction *int*

```
}  
where
```

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

4.22.3 canal

Description: Keyword for statistics on a periodic plane channel.

See also: `traitement_particulier_base` ([4.22](#))

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 restart a calculation with previous average quantities.

Note that for thermal and turbulent problems, averages on temperature and turbulent viscosity are automatically calculated. To restart 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).

4.22.4 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` ([4.22](#))

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.

4.22.5 thi

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

See also: traitement_particulier_base ([4.22](#))

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 as kinetic energy equals to the value given by keyword val_Ec.
- **val_Ec** *float*: Keyword to impose a value for kinetic energy by velocity renormalized 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

4.22.6 `chmoy_faceperio`

Description: non documente

See also: `traitement_particulier_base` (4.22)

Usage:

`chmoy_faceperio` **`bloc`**

where

- **`bloc`** *`bloc_lecture`* (2.37)

4.23 `navier_stokes_standard`

Description: NAVIER STOKES equations.

Keyword `Discretiser` should have already be used to read the object.

See also: `eqn_base` (4.17) `navier_stokes_turbulent` (4.23) `navier_stokes_qc` (4.18)

Usage:

`navier_stokes_standard` **`obj Lire obj`** {

```
[ 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]
[ 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 first 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 equation) 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 equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.
- **`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* (8.11): Linear pressure system resolution method.

- **solveur_bar** *solveur_sys_base* (8.11): 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* (4.19): 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* (4.20): 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 t_n , the linear system $Ax=B$ is considered as solved if the residual $\|Ax-B\| < \text{seuil}(t_n)$. For t_{n+1} , the threshold value $\text{seuil}(t_{n+1})$ will be evaluated as:
 If ($\text{lmax}(\text{DivU}) * dt < \text{value}$)
 Seuil(t_{n+1}) = Seuil(t_n) * factor
 Else
 Seuil(t_{n+1}) = Seuil(t_n) * 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* (4.21): Keyword to post-process particular values.
- **convection** *bloc_convection* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.6.2) 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. Example: The Navier Stokes is not solved between time t0 and t1.
 Navier_Sokes_Standard
 { equation_non_resolue (t>t0)*(t<t1) }

4.24 navier_stokes_turbulent

Description: NAVIER STOKES equations as well as the associated turbulence model equations.

Keyword Discretiser should have already be used to read the object.

See also: `navier_stokes_standard` (4.22.6) `navier_stokes_turbulent_qc` (4.25.11)

Usage:

```
navier_stokes_turbulent obj Lire obj {
    [ modele_turbulence modele_turbulence_hyd_deriv]
    [ 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]
    [ 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
```

- **modele_turbulence** *modele_turbulence_hyd_deriv* (4.24): Turbulence model for NAVIER STOKES equations.
- **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 first time step. Options are : *avec_les_cl* (default option $\text{lapP}=0$ is solved with Neuman boundary conditions on pressure if any), *avec_sources* ($\text{lapP}=f$ is solved with Neuman boundaries conditions and f integrating the source terms of the Navier Stokes equation) and *avec_sources_et_operateurs* ($\text{lapP}=f$ is solved as with the previous option *avec_sources* but f integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks $\text{DivU}=0$. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (8.11) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (8.11) 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* (4.19) 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* (4.20) 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 t_n , the linear system $Ax=B$ is considered as solved if the residual $\|Ax-B\| < \text{seuil}(t_n)$. For t_{n+1} , the threshold value $\text{seuil}(t_{n+1})$ will be evaluated as:
If ($\text{lmax}(\text{DivU}) * \text{dt} < \text{value}$)

```

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* (4.21) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limite** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.6.2) 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 is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.25 modele_turbulence_hyd_deriv

Description: Basic class for turbulence model for NAVIER STOKES equations.

See also: objet_lecture (30) NUL (4.25.1) mod_turb_hyd_ss_maille (4.25.2) k_epsilon (4.25.9)

Usage:

```

modele_turbulence_hyd_deriv {
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
    [ turbulence_paroit turbulence_paroit_base]
    [ dt_impr_ustar float]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
    [ nut_max float]
    [ eps_min float]
}

```

```

[ k_min float]
[ prandtl_k float]
[ prandtl_eps 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_paro** *turbulence_paro_base* (27): Keyword to set the wall law.
- **dt_impr_ustar** *float*: This keyword is used to print the values (U^+ , d^+ , u^*) 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* (4.25): 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).
- **eps_min** *float*: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float*: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float*: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float*: Keyword to change the Pre value (default 1.3).

4.25.1 dt_impr_ustar_mean_only

Description: `not_set`

See also: `objet_lecture` (30)

Usage:

```

{
    dt_impr float
    [ boundaries n word1 word2 ... wordn]
}
where

```

- **dt_impr** *float*
- **boundaries** *n word1 word2 ... wordn*

4.25.2 NUL

Description: not_set

See also: modele_turbulence_hyd_deriv (4.24)

Usage:

```
NUL [ correction_visco_turb_pour_controle_pas_de_temps ] [ correction_visco_turb_pour_controle-  
_pas_de_temps_parametre ] [ turbulence_paroil ] [ dt_impr_ustar ] [ dt_impr_ustar_mean_only ] [  
nut_max ] [ eps_min ] [ k_min ] [ prandtl_k ] [ prandtl_eps ]
```

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_paroil** *turbulence_paroil_base* (27): Keyword to set the wall law.
- **dt_impr_ustar** *float*: This keyword is used to print the values (U +, d+, u*) 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* (4.25): 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).
- **eps_min** *float*: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float*: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float*: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float*: Keyword to change the Pre value (default 1.3).

4.25.3 mod_turb_hyd_ss_maille

Description: Class for sub-grid turbulence model for NAVIER STOKES equations.

See also: modele_turbulence_hyd_deriv (4.24) sous_maille_wale (4.25.4) sous_maille_smago (4.25.5) combinaison (4.25.6) longueur_melange (4.25.7) sous_maille (4.25.8)

Usage:

```
mod_turb_hyd_ss_maille {  
    [ formulation_a_nb_points form_a_nb_points]  
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]  
    [ correction_visco_turb_pour_controle_pas_de_temps ]  
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]  
    [ turbulence_paroil turbulence_paroil_base]  
    [ dt_impr_ustar float]
```

```

[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
[ eps_min float]
[ k_min float]
[ prandtl_k float]
[ prandtl_eps float]
}
where

```

- **formulation_a_nb_points** *form_a_nb_points* (4.25.3): The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str* into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']: different ways to calculate the characteristic length may be specified :
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: 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* for inheritance: 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_paro** *turbulence_paro_base* (27) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) 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* (4.25) for inheritance: 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* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.25.4 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 homogeneity planes. Example for channel flows, planes parallel to the walls.

See also: `objet_lecture` ([30](#))

Usage:

nb dir1 dir2

where

- **nb** *int into [4]*: Number of points.
- **dir1** *int*: First direction.
- **dir2** *int*: Second direction.

4.25.5 sous_maille_wale

Description: This is the WALE-model. It is a new sub-grid scale model for eddy-viscosity in LES that has the following properties :

- it goes naturally to 0 at the wall (it doesn't need any information on the wall position or geometry)
- it has the proper wall scaling in $o(y^3)$ in the vicinity of the wall
- it reproduces correctly the laminar to turbulent transition.

See also: `mod_turb_hyd_ss_maille` ([4.25.2](#))

Usage:

sous_maille_wale {

```
[ cw float]
[ formulation_a_nb_points form_a_nb_points]
[ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]
[ correction_visco_turb_pour_controle_pas_de_temps ]
[ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
[ turbulence_paroit turbulence_paroit_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
[ eps_min float]
[ k_min float]
[ prandtl_k float]
[ prandtl_eps float]
```

}

where

- **cw** *float*: The unique parameter (constant) of the WALE-model (by default value 0.5).
- **formulation_a_nb_points** *form_a_nb_points* ([4.25.3](#)) for inheritance: The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
 - volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 - volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 - scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.

arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.

- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: 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* for inheritance: 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_paro** *turbulence_paro_base* (27) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) 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* (4.25) for inheritance: 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* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.25.6 sous_maille_smago

Description: Smagorinsky sub-grid turbulence model.

$$\text{Nut} = C_s1 * C_s1 * l * \sqrt{2 * S * S}$$

$$K = C_s2 * C_s2 * l * 2 * S$$

See also: `mod_turb_hyd_ss_maille` (4.25.2)

Usage:

```
sous_maille_smago {
    [ cs float]
    [ formulation_a_nb_points form_a_nb_points]
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
    [ turbulence_paro turbulence_paro_base]
    [ dt_impr_ustar float]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
    [ nut_max float]
    [ eps_min float]
    [ k_min float]
    [ prandtl_k float]
    [ prandtl_eps float]
}
```

where

- **cs** *float*: This is an optional keyword and the value is used to set the constant used in the Smagorinsky model (This is currently only valid for Smagorinsky models and it is set to 0.18 by default) .
- **formulation_a_nb_points** *form_a_nb_points* (4.25.3) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: 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* for inheritance: 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_paro** *turbulence_paro_base* (27) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) 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* (4.25) for inheritance: 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* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.25.7 combinaison

Description: This keyword specify a turbulent viscosity model where the turbulent viscosity is user-defined.

See also: mod_turb_hyd_ss_maille (4.25.2)

Usage:

```
combinaison {  
    [ nb_var  n word1 word2 ... wordn ]
```

```

[ fonction str]
[ formulation_a_nb_points form_a_nb_points]
[ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]
[ correction_visco_turb_pour_controle_pas_de_temps ]
[ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
[ turbulence_paroit turbulence_paroit_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
[ eps_min float]
[ k_min float]
[ prandtl_k float]
[ prandtl_eps float]
}

```

where

- **nb_var** *n word1 word2 ... wordn*: Number and names of variables which will be used in the turbulent viscosity definition (by default 0)
- **fonction** *str*: Fonction for turbulent viscosity. X,Y,Z and variables defined previously can be used.
- **formulation_a_nb_points** *form_a_nb_points* (4.25.3) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: 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* for inheritance: 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_paroit** *turbulence_paroit_base* (27) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) 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* (4.25) for inheritance: 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* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.25.8 longueur_melange

Description: This model is based on mixing length modelling. For a non academic configuration, formulation used in the code can be expressed basically as :

$$\nu_t = (Kappa.y)^2.dU/dy$$

Till a maximum distance (dmax) set by the user in the data file, y is set equal to the distance from the wall (dist_w) calculated previously and saved in file Wall_length.xyz. [see Distance_paro keyword]

Then (from y=dmax), y decreases as an exponential function : $y = dmax * \exp[-2. * (dist_w - dmax) / dmax]$

See also: mod_turb_hyd_ss_maille (4.25.2)

Usage:

```
longueur_melange {
    [ canalx float]
    [ tuyauz float]
    [ verif_dparoi str]
    [ dmax float]
    [ fichier str]
    [ fichier_ecriture_K_Eps str]
    [ formulation_a_nb_points form_a_nb_points]
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
    [ turbulence_paro turbulence_paro_base]
    [ dt_impr_ustar float]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
    [ nut_max float]
    [ eps_min float]
    [ k_min float]
    [ prandtl_k float]
    [ prandtl_eps float]
}
```

where

- **canalx** *float*: [height] : plane channel according to Ox direction (for the moment, formulation in the code relies on fixed height : H=2).
- **tuyauz** *float*: [diameter] : pipe according to Oz direction (for the moment, formulation in the code relies on fixed diameter : D=2).
- **verif_dparoi** *str*
- **dmax** *float*: Maximum distance.
- **fichier** *str*
- **fichier_ecriture_K_Eps** *str*: When a restart with k-epsilon model is envisaged, this keyword allows to generate external MED-format file with evaluation of k and epsilon quantities (based on eddy turbulent viscosity and turbulent characteristic length returned by mixing length model). The frequency of the MED file print is set equal to dt_impr_ustar. Moreover, k-eps MED field is automatically saved at the last time step. MED file is then used for the restarting K-Epsilon calculation with the Champ_Fonc_Med keyword.

- **formulation_a_nb_points** *form_a_nb_points* (4.25.3) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: 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* for inheritance: 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_paro** *turbulence_paro_base* (27) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) 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* (4.25) for inheritance: 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* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.25.9 sous_maille

Description: Structure sub-grid function model.

See also: mod_turb_hyd_ss_maille (4.25.2)

Usage:

```
sous_maille {
    [ formulation_a_nb_points form_a_nb_points ]
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete'] ]
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float ]
    [ turbulence_paro turbulence_paro_base ]
}
```

```

[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
[ eps_min float]
[ k_min float]
[ prandtl_k float]
[ prandtl_eps float]
}

```

where

- **formulation_a_nb_points** *form_a_nb_points* (4.25.3) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: 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* for inheritance: 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_paro** *turbulence_paro_base* (27) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) 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* (4.25) for inheritance: 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* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.25.10 k_epsilon

Description: Turbulence model (k-eps).

See also: `modele_turbulence_hyd_deriv` (4.24)

Usage:

```
k_epsilon {  
    [ cmu float]  
    transport_k_epsilon transport_k_epsilon  
    [ modele_fonc_bas_reynolds modele_fonction_bas_reynolds_base]  
    [ correction_visco_turb_pour_controle_pas_de_temps ]  
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]  
    [ turbulence_paro turbulence_paro_base]  
    [ dt_impr_ustar float]  
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]  
    [ nut_max float]  
    [ eps_min float]  
    [ k_min float]  
    [ prandtl_k float]  
    [ prandtl_eps float]  
}
```

where

- **cmu** *float*: Keyword to modify the Cmu constant of k-eps model : $Nut = Cmu * k / \epsilon$ Default value is 0.09
- **transport_k_epsilon** *transport_k_epsilon* (4.26): Keyword to define the (k-eps) transportation equation.
- **modele_fonc_bas_reynolds** *modele_fonction_bas_reynolds_base* (4.25.10): This keyword is used to set the bas Reynolds model used.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: 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* for inheritance: 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_paro** *turbulence_paro_base* (27) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) 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* (4.25) for inheritance: 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* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.25.11 modele_fonction_bas_reynolds_base

Description: not_set

See also: objet_lecture (30)

Usage:

4.26 navier_stokes_turbulent_qc

Description: NAVIER STOKES equations under small Mach number as well as the associated turbulence model equations.

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

See also: navier_stokes_turbulent (4.23)

Usage:

```
navier_stokes_turbulent_qc obj Lire obj {  
    [ modele_turbulence modele_turbulence_hyd_deriv]  
    [ 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]  
    [ 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
```

- **modele_turbulence** *modele_turbulence_hyd_deriv* (4.24) for inheritance: Turbulence model for NAVIER STOKES equations.
- **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 first time step. Options are : *avec_les_cl* (default option $\text{lapP}=0$ is solved with Neuman boundary conditions on pressure if any), *avec_sources* ($\text{lapP}=f$ is solved with Neuman boundaries conditions and f integrating the source terms of the Navier Stokes equation) and *avec_sources_et_operateurs* ($\text{lapP}=f$ is solved as with the previous option *avec_sources* but f integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicit when using an implicit time scheme to solve the Navier Stokes equation.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks $\text{DivU}=0$. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (8.11) for inheritance: Linear pressure system resolution method.

- **solveur_bar** *solveur_sys_base* (8.11) 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* (4.19) 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* (4.20) 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 $\|Ax-B\| < \text{seuil}(tn)$. For tn+1, the threshold value $\text{seuil}(tn+1)$ will be evaluated as:
 If ($\text{lmax}(\text{DivU}) * dt < \text{value}$)
 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* (4.21) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.3.1) for inheritance: Boundary conditions.
- **sources** *sources* (4.4.1) 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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* (4.6.2) 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. Example: The Navier Stokes is not solved between time t0 and t1.
 Navier_Sokes_Standard
 { equation_non_resolue (t>t0)*(t<t1) }

4.27 transport_k_epsilon

Description: The (k-eps) transportation equation. To restart from a previous mixing length calculation, an external MED-format file containing reconstructed K and Epsilon quantities can be read (see fichier_écriture_k_eps) thanks to the Champ_fonc_MED keyword.

Warning, When used with the Quasi-compressible model, k and eps should be viewed as rho k and rho epsilon when defining initial and boundary conditions or when visualizing values for k and eps. This bug will be fixed in a future version.

Keyword Discretiser should have already be used to read the object.

See also: eqn_base (4.17)

Usage:

```
transport_k_epsilon obj Lire obj {  
    [ with_nu str into ['yes', 'no']]  
    [ 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

- **with_nu** str into ['yes', 'no']: yes/no
- **convection** bloc_convection (4.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** bloc_diffusion (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** condinits (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** condlims (4.3.1) for inheritance: Boundary conditions.
- **sources** sources (4.4.1) 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 (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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 (4.5) for inheritance: This keyword is used to write the values of a field for the whole domain or 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 (4.6.2) 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 is not solved between

```

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

```

5 /*

5.1 /*

Description: bloc of Comment in a data file.

See also: [objet_u \(31\)](#)

Usage:

```
/* comm
```

where

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

6 champ_generique_base

Description: not_set

See also: [objet_u \(31\)](#) [champ_post_de_champs_post \(6\)](#) [predefini \(6.14\)](#) [champ_post_refchamp \(6.16\)](#)

Usage:

6.1 champ_post_de_champs_post

Description: not_set

See also: [champ_generique_base \(6\)](#) [champ_post_operateur_eqn \(6.4\)](#) [champ_post_transformation \(6.18\)](#) [champ_post_reduction_0d \(6.15\)](#) [champ_post_operateur_base \(6.3\)](#) [champ_post_statistiques_base \(6.5\)](#) [champ_post_extraction \(6.9\)](#) [champ_post_morceau_equation \(6.12\)](#) [champ_post_tparoi_vef \(6.17\)](#) [champ_post_interpolation \(6.11\)](#)

Usage:

```
champ_post_de_champs_post obj Lire obj {
```

```

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

```

```
}
```

where

- **source** *champ_generique_base (6)*: the source field.
- **nom_source** *str*: To name a source field with the nom_source keyword
- **source_reference** *str*
- **sources_reference** *list_nom_virgule (6.1)*
- **sources** *listchamp_generique (6.2)*: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

6.2 list_nom_virgule

Description: List of name.

See also: listobj (29.2)

Usage:

{ object1 , object2 }

list of *nom_anonyme* (20) separated with ,

6.3 listchamp_generique

Description: XXX

See also: listobj (29.2)

Usage:

{ object1 , object2 }

list of *champ_generique_base* (6) separated with ,

6.4 champ_post_operateur_base

Description: not_set

See also: champ_post_de_champs_post (6) champ_post_operateur_gradient (6.10) champ_post_operateur_divergence (6.7)

Usage:

champ_post_operateur_base obj Lire obj {

 [**source** *champ_generique_base*]

 [**nom_source** *str*]

 [**source_reference** *str*]

 [**sources_reference** *list_nom_virgule*]

 [**sources** *listchamp_generique*]

}

where

- **source** *champ_generique_base* (6) 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* (6.1) for inheritance
- **sources** *listchamp_generique* (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

6.5 champ_post_operateur_eqn

Description: not_set

See also: champ_post_de_champs_post (6)

Usage:

champ_post_operateur_eqn obj Lire obj {

```

[ numero_op int]
[ numero_source int]
[ sans_solveur_masse ]
[ 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**
- **source** *champ_generique_base* (6) 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* (6.1) for inheritance
- **sources** *listchamp_generique* (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

6.6 champ_post_statistiques_base

Description: not_set

See also: champ_post_de_champs_post (6) correlation (6.6) moyenne (6.13) ecart_type (6.8)

Usage:

champ_post_statistiques_base obj Lire obj {

```

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* (6) 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* (6.1) for inheritance
- **sources** *listchamp_generique* (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

6.7 correlation

Description: to calculate the correlation between the two fields.

See also: `champ_post_statistiques_base` (6.5)

Usage:

```
correlation obj Lire obj {  
    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 (6) 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 (6.1) for inheritance
- **sources** listchamp_generique (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

6.8 champ_post_operateur_divergence

Description: To calculate divergency of a given field.

See also: `champ_post_operateur_base` (6.3)

Usage:

```
champ_post_operateur_divergence obj Lire obj {  
    [ source champ_generique_base]  
    [ nom_source str]  
    [ source_reference str]  
    [ sources_reference list_nom_virgule]  
    [ sources listchamp_generique]  
}
```

where

- **source** champ_generique_base (6) 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 (6.1) for inheritance
- **sources** listchamp_generique (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

6.9 ecart_type

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

See also: `champ_post_statistiques_base` (6.5)

Usage:

```
ecart_type obj Lire obj {  
    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 (6) 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 (6.1) for inheritance
- **sources** listchamp_generique (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post..
{ ... }}

6.10 champ_post_extraction

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

See also: `champ_post_de_champs_post` (6)

Usage:

```
champ_post_extraction obj Lire obj {  
    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 (6) 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* (6.1) for inheritance
- **sources** *listchamp_generique* (6.2) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

6.11 champ_post_operateur_gradient

Description: To calculate gradient of a given field.

See also: `champ_post_operateur_base` (6.3)

Usage:

champ_post_operateur_gradient obj Lire obj {

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

}

where

- **source** *champ_generique_base* (6) 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* (6.1) for inheritance
- **sources** *listchamp_generique* (6.2) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

6.12 champ_post_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` (6)

Usage:

champ_post_interpolation obj Lire obj {

```
localisation str
[ optimisation_sous_maillage str into ['default', 'yes', 'no']]
[ methode str]
[ domaine str]
[ 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).

- **optimisation_sous_maillage** *str* into ['default', 'yes', 'no']
- **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)
- **source** *champ_generique_base* (6) 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* (6.1) for inheritance
- **sources** *listchamp_generique* (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

6.13 champ_post_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 (6)

Usage:

champ_post_morceau_equation obj Lire obj {

```

    type str
    numero int
    option str into ['stabilite', 'flux_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']: option is stability for time steps or flux_bords for boundary 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* (6) 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* (6.1) for inheritance
- **sources** *listchamp_generique* (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

6.14 moyenne

Description: to calculate the average of the field over time

See also: champ_post_statistiques_base (6.5)

Usage:

```

moyenne obj Lire obj {
    [ 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): This option allows to read a converged time averaged field in a .xyz file in order to calculate, when restarting the calculation, the statistics fields (rms, correlation) which depend on this average. In that case, the time averaged field is not updated during the restarting 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* (6) 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* (6.1) for inheritance
- **sources** *listchamp_generique* (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

6.15 predefini

Description: These keyword is used to post process predefined postprocessing fields. For the moment, only kinetic energy (**energie_cinetique** keyword to use for **field_name**) is available.

See also: **champ_generique_base** (6)

Usage:

```

predefini obj Lire obj {

```

```

    pb_champ deuxmots

```

```

}

```

where

- **pb_champ** *deuxmots* (4.19): { **Pb_champ** **nom_pb** **nom_champ** } : **nom_pb** is the problem name and **nom_champ** is the selected field name.

6.16 champ_post_reduction_0d

Description: To calculate the min, max, or mean value of a field.

See also: **champ_post_de_champs_post** (6)

Usage:

```

champ_post_reduction_0d obj Lire obj {

```

```

methode str into ['min', 'max', 'moyenne', 'somme', 'moyenne_ponderee', 'somme_ponderee',
'norme_l2']
[ 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', 'somme', 'moyenne_ponderee', 'somme_ponderee', 'norme_l2']: name of the reduction method (min, max, somme for the sum, somme_ponderee for a weighted sum (integral), norme_L2 for the L2 norm, moyenne for a mean and moyenne_ponderee for a mean ponderated by integration volumes, e.g: cell volumes for temperature or pressure in VDF, volumes around faces for velocity and temperature in VEF)
- **source** *champ_generique_base* (6) 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* (6.1) for inheritance
- **sources** *listchamp_generique* (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

6.17 champ_post_refchamp

Description: Field of prolem

See also: champ_generique_base (6)

Usage:

```

champ_post_refchamp obj Lire obj {
    pb_champ deuxmots
    [ nom_source str]
}
where

```

- **pb_champ** *deuxmots* (4.19): { 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

6.18 champ_post_tparoi_vef

Description: These 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 (6)

Usage:

```

champ_post_tparoi_vef obj Lire obj {

```

```

[ source champ_generique_base]
[ nom_source str]
[ source_reference str]
[ sources_reference list_nom_virgule]
[ sources listchamp_generique]
}
where

```

- **source** *champ_generique_base* (6) 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* (6.1) for inheritance
- **sources** *listchamp_generique* (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

6.19 champ_post_transformation

Description: To create a field with a transformation.

See also: `champ_post_de_champs_post` (6)

Usage:

champ_post_transformation obj Lire obj {

```

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* (6) 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* (6.1) for inheritance
- **sources** *listchamp_generique* (6.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

7 chimie

Description: Keyword to describe the chemical reactions

See also: `objet_u` (31)

Usage:

```
chimie obj Lire obj {
    reactions reactions
    [ modele_micro_melange int ]
    [ constante_modele_micro_melange float ]
    [ espece_en_competition_micro_melange str ]
}
```

where

- **reactions** *reactions* (7): 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

7.1 reactions

Description: list of reactions

See also: `listobj` (29.2)

Usage:

```
{ object1 , object2 .... }
```

list of *reaction* (7.1) separated with ,

7.1.1 reaction

Description: Keyword to describe reaction:

$w = K \text{ pow}(T, \beta) \exp(-E_a / (R T)) \prod \text{pow}(\text{Reactif}_i, \text{activity}_i)$.

If $K_{\text{inv}} > 0$,

$w = K \text{ pow}(T, \beta) \exp(-E_a / (R T)) (\prod \text{pow}(\text{Reactif}_i, \text{activity}_i) - K_{\text{inv}} / \exp(-c_r E_a / (R T)) \prod \text{pow}(\text{Produit}_i, \text{activity}_i))$

See also: `objet_lecture` (30)

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*H2O)
- **constante_taux_reaction** *float*: constante of cinetic K
- **coefficients_activites** *bloc_lecture* (2.37): coefficients of activity (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

8 class_generic

Description: not_set

See also: objet_u (31) dt_start (8.4) solveur_sys_base (8.11)

Usage:

8.1 cholesky

Description: Cholesky direct method.

See also: solveur_sys_base (8.11)

Usage:

cholesky obj Lire obj {

```

    [ impr ]
    [ quiet ]

```

}

where

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

8.2 dt_calc

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

See also: dt_start (8.4)

Usage:

dt_calc

8.3 dt_fixe

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

See also: [dt_start \(8.4\)](#)

Usage:

dt_fixe **value**

where

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

8.4 dt_min

Description: The first iteration is based on dt_min.

See also: [dt_start \(8.4\)](#)

Usage:

dt_min

8.5 dt_start

Description: not_set

See also: [class_generic \(8\)](#) [dt_calc \(8.1\)](#) [dt_min \(8.3\)](#) [dt_fixe \(8.2\)](#)

Usage:

dt_start

8.6 gcp_ns

Description: not_set

See also: [gcp \(8.10\)](#)

Usage:

gcp_ns obj Lire obj {

```
    solveur0 solveur_sys_base
    solveur1 solveur_sys_base
    [ precond precond_base ]
    [ precond_nul ]
    seuil float
    [ impr ]
    [ quiet ]
    [ save_matrix|save_matrice ]
    [ optimized ]
```

}

where

- **solveur0** *solveur_sys_base* [\(8.11\)](#): Solver type.
- **solveur1** *solveur_sys_base* [\(8.11\)](#): Solver type.

- **precond** *precond_base* (22) 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.

8.7 gen

Description: not_set

See also: solveur_sys_base (8.11)

Usage:

gen data

where

- **data** *bloc_lecture* (2.37)

8.8 gmres

Description: Gmres method (for non symmetric matrix).

See also: solveur_sys_base (8.11)

Usage:

gmres obj Lire obj {

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



```
}
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.

8.9 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 ([8.11](#))

Usage:

```
optimal obj Lire obj {
```

```
    seuil float
    [ impr ]
    [ quiet ]
    [ save_matrix|save_matrice ]
    [ frequence_recalc int]
    [ nom_fichier_solveur str]
    [ fichier_solveur_non_recre ]
```

```
}
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 fastest solver
- **nom_fichier_solveur** *str*: To specify the file containing the list of the tested solvers
- **fichier_solveur_non_recre** : To avoid the creation of the file containing the list

8.10 petsc

Description: Solveur 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

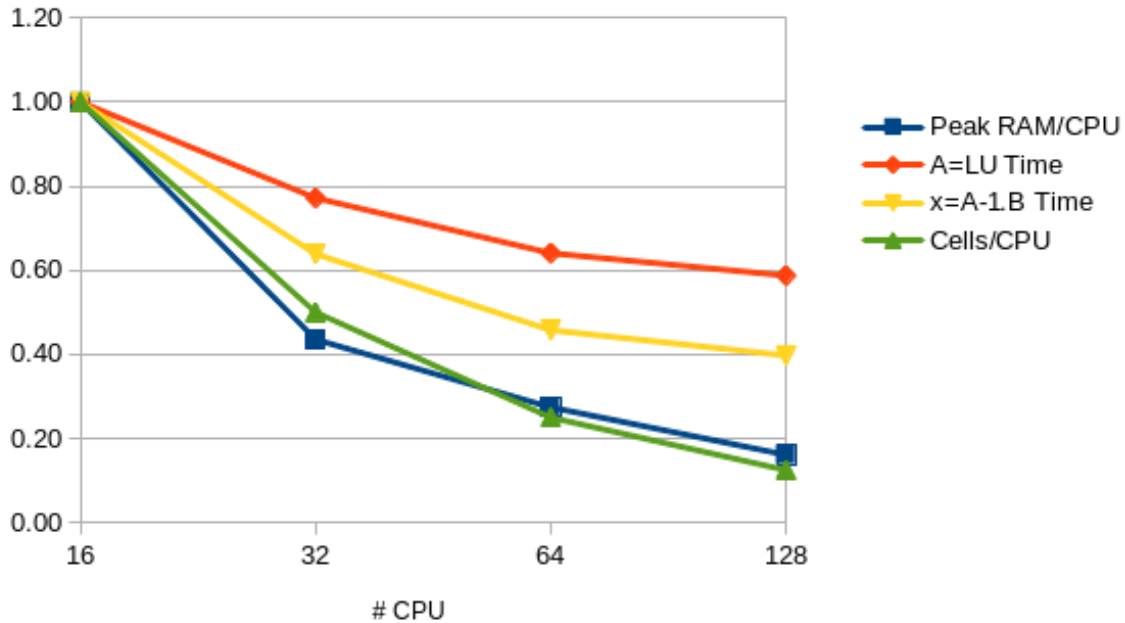
IBICGSTAB : Improved version of previous one for massive parallel computations (only a single global reduction operation instead of the usual 3 or 4).

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't 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 than the default one. Notice that this solver requires a huge amount of memory compared to iterative methods. To know how many RAM you will need by core, then use the **impr** option to have detailed 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) :

Relative evolution compare to a 16 CPUs parallel calculation
on a 2.6e6 cells mesh (163000 cells/CPU) where:
Peak RAM/CPU is 6.2GB
A=LU in factorization in 206 s
 $x=A^{-1}B$ solve in 0.83 s



CHOLESKY_OUT_OF_CORE : Same as the previous one but with a written LU decomposition of disc (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 bandwidth of the local matrix, may interestingly improve the quality of the decomposition and reduce 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 Threshold 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 (generally 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 disc (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 completely 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 equation):

-The **CHOLSKY** 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 scalability 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 there: `$TRUST_ROOT/lib/src/LIBPETSC/petsc/*/do`

See also: `solveur_sys_base` (8.11)

Usage:

petsc solveur option_solveur

where

- **solveur** *str*
- **option_solveur** *bloc_lecture* (2.37)

8.11 gcp

Description: Preconditioned conjugated gradient.

See also: `solveur_sys_base` (8.11) `gcp_ns` (8.5)

Usage:

gcp obj Lire obj {

```

[ precond precond_base]
[ precond_nul ]
seuil float
[ impr ]
[ quiet ]
[ save_matrix|save_matrice ]
[ optimized ]
}
where

```

- **precond** *precond_base* (22): 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.

8.12 solveur_sys_base

Description: Basic class to solve the linear system.

See also: [class_generic \(8\)](#) [optimal \(8.8\)](#) [gen \(8.6\)](#) [petsc \(8.9\)](#) [gcp \(8.10\)](#) [cholesky \(8\)](#) [gmres \(8.7\)](#)

Usage:

9

9.1

Description: Comments in a data file.

See also: [objet_u \(31\)](#)

Usage:

comm

where

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

10 condlim_base

Description: Basic class of boundary conditions.

See also: objet_u (31) paroi_fixe (10.32) symetrie (10.40) periodique (10.38) paroi_decalee_robin (10.24) paroi_adiabatique (10.20) dirichlet (10.1) neumann (10.19) paroi_couple (10.23) paroi_contact (10.21) paroi_contact_fictif (10.22) paroi_echange_contact_vdf (10.28) paroi_echange_externe_impose (10.29) paroi_echange_global_impose (10.31) Paroi (10) frontiere_ouverte_k_eps_impose (10.11) paroi_flux_impose (10.34) frontiere_ouverte_fraction_massique_imposee (10.5) paroi_echange_contact_correlation_vdf (10.26) paroi_echange_contact_correlation_vef (10.27)

Usage:

condlim_base

10.1 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 (10)

Usage:

Paroi

10.2 dirichlet

Description: Dirichlet condition at the boundary called bord (edge) : 1). For NAVIER STOKES equations, speed imposed at the boundary; 2). For scalar transport equation, scalar imposed at the boundary.

See also: condlim_base (10) paroi_defilante (10.25) paroi_knudsen_non_negligeable (10.35) paroi_rugueuse (10.36) frontiere_ouverte_vitesse_imposee (10.17) frontiere_ouverte_temperature_imposee (10.16) frontiere_ouverte_concentration_imposee (10.4) paroi_temperature_imposee (10.37)

Usage:

dirichlet

10.3 entree_temperature_imposee_h

Description: Particular case of class frontiere_ouverte_temperature_imposee for enthalpy equation.

See also: frontiere_ouverte_temperature_imposee (10.16)

Usage:

entree_temperature_imposee_h ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.4 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 \(10.19\)](#)

Usage:

frontiere_ouverte var_name ch

where

- **var_name** *str* into ['T_ext', 'C_ext', 'K_Eps_ext', 'Fluctu_Temperature_ext', 'Flux_Chaleur_Turb-_ext', 'V2_ext']: Field name.
- **ch** *champ_front_base* (15): Boundary field type.

10.5 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 speed condition.

See also: [dirichlet \(10.1\)](#)

Usage:

frontiere_ouverte_concentration_imposee ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.6 frontiere_ouverte_fraction_massique_imposee

Description: not_set

See also: [condlim_base \(10\)](#)

Usage:

frontiere_ouverte_fraction_massique_imposee ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.7 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 \(10.19\)](#)

Usage:

frontiere_ouverte_gradient_pression_impose ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.8 **frontiere_ouverte_gradient_pression_impose_vef**

Description: Keyword for an outlet boundary condition on the gradient of the pressure. This boundary condition may only be applied in the VEF module.

See also: `frontiere_ouverte_pression_imposee` ([10.12](#)) `frontiere_ouverte_gradient_pression_impose_vefprep1b` ([10.8](#))

Usage:

frontiere_ouverte_gradient_pression_impose_vef ch

where

- **ch** *champ_front_base* ([15](#)): Boundary field type.

10.9 **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_vef` ([10.7](#))

Usage:

frontiere_ouverte_gradient_pression_impose_vefprep1b ch

where

- **ch** *champ_front_base* ([15](#)): Boundary field type.

10.10 **frontiere_ouverte_gradient_pression_libre_vef**

Description: Class for outlet boundary condition in VEF like Orlansky. There is no reference for pressure for these 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` ([10.19](#))

Usage:

frontiere_ouverte_gradient_pression_libre_vef

10.11 **frontiere_ouverte_gradient_pression_libre_vefprep1b**

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

See also: `neumann` ([10.19](#))

Usage:

frontiere_ouverte_gradient_pression_libre_vefprep1b

10.12 **frontiere_ouverte_k_eps_impose**

Description: Turbulence condition imposed on an open boundary called bord (edge) (this situation corresponds to a fluid inlet). This condition must be associated with an imposed inlet speed condition.

See also: `condlim_base` ([10](#))

Usage:

frontiere_ouverte_k_eps_impose ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.13 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* (10.19) *frontiere_ouverte_gradient_pression_impose_vef* (10.7)

Usage:

frontiere_ouverte_pression_imposee ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.14 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 symmetry in a channel) of the boundary where Orlansky conditions are imposed.

See also: *neumann* (10.19)

Usage:

frontiere_ouverte_pression_imposee_orlansky

10.15 frontiere_ouverte_pression_moyenne_imposee

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

See also: *neumann* (10.19)

Usage:

frontiere_ouverte_pression_moyenne_imposee pext

where

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

10.16 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 speed 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* (10.18)

Usage:

frontiere_ouverte_rho_u_impose ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.17 **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 speed condition. The imposed temperature value is expressed in oC or K.

See also: *dirichlet* (10.1) *entree_temperature_imposee_h* (10.2)

Usage:

frontiere_ouverte_temperature_imposee ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.18 **frontiere_ouverte_vitesse_imposee**

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

See also: *dirichlet* (10.1) *frontiere_ouverte_vitesse_imposee_sortie* (10.18)

Usage:

frontiere_ouverte_vitesse_imposee ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.19 **frontiere_ouverte_vitesse_imposee_sortie**

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

See also: *frontiere_ouverte_vitesse_imposee* (10.17) *frontiere_ouverte_rho_u_impose* (10.15)

Usage:

frontiere_ouverte_vitesse_imposee_sortie ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.20 **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` (10) `frontiere_ouverte_gradient_pression_libre_vef` (10.9) `frontiere_ouverte_gradient_pression_libre_vefprep1b` (10.10) `frontiere_ouverte_gradient_pression_impose` (10.6) `frontiere_ouverte_pression_imposee` (10.12) `frontiere_ouverte_pression_imposee_orlansky` (10.13) `frontiere_ouverte_pression_moyenne_imposee` (10.14) `frontiere_ouverte` (10.3) `sortie_libre_temperature_imposee_h` (10.39)

Usage:

neumann

10.21 `paroi_adiabatique`

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

See also: `condlim_base` (10)

Usage:

paroi_adiabatique

10.22 `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 2-4-4-4-4-2

2-4-4-4-4-2 2-2-2

2-2-2-2-2 2-4-2

2-2

OK

2-2 2-2-2

2-4-2 2-2

2-2 2-2

NOT OK

See also: `condlim_base` (10)

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

10.23 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 ([10](#))

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

10.24 paroi_couple

Description: not_set

See also: condlim_base ([10](#))

Usage:

paroi_couple **autrepb**
where

- **autrepb** *str*: Name of other problem.

10.25 paroi_decalee_robin

Description: This keyword is used to designate a Robin boundary condition ($a.u+b.du/dn=c$) associated with the Pironneau methodology for the wall laws. The value of given by the delta option is the distance between the mesh (where symmetry boundary condition is applied) and the fictious wall. This boundary condition needs the definition of the dedicated source terms (Source_Robin or Source_Robin_Scalaire) according the equations used.

See also: condlim_base ([10](#))

Usage:

paroi_decalee_robin obj Lire obj {

delta *float*

}

where

- **delta** *float*

10.26 paroi_defilante

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

See also: [dirichlet \(10.1\)](#)

Usage:

paroi_defilante **ch**

where

- **ch** *champ_front_base* (15): Boundary field type.

10.27 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 \(10\)](#)

Usage:

paroi_echange_contact_correlation_vdf obj Lire obj {

```
    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 the temperature 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 restarting calculation with this correlation.

10.28 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 ([10](#))

Usage:

```
paroi_echange_contact_correlation_vef obj Lire obj {
    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 the temperature 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 restarting calculation with this correlation.

10.29 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 \(10\)](#)

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 :
$$f_i = h (T_1 - T_2)$$
 where $1/h = d_1/\lambda_{a1} + 1/\text{val_h_contact} + d_2/\lambda_{a2}$
where d_i : distance between the node where T_i and the wall is found.

10.30 paroi_echange_externe_impose

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

See also: [condlim_base \(10\)](#) [paroi_echange_externe_impose_h \(10.30\)](#)

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)*: Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base (15)*: Boundary field type.

10.31 paroi_echange_externe_impose_h

Description: Particular case of class `paroi_echange_externe_impose` for enthalpy equation.

See also: [paroi_echange_externe_impose \(10.29\)](#)

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): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* (15): Boundary field type.

10.32 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 \(10\)](#)

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): Boundary field type.
- **text** *str*: External temperature value. The external temperature value is expressed in oC or K.
- **ch** *champ_front_base* (15): Boundary field type.

10.33 paroi_fixe

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

See also: [condlim_base \(10\)](#) [paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets \(10.33\)](#)

Usage:

paroi_fixe

10.34 paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets

Description: CL to obtain iso Genepi2...

See also: [paroi_fixe \(10.32\)](#)

Usage:

paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets

10.35 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 \(10\)](#)

Usage:

paroi_flux_impose ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.36 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.

$$U(y=0)-U_{wall}=k(dU/dY)$$

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 \(10.1\)](#)

Usage:

paroi_knudsen_non_negligeable **name_champ_1** **champ_1** **name_champ_2** **champ_2**

where

- **name_champ_1** *str into ['vitesse_pari', 'k']*: Field name.
- **champ_1** *champ_front_base (15)*: Boundary field type.
- **name_champ_2** *str into ['vitesse_pari', 'k']*: Field name.
- **champ_2** *champ_front_base (15)*: Boundary field type.

10.37 paroi_rugueuse

Description: Rough wall boundary

See also: [dirichlet \(10.1\)](#)

Usage:

paroi_rugueuse **obj** Lire obj {

erugu *float*

}

where

- **erugu** *float*: Constant value for roughness

10.38 paroi_temperature_imposee

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

See also: [dirichlet \(10.1\)](#) [temperature_imposee_pari \(10.41\)](#)

Usage:

paroi_temperature_imposee **ch**

where

- **ch** *champ_front_base (15)*: Boundary field type.

10.39 periodique

Description: 1). For NAVIER STOKES equations, this keyword is used to indicate the fact that the horizontal speed inlet values are the same as the outlet speed 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` (10)

Usage:

periodique

10.40 sortie_libre_temperature_imposee_h

Description: Open boundary for heat equation with enthalpy as unknown.

See also: `neumann` (10.19)

Usage:

sortie_libre_temperature_imposee_h ch

where

- **ch** *champ_front_base* (15): Boundary field type.

10.41 symetrie

Description: 1). For NAVIER STOKES equations, this keyword is used to designate a symmetry condition concerning the speed at the boundary called bord (edge) (normal speed at the edge equal to zero and tangential speed 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` (10)

Usage:

symetrie

10.42 temperature_imposee_pari

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

See also: `paroi_temperature_imposee` (10.37)

Usage:

temperature_imposee_pari ch

where

- **ch** *champ_front_base* (15): Boundary field type.

11 discretisation_base

Description: Basic class for space discretization of thermohydraulic turbulent problems.

See also: `objet_u` (31) `vdf` (11.1) `vef` (11.2) `ef` (11)

Usage:

11.1 ef

Description: Element Finite discretization.

See also: discretisation_base (11)

Usage:

11.2 vdf

Description: Finite difference volume discretization.

See also: discretisation_base (11)

Usage:

11.3 vef

Description: Finite element volume discretization (P1NC/P0 element)

Warning: it becomes an obsolete discretization.

See also: discretisation_base (11) vefprep1b (11.3)

Usage:

11.4 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 Lire. 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 Lire dis { P0 P1 Changement_de_base_P1Bulle 1 Cl_pression_sommet_faible 0 }

See also: vef (11.2)

Usage:

vefprep1b obj Lire obj {

```
[ p0 ]
[ p1 ]
[ pa ]
[ changement_de_base_p1bulle int into [0, 1]]
[ cl_pression_sommet_faible int into [0, 1]]
[ modif_div_face_dirichlet int into [0, 1]]
```

}

where

- **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

- **changement_de_base_p1bulle** *int into [0, 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).
- **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).
- **modif_div_face_dirichlet** *int into [0, 1]*: This option (by default 0) is used to extend control volumes for the momentum equation.

12 domaine

Description: Keyword to create a domain.

See also: `objet_u` (31)

Usage:

13 espece

Description: `not_set`

See also: `objet_u` (31)

Usage:

```
espece obj Lire obj {
    cp champ_base
    lambda champ_base
    mu champ_base
    masse_molaire float
}
```

where

- **cp** *champ_base* (14): Specific heat value (J.kg-1.K-1).
- **lambda** *champ_base* (14): Conductivity value (W.m-1.K-1).
- **mu** *champ_base* (14): Dynamic viscosity value (kg.m-1.s-1).
- **masse_molaire** *float*: Gas molar mass.

14 champ_base

14.1 champ_base

Description: Basic class of fields.

See also: `objet_u` (31) `champ_don_base` (14.1) `champ_ostwald` (14.14) `champ_input_base` (14.12) `champ_fonc_med` (14.5) `field_uniform_keps_from_ud` (14.22)

Usage:

14.2 champ_don_base

Description: Basic class for data fields (not calculated), p.e. physics properties.

See also: [champ_base \(14\)](#) [uniform_field \(14.25\)](#) [champ_uniforme_morceaux \(14.18\)](#) [champ_fonc_xyz \(14.21\)](#) [champ_fonc_txyz \(14.20\)](#) [champ_don_lu \(14.2\)](#) [init_par_partie \(14.23\)](#) [champ_tabule_temps \(14.17\)](#) [champ_fonc_t \(14.8\)](#) [champ_fonc_tabule \(14.9\)](#) [champ_init_canal_sinal \(14.10\)](#) [champ_som_lu_vdf \(14.15\)](#) [champ_som_lu_vef \(14.16\)](#) [tayl_green \(14.24\)](#) [champ_fonc_reprise \(14.6\)](#)

Usage:

14.3 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 \(14.1\)](#)

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

14.4 champ_fonc_fonction

Description: Field that is a function of another field.

See also: [champ_fonc_tabule \(14.9\)](#) [champ_fonc_fonction_txyz \(14.4\)](#)

Usage:

champ_fonc_fonction dim inco bloc

where

- **dim** *int*: Number of field components.
- **inco** *str*: Name of the field (for example: temperature).
- **bloc** *bloc_lecture (2.37)*: 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)).

14.5 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 \(14.3\)](#)

Usage:

champ_fonc_fonction_txyz dim inco bloc

where

- **dim** *int*: Number of field components.
- **inco** *str*: Name of the field (for example: temperature).
- **bloc** *bloc_lecture* (2.37): 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)).

14.6 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 restart 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 restarting.

See also: champ_base (14)

Usage:

champ_fonc_med [**use_existing_domain**] [**last_time**] **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.
- **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.

14.7 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 (14.1)

Usage:

champ_fonc_reprise [**format**] **filename** **pb_name** **champ** [**fonction**] **temps**

where

- **format** *str* into [*'binaire'*, *'formatte'*, *'xyz'*]: 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.
- **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* (14.7): 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.

14.8 fonction_champ_reprise

Description: not_set

See also: objet_lecture (30)

Usage:

mot fonction

where

- **mot** *str* into ['fonction']
- **fonction** *n word1 word2 ... wordn*: n f1(val) f2(val) ... fn(val)] time

14.9 champ_fonc_t

Description: Field that is constant in space and is a function of time.

See also: champ_don_base (14.1)

Usage:

champ_fonc_t val

where

- **val** *n word1 word2 ... wordn*: Values of field components (time dependant functions).

14.10 champ_fonc_tabule

Description: Field that is tabulated as a function of another field.

See also: champ_don_base (14.1) champ_fonc_fonction (14.3)

Usage:

champ_fonc_tabule dim inco bloc

where

- **dim** *int*: Number of field components.
- **inco** *str*: Name of the field (for example: temperature).
- **bloc** *bloc_lecture* (2.37): 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)).

14.11 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 (14.1)

Usage:

champ_init_canal_sinal dim bloc

where

- **dim** *int*: Number of field components.
- **bloc** *bloc_lec_champ_init_canal_sinal* (14.11): Parameters for the class champ_init_canal_sinal.

14.12 bloc_lec_champ_init_canal_sinal

Description: Parameters for the class champ_init_canal_sinal.

in 2D:

$U = u_{cent} * y(2h - y) / h / h$

$V = ampli_bruit * rand + ampli_sin * \sin(\omega * x)$

rand: unpredictable value between -1 and 1.

in 3D:

$U = u_{cent} * y(2h - y) / h / h$

$V = ampli_bruit * rand1 + ampli_sin * \sin(\omega * x)$

$W = ampli_bruit * rand2$

rand1 and rand2: unpredictable values between -1 and 1.

See also: [objet_lecture \(30\)](#)

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 length 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 ZDefault 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 directionDefault 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.

14.13 champ_input_base

Description: not_set

See also: `champ_base` ([14](#)) `champ_input_p0` ([14.13](#))

Usage:

```
champ_input_base obj Lire obj {  
    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*

14.14 `champ_input_p0`

Description: `not_set`

See also: `champ_input_base` ([14.12](#))

Usage:

```
champ_input_p0 obj Lire obj {  
    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

14.15 `champ_ostwald`

Description: This keyword is used to define the viscosity variation law:
$$\mu(T) = K(T) \cdot (D:D/2)^{**}((n-1)/2)$$

See also: `champ_base` ([14](#))

Usage:

```
champ_ostwald
```

14.16 champ_som_lu_vdf

Description: Keyword to read in a file values located at the nodes of a mesh in VDF discretisation.

See also: champ_don_base (14.1)

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 ...

14.17 champ_som_lu_vef

Description: Keyword to read in a file values located at the nodes of a mesh in VEF discretisation.

See also: champ_don_base (14.1)

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 ...

14.18 champ_tabule_temps

Description: Field that is constant in space and tabulated as a function of time.

See also: champ_don_base (14.1)

Usage:

champ_tabule_temps dim bloc

where

- **dim** *int*: Number of field components.
- **bloc** *bloc_lecture* (2.37): Values as a table. The value of the field at any time is calculated by linear interpolation from this table.

14.19 champ_uniforme_morceaux

Description: Field which is partly constant in space and stationary.

See also: champ_don_base (14.1) champ_uniforme_morceaux_tabule_temps (14.19) valeur_totale_sur_volume (14.26)

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* (2.37): { 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.

14.20 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 (14.18)

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* (2.37): { 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.

14.21 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 (14.1)

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).

14.22 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 ([14.1](#))

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).

14.23 field_uniform_keps_from_ud

Description: field which allows to impose on a domain K and EPS values derived from U velocity and D hydraulic diameter

See also: champ_base ([14](#))

Usage:

field_uniform_keps_from_ud obj Lire obj {

u *float*

d *float*

}

where

- **u** *float*: value of velocity specified in boundary condition.
- **d** *float*: value of hydraulic diameter specified in boundary condition

14.24 init_par_partie

Description: ne marche que pour n_comp=1

See also: champ_don_base ([14.1](#))

Usage:

init_par_partie **n_comp** **val1** **val2** **val3**

where

- **n_comp** *int into [1]*
- **val1** *float*
- **val2** *float*
- **val3** *float*

14.25 tayl_green

Description: Class Tayl_green.

See also: champ_don_base ([14.1](#))

Usage:

tayl_green dim

where

- **dim** *int*: Dimension.

14.26 uniform_field

Description: Field that is constant in space and stationary.

See also: `champ_don_base` ([14.1](#))

Usage:

uniform_field val

where

- **val** *n x1 x2 ... xn*: Values of field components.

14.27 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` ([14.18](#))

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* ([2.37](#)): { Default 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` ([31](#)) `champ_front_uniforme` ([15.20](#)) `champ_front_fonc_xyz` ([15.12](#)) `champ_front_fonc_txyz` ([15.11](#)) `champ_front_fonc_pois_ipsn` ([15.9](#)) `champ_front_fonc_pois_tube` ([15.10](#)) `champ_front_tabule` ([15.18](#)) `champ_front_fonction` ([15.13](#)) `champ_front_bruite` ([15.5](#)) `champ_front_tangentiel_vef` ([15.19](#)) `champ_front_lu` ([15.14](#)) `boundary_field_inward` ([15.1](#)) `champ_front_pression_from_u` ([15.16](#)) `champ_front_debit` ([15.8](#)) `champ_front_contact_vef` ([15.7](#)) `champ_front_calc` ([15.6](#)) `champ_front_recyclage` ([15.17](#)) `ch_front_input` ([15.3](#)) `boundary_field_uniform_keps_from_ud` ([15.2](#)) `champ_front_normal_vef` ([15.15](#))

Usage:

15.2 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](#))

Usage:

boundary_field_inward obj Lire obj {

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.

15.3 boundary_field_uniform_keps_from_ud

Description: field which allows to impose on a boundary K and EPS values derived from U velocity and D hydraulic diameter

See also: `champ_front_base` ([15](#))

Usage:

boundary_field_uniform_keps_from_ud obj Lire obj {

u *float*

d *float*

}

where

- **u** *float*: value of velocity
- **d** *float*: value of hydraulic diameter

15.4 ch_front_input

Description: `not_set`

See also: `champ_front_base` ([15](#)) `ch_front_input_uniforme` ([15.4](#))

Usage:

ch_front_input obj Lire obj {

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*

15.5 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` ([15.3](#))

Usage:

ch_front_input_uniforme obj Lire obj {

```

    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

15.6 champ_front_bruite

Description: Field which is variable in time and space in a random manner.

See also: `champ_front_base` ([15](#))

Usage:

champ_front_bruite **nb_comp** **bloc**

where

- **nb_comp** *int*: Number of field components.
- **bloc** *bloc_lecture* ([2.37](#)): { [N val L val] Moyenne m_1.....[m_i] Amplitude A_1.....[A_i]}:
 Random noise: 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 $2\pi N/(4L)$.
 For example, formula for speed: $u=U0(t)$ $v=U1(t)Uj(t)=Mj+2\cdot Aj\cdot \text{bruit_blanc}$ 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).

15.7 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)

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 recognised by the problem_name object.

15.8 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)

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.

15.9 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 equation.

See also: champ_front_base (15)

Usage:

champ_front_debit ch

where

- **ch** *champ_front_base* (15): field (champ_front_uniforme) to define the flow rate.

15.10 champ_front_fonc_pois_ipsn

Description: Boundary field champ_front_fonc_pois_ipsn.

See also: champ_front_base (15)

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)*

15.11 champ_front_fonc_pois_tube

Description: Boundary field champ_front_fonc_pois_tube.

See also: champ_front_base ([15](#))

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)*

15.12 champ_front_fonc_txyz

Description: Boundary field which is not constant in space and in time.

See also: champ_front_base ([15](#))

Usage:

champ_front_fonc_txyz val

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

15.13 champ_front_fonc_xyz

Description: Boundary field which is not constant in space.

See also: champ_front_base ([15](#))

Usage:

champ_front_fonc_xyz val

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

15.14 champ_front_fonction

Description: boundary field that is function of another field

See also: champ_front_base ([15](#))

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.

15.15 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](#))

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

15.16 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](#))

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).

15.17 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](#))

Usage:

champ_front_pression_from_u expression

where

- **expression** *str*: value depending of a velocity (like $2 * u_{moy}^2$).

15.18 champ_front_recyclage

Description: This keyword is used on a boundary to get a field from another boundary. New keyword in 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
Champ_front_recyclage {
pb_champ_evaluateur pb field nb_comp
[ distance_plan dist0 dist1 [dist2] ]
[ moyenne_imposee methode_moy [fichier file [second_file] ]
[ moyenne_recyclee methode_recyc [fichier file [second_file] ]
[ direction_anisotrope 1|2|3 ]
[ ampli_moyenne_imposee 2|3 alpha(0) alpha(1) [alpha(2)] ]
[ ampli_moyenne_recyclee 2|3 beta(0) beta(1) [beta(2)] ]
[ ampli_fluctuation 2|3 gamma(0) gamma(1) [gamma(2)] ]
}

```

This keyword 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 $\langle f \rangle(t)$ or a temporal mean field $\langle f \rangle(x,y,z)$ field 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:

$$F_i(x,y,z,t) = \alpha_i g_i(x,y,z,t) + \chi_i [f_i(x,y,z,t) - \beta_i \langle f_i \rangle]$$

The different options are:

pb_champ_evaluateur pb field nb_comp : To give the name of the pb problem, the name of the field of the problem and its number of components nb_comp.

distance_plan dist0 dist1 [dist2] : 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)] : α_i coefficients (by default =1)

ampli_moyenne_recyclee 2|3 beta(0) beta(1) [beta(2)] : β_i coefficients (by default =1)

ampli_fluctuation 2|3 gamma(0) gamma(1) [gamma(2)] : γ_i coefficients (by default =1)

direction_anisotrope direction : 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 a imposed field built by interpolation of values read into 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 into a file where positions and values are given (it is not necessary that the coordinates of the points match the coordinates of the faces of the boundary, 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 into two files. The first file contains the points coordinates (which should be the same than the coordinates of each faces of the boundary) 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)]

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 double u_tau double visco_cin double direction integer : 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 set to 3)

moyenne_recylee methode_recyc : Method used to do 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

Same options of methode_moy options but applied to read a temporal mean field <f>(x,y,z):

interpolation

connexion_approchee fichier file

connexion_exacte fichier file second_file

See also: champ_front_base ([15](#))

Usage:

champ_front_recyclage bloc

where

- **bloc** *str*

15.19 champ_front_tabule

Description: Constant field on the boundary, tabulated as a function of time.

See also: champ_front_base ([15](#))

Usage:

champ_front_tabule nb_comp bloc

where

- **nb_comp** *int*: Number of field components.

- **bloc** *bloc_lecture* (2.37): {nt1 t2 t3tn 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.

15.20 champ_front_tangentiel_vef

Description: Field to define the tangential speed vector field standard at the boundary in VEF discretisation.

See also: *champ_front_base* (15)

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].

15.21 champ_front_uniforme

Description: Boundary field which is constant in space and stationary.

See also: *champ_front_base* (15)

Usage:

champ_front_uniforme val

where

- **val** *n x1 x2 ... xn*: Values of field components.

16 loi_etat_base

Description: Basic class for state laws.

See also: *objet_u* (31) *gaz_parfait* (16.2) *melange_gaz_parfait* (16.1) *gaz_reel_rhot* (16)

Usage:

16.1 gaz_reel_rhot

Description: Real gas.

See also: *loi_etat_base* (16)

Usage:

gaz_reel_rhot bloc

where

- **bloc** *bloc_lecture* (2.37): Description.

16.2 melange_gaz_parfait

Description: Mixing of perfect gas.

See also: `loi_etat_base` ([16](#))

Usage:

melange_gaz_parfait obj Lire obj {

 [**Sc** *float*]
 Prandtl *float*

}

where

- **Sc** *float*: Schmidt number of the gas $Sc = \nu/D$ (D: diffusion coefficient of the mixing).
- **Prandtl** *float*: Prandtl number of the gas $Pr = \mu * Cp / \lambda$

16.3 gaz_parfait

Description: Perfect gas.

See also: `loi_etat_base` ([16](#))

Usage:

gaz_parfait obj Lire obj {

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](#))

17 loi_horaire

Description: to define the movement with a time-dependant law for the solid interface.

See also: `objet_u` ([31](#))

Usage:

loi_horaire obj Lire obj {

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*

18 milieu_base

Description: Basic class for medium (physics properties of medium).

See also: objet_u (31) solide (18.5) constituant (18) fluide_incompressible (18.1)

Usage:

```
milieu_base obj Lire obj {  
    [ rho champ_base]  
    [ cp champ_base]  
    [ lambda champ_base]  
}
```

where

- **rho** *champ_base* (14): Density (kg.m-3).
- **cp** *champ_base* (14): Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (14): Conductivity (W.m-1.K-1).

18.1 constituant

Description: Constituent.

See also: milieu_base (18)

Usage:

```
constituant obj Lire obj {  
    [ coefficient_diffusion champ_base]  
    [ rho champ_base]  
    [ cp champ_base]  
    [ lambda champ_base]  
}
```

where

- **coefficient_diffusion** *champ_base* (14): Constituent diffusion coefficient value (m2.s-1). If a multi-constituent problem is being processed, the diffusivity will be a vectorial and each components will be the diffusion of the constituent.
- **rho** *champ_base* (14) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (14) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (14) for inheritance: Conductivity (W.m-1.K-1).

18.2 fluide_incompressible

Description: This is a uncompressible fluid.

See also: milieu_base (18) fluide_quasi_compressible (18.3) fluide_ostwald (18.2)

Usage:

fluide_incompressible obj Lire obj {

```
[ beta_th champ_base]
[ mu champ_base]
[ beta_co champ_base]
[ indice champ_base]
[ kappa champ_base]
[ rho champ_base]
[ cp champ_base]
[ lambda champ_base]
```

}

where

- **beta_th** champ_base (14): Thermal expansion (K-1).
- **mu** champ_base (14): Dynamic viscosity (kg.m-1.s-1).
- **beta_co** champ_base (14): Volume expansion coefficient values in concentration.
- **indice** champ_base (14): Refractivity of fluid.
- **kappa** champ_base (14): Absorptivity of fluid (m-1).
- **rho** champ_base (14) for inheritance: Density (kg.m-3).
- **cp** champ_base (14) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** champ_base (14) for inheritance: Conductivity (W.m-1.K-1).

18.3 fluide_ostwald

Description: Non-Newtonian fluids governed by Ostwald's law. The law applicable to stress tensor is:

$\tau = K(T) \cdot (D:D/2)^{((n-1)/2)} \cdot D$ Where:

D refers to the deformation speed 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 (18.1)

Usage:

fluide_ostwald obj Lire obj {

```
[ k champ_base]
[ n champ_base]
[ beta_th champ_base]
[ mu champ_base]
[ beta_co champ_base]
[ indice champ_base]
[ kappa champ_base]
[ rho champ_base]
[ cp champ_base]
[ lambda champ_base]
```

```
}
where
```

- **k** *champ_base* (14): Fluid consistency.
- **n** *champ_base* (14): Fluid structure index.
- **beta_th** *champ_base* (14) for inheritance: Thermal expansion (K-1).
- **mu** *champ_base* (14) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **beta_co** *champ_base* (14) for inheritance: Volume expansion coefficient values in concentration.
- **indice** *champ_base* (14) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (14) for inheritance: Absorptivity of fluid (m-1).
- **rho** *champ_base* (14) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (14) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (14) for inheritance: Conductivity (W.m-1.K-1).

18.4 fluide_quasi_compressible

Description: Compressible flow at low mach number.

See also: fluide_incompressible (18.1)

Usage:

```
fluide_quasi_compressible obj Lire obj {
    [ 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]
    [ mu champ_base]
    [ indice champ_base]
    [ kappa champ_base]
    [ rho champ_base]
    [ cp champ_base]
    [ lambda champ_base]
}
```

where

- **sutherland** *bloc_sutherland* (18.4): Sutherland law for viscosity and for conductivity.
- **pression** *float*: Initial pression.
- **loi_etat** *loi_etat_base* (16): State law.
- **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).
- **traitement_rho_gravite** *str into ['standard', 'moins_rho_moyen']*: It may be :1) 'standard': the gravity term is evaluated with $\rho \cdot g$ (It is the default). 2) 'moins_rho_moyen': the gravity term is evaluated with $(\rho - \rho_{moy}) \cdot g$.

- **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.
- **mu** *champ_base* (14) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **indice** *champ_base* (14) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (14) for inheritance: Absorptivity of fluid (m-1).
- **rho** *champ_base* (14) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (14) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (14) for inheritance: Conductivity (W.m-1.K-1).

18.5 bloc_sutherland

Description: Sutherland law for viscosity $\mu(T)=\mu_0*((T_0+C)/(T+C))*(T/T_0)**1.5$ and (optional) for conductivity $\lambda(T)=\mu_0*C_p/Prandtl*((T_0+S\lambda)/(T+S\lambda))*(T/T_0)**1.5$

See also: [objet_lecture \(30\)](#)

Usage:

m mu0 t t0 [ms] [s] mc c

where

- **m** *str* into ['mu0']
- **mu0** *float*
- **t** *str* into ['T0']
- **t0** *float*
- **ms** *str* into ['Slambda']
- **s** *float*
- **mc** *str* into ['C']
- **c** *float*

18.6 solide

Description: Solid.

See also: [milieu_base \(18\)](#)

Usage:

solide obj Lire obj {

 [**rho** *champ_base*]
 [**cp** *champ_base*]
 [**lambda** *champ_base*]

}

where

- **rho** *champ_base* (14) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (14) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (14) for inheritance: Conductivity (W.m-1.K-1).

19 modele_turbulence_scal_base

Description: Basic class for turbulence model for energy equation.

See also: objet_u (31) prandtl (19) schmidt (19.1)

Usage:

```
modele_turbulence_scal_base obj Lire obj {  
    [ turbulence_paro turbulence_paro_scalaire_base]  
    [ dt_impr_nusselt float]  
}  
where
```

- **turbulence_paro** turbulence_paro_scalaire_base (28): 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».

19.1 prandtl

Description: The Prandtl model. For the scalar equations, only the model based on Reynolds analogy is available. If K_Epsilon was selected in the hydraulic equation, Prandtl must be selected for the convection-diffusion temperature equation coupled to the hydraulic equation and Schmidt for the concentration equations.

See also: modele_turbulence_scal_base (19)

Usage:

```
prandtl obj Lire obj {  
    [ prdt str]  
    [ prandt_turbulent_fonction_nu_t_alpha str]  
    [ turbulence_paro turbulence_paro_scalaire_base]  
    [ dt_impr_nusselt float]  
}  
where
```

- **prdt** str: Keyword to modify the constant (Prdt) of Prandtl model : $Alphat = \text{Nut} / \text{Prdt}$ Default value is 0.9
- **prandt_turbulent_fonction_nu_t_alpha** str: Optional keyword to specify turbulent diffusivity (by default, $\alpha_t = \text{nu}_t / \text{Prt}$) with another formulae, for example: $\alpha_t = \text{nu}_t^2 / (0.7 * \alpha + 0.85 * \text{nu}_t)$ with the string $\text{nu}_t * \text{nu}_t / (0.7 * \alpha + 0.85 * \text{nu}_t)$ where α is the thermal diffusivity.
- **turbulence_paro** turbulence_paro_scalaire_base (28) for inheritance: Keyword to set the wall law.

- **dt_impr_nusselt** *float* for inheritance: 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».

19.2 schmidt

Description: The Schmidt model. For the scalar equations, only the model based on Reynolds analogy is available. If `K_Epsilon` was selected in the hydraulic equation, `Prandtl` must be selected for the convection-diffusion temperature equation coupled to the hydraulic equation and `Schmidt` for the concentration equations.

See also: `modele_turbulence_scal_base` (19)

Usage:

```
schmidt obj Lire obj {
    [ seturb float ]
    [ turbulence_pari turbulence_pari_scalaire_base ]
    [ dt_impr_nusselt float ]
}
```

where

- **seturb** *float*: Keyword to modify the constant (`Sct`) of Schmidt model : $Dt = \text{Nut} / \text{Sct}$ Default value is 0.7.
- **turbulence_pari** *turbulence_pari_scalaire_base* (28) for inheritance: Keyword to set the wall law.
- **dt_impr_nusselt** *float* for inheritance: 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».

20 nom

Description: Class to name the TRUST objects.

See also: `objet_u` (31) `nom_anonyme` (20)

Usage:

```
nom [ mot ]
where
```

- **mot** *str*: Chain of characters.

20.1 nom_anonyme

Description: not_set

See also: nom (20)

Usage:

[**mot**]

where

- **mot** *str*: Chain of characters.

21 partitionneur_deriv

Description: not_set

See also: objet_u (31) metis (21.1) sous_zones (21.3) tranche (21.4) partition (21.2) fichier_decoupage (21)

Usage:

partitionneur_deriv obj Lire obj {

 [**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).

21.1 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 $n \geq 0$ 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 (21)

Usage:

fichier_decoupage obj Lire obj {

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).

21.2 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` (21)

Usage:

```
metis obj Lire obj {
    [ 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).

21.3 partition

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` (21)

Usage:

```
partition obj Lire obj {
    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).

21.4 sous_zones

Description: This algorithm will create one part for each specified subzone. All elements contained in the first subzone are put in the first part, all remaining elements contained in the second subzone in the second part, etc...

If all elements of the domain are contained in the specified subzones, 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` (21)

Usage:

sous_zones obj Lire obj {

sous_zones *n word1 word2 ... wordn*
 [**nb_parts** *int*]

}

where

- **sous_zones** *n word1 word2 ... wordn*: N SUBZONE_NAME_1 SUBZONE_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).

21.5 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` (21)

Usage:

tranche obj Lire obj {

 [**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).

22 **precond_base**

Description: Basic class for preconditioning.

See also: `objet_u` (31) `ssor` (22.2) `ssor_bloc` (22.3) `precondsolv` (22.1) `precond_local` (22)

Usage:

22.1 **precond_local**

Description: This keyword can be used with the conjugate gradient (GCP) to choose a local preconditionment for parallel calculation (ie: Cholesky, SSOR,...).

See also: `precond_base` (22)

Usage:

precond_local **solveur**

where

- **solveur** *solveur_sys_base* (8.11): Solver type.

22.2 **precondsolv**

Description: `not_set`

See also: `precond_base` (22)

Usage:

precondsolv **solveur**

where

- **solveur** *solveur_sys_base* (8.11): Solver type.

22.3 **ssor**

Description: Symmetric successive over-relaxation algorithm.

See also: `precond_base` (22)

Usage:

ssor `obj Lire obj {`

omega *float*

`}`

where

- **omega** *float*: Over-relaxation facteur (between 1 and 2, optimal value around 1.5-1.6).

22.4 ssor_bloc

Description: not_set

See also: [precond_base \(22\)](#)

Usage:

```
ssor_bloc obj Lire obj {  
    [ 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* ([22](#))
- **alpha_1** *float*
- **precond1** *precond_base* ([22](#))
- **alpha_a** *float*
- **preconda** *precond_base* ([22](#))

23 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 \(31\)](#) [scheme_euler_explicit \(23.2\)](#) [schema_predictor_corrector \(23.15\)](#) [Sch_CN_iteratif \(23.1\)](#) [runge_kutta_ordre_3 \(23.4\)](#) [runge_kutta_ordre_4_d3p \(23.5\)](#) [leap_frog \(23.3\)](#) [runge_kutta_rationnel_ordre_2 \(23.6\)](#) [schema_implicite_base \(23.14\)](#) [schema_adams_bashforth_order_2 \(23.7\)](#) [schema_adams_bashforth_order_3 \(23.8\)](#)

Usage:

```
schema_temps_base obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max float]  
    [ facsec float]  
    [ nb_pas_dt_max int]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ dt_start dt_start]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int into [0, 1]]  
    [ diffusion_implicite int into [0, 1]]  
    [ niter_max_diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int into [0, 1]]
```

```

[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
where

```

- **tinit** *float*: Value of initial calculation time (0 by default).
- **tmax** *float*: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *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** *float*: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int*: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4): 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float*: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi 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 into [0, 1]*
- **diffusion_implicit** *int into [0, 1]*: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int*: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float*: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]*: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int*: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]*
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]*

- **periode_sauvegarde_securite_en_heures** *int*: 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.

23.1 Sch_CN_EX_iteratif

Description: This keyword also describes a Crank-Nicholson method of second order accuracy but here, for scalars, because of instabilities 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 $dt \leq dt_{CFL}$). Parameters are the same (but default values may change) compare to the Sch_CN_iterative scheme plus a relaxation keyword: niter_min (2), niter_max (6), niter_avg (3), facsec_max (20), seuil (0.05)

See also: Sch_CN_iteratif ([23.1](#))

Usage:

Sch_CN_EX_iteratif obj Lire obj {

```
[ 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 float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicite int into [0, 1]]
[ niter_max_diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
[ no_conv_subiteration_diffusion_implicite int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

where

- **omega** *float*: relaxation factor (0.1)
- **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)
- **niter_avg** *int* for inheritance: threshold of p-iterations (3). 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).
- **seuil** *float* for inheritance: criteria for ending iterative process ($\text{Max}(\|u(p) - u(p-1)\|/\text{Max} \|u(p)\|) < \text{seuil}$) (0.001)
- **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).
- **tcputmax** *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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step ($\text{dt}=\text{facsec}*\text{dt_convection}$). 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $\text{dt}=\text{facsec}*\text{dt_max}$.
- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.

- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.2 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` (23) `Sch_CN_EX_iteratif` (23)

Usage:

Sch_CN_iteratif obj Lire obj {

```
[ 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     float]
[ facsec     float]
[ nb_pas_dt_max int]
[ dt_sauv    float]
[ dt_impr    float]
[ dt_start   dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

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)
- **niter_avg** *int*: threshold of p-iterations (3). 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).
- **seuil** *float*: criteria for ending iterative process ($\text{Max}(\|u(p) - u(p-1)\|/\text{Max}\|u(p)\|) < \text{seuil}$) (0.001)
- **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).
- **tcputmax** *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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step ($dt = \text{facsec} * dt_{\text{convection}}$). 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $dt = \text{facsec} * dt_{\text{max}}$.
- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.

- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicite** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.3 scheme_euler_explicit

Description: This is the Euler explicite scheme.

See also: `schema_temps_base` (23)

Usage:

scheme_euler_explicit obj Lire obj {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicite int into [0, 1]]
[ niter_max_diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
[ no_conv_subiteration_diffusion_implicite int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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

- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.4 leap_frog

Description: This is the leap-frog scheme.

See also: [schema_temps_base \(23\)](#)

Usage:

leap_frog obj Lire obj {

[**tinit** *float*]

[**tmax** *float*]

```

[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel 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 should avoid exceeding the calculation convection time step by selecting a `facsec` that is too large. Start with a `facsec` of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore `dt=facsec*dt_max`.

- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.5 runge_kutta_ordre_3

Description: This is the Runge-Kutta scheme of third order.

See also: [schema_temps_base \(23\)](#)

Usage:

```
runge_kutta_ordre_3 obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicit int into [0, 1]]
    [ niter_max_diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicit int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
```

}
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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.6 runge_kutta_ordre_4_d3p

Description: not_set

See also: schema_temps_base (23)

Usage:

```
runge_kutta_ordre_4_d3p obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max float]  
    [ facsec float]  
    [ nb_pas_dt_max int]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ dt_start dt_start]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int into [0, 1]]  
    [ diffusion_implicite int into [0, 1]]  
    [ niter_max_diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int into [0, 1]]  
    [ precision_impr int]  
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]  
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
}
```

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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calcula-

tion with Crank Nicholson temporal scheme to ensure continuity).

By default, the first iteration is based on `dt_calc`.

- **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 dG_i/G_i of all the unknown transported values G_i 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, `vrel` 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 should avoid exceeding the calculation convection time step by selecting a `facsec` that is too large. Start with a `facsec` of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_max$.
- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into `.out` files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.7 runge_kutta_rationnel_ordre_2

Description: This is the Runge-Kutta rational scheme of second order.

See also: `schema_temps_base` (23)

Usage:

```
runge_kutta_rationnel_ordre_2 obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max float]  
    [ facsec float]  
    [ nb_pas_dt_max int]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ dt_start dt_start]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int into [0, 1]]
```

```

[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]

```

}

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).
- **tcputmax** *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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi 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* into [0, 1] for inheritance
- **diffusion_implicit** *int* into [0, 1] for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.8 schema_adams_bashforth_order_2

Description: not_set

See also: schema_temps_base (23)

Usage:

schema_adams_bashforth_order_2 obj Lire obj {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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

- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.9 schema_adams_bashforth_order_3

Description: not_set

See also: schema_temps_base (23)

Usage:

```
schema_adams_bashforth_order_3 obj Lire obj {
    [ tinit float]
```

```

[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
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** float for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** int for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** dt_start (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi 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 into [0, 1] for inheritance
- **diffusion_implicit** int into [0, 1] for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_max$.

- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.10 schema_adams_moulton_order_2

Description: not_set

See also: schema_implicit_base ([23.14](#))

Usage:

```
schema_adams_moulton_order_2 obj Lire obj {
    [ facsec_max float]
    [ max_iter_implicit int]
    solveur solveur_implicit_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicit int into [0, 1]]
    [ niter_max_diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicit int into [0, 1]]
}
```

```
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

```
}
```

where

- **facsec_max** *float* for inheritance: 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
-Thermohydraulic 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_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (24) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solveur* 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. 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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.

- **dt_start** *dt_start* (8.4) for inheritance: **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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on **dt_calc**.
- **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 dG_i/G_i of all the unknown transported values G_i 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, **vrel** 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 should avoid exceeding the calculation convection time step by selecting a **facsec** that is too large. Start with a **facsec** of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_max$.
- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.11 schema_adams_moulton_order_3

Description: not_set

See also: [schema_implicit_base](#) (23.14)

Usage:

schema_adams_moulton_order_3 obj Lire obj {

```
[ facsec_max float]
[ max_iter_implicit int]
solveur solveur_implicit_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
```

```

[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicite int into [0, 1]]
[ niter_max_diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
[ no_conv_subiteration_diffusion_implicite int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]

```

}

where

- **facsec_max** *float* for inheritance: 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
-Thermohydraulic 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* (24) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solveur* 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. 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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi 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 into [0, 1]* for inheritance
- **diffusion_implicite** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **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.
- **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 into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicite** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.12 schema_backward_differentiation_order_2

Description: not_set

See also: schema_implicite_base (23.14)

Usage:

schema_backward_differentiation_order_2 obj Lire obj {

```
[ facsec_max float]
[ max_iter_implicit int]
solveur solveur_implicit_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

where

- **facsec_max** float for inheritance: 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
- Thermohydraulic 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_implicit** int for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (24) 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. 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 Simplr. Because the two first give a fastest convergence (several times) than Piso and the Simplr 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).
- **tcputmax** *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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi 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 into [0, 1]* for inheritance
- **diffusion_implicite** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $dt = facsec * dt_{max}$.
- **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.
- **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 into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).

- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.13 schema_backward_differentiation_order_3

Description: not_set

See also: `schema_implicit_base` (23.14)

Usage:

schema_backward_differentiation_order_3 obj Lire obj {

```
[ facsec_max float]
[ max_iter_implicit int]
solveur solveur_implicit_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

where

- **facsec_max** *float* for inheritance: 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

-Thermohydraulic 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_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (24) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solveur* 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. 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).
- **tcumax** *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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: *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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on *dt_calc*.
- **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 dG_i/G_i of all the unknown transported values G_i 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, *vrel* 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 should avoid exceeding the

calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_max$.

- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.14 scheme_euler_implicit

Description: This is the Euler implicite scheme.

See also: `schema_implicite_base` ([23.14](#))

Usage:

```
scheme_euler_implicit obj Lire obj {
    [ facsec_max float]
    [ max_iter_implicit int]
    solveur solveur_implicit_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicit int into [0, 1]]
    [ niter_max_diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicit int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
```

}
where

- **facsec_max** *float* for inheritance: 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
 - Thermohydraulic with natural convection, facsec around 300
 - Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stableThese values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.
- **max_iter_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (24) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. solveur 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. 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).
- **tcputmax** *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** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).

By default, the first iteration is based on dt_calc.

- **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/Gi 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.15 schema_implicit_base

Description: Basic class for implicit time scheme.

See also: [schema_temps_base \(23\)](#) [scheme_euler_implicit \(23.13\)](#) [schema_adams_moulton_order_2 \(23.9\)](#) [schema_adams_moulton_order_3 \(23.10\)](#) [schema_backward_differentiation_order_2 \(23.11\)](#) [schema_backward_differentiation_order_3 \(23.12\)](#)

Usage:

```
schema_implicit_base obj Lire obj {
    [ facsec_max float]
    [ max_iter_implicit int]
    solveur solveur_implicit_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
```



```

[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicite int into [0, 1]]
[ niter_max_diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
[ no_conv_subiteration_diffusion_implicite int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}

```

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
- Thermohydraulic 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*: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (24): This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solveur* 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. 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** *float* for inheritance: Maximum calculation time step (1e30s by default).

- **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
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** *dt_start* (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel 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 should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

23.16 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 (23)

Usage:

schema_predictor_corrector obj Lire obj {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

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** float for inheritance: Maximum calculation time step (1e30s by default).
- **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
- **nb_pas_dt_max** int for inheritance: Maximum number of calculation time steps (1e9).
- **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.
- **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.
- **dt_start** dt_start (8.4) for inheritance: 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 restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **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/Gi of all the unknown transported

values G_i 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 into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, *vrel* 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 should avoid exceeding the calculation convection time step by selecting a *facsec* that is too large. Start with a *facsec* of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_max$.
- **niter_max_diffusion_implicit** *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.
- **seuil_diffusion_implicit** *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_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* 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.

24 solveur_implicit_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 \(31\)](#) [solveur_lineaire_std \(24.4\)](#) [simpler \(24.3\)](#)

Usage:

24.1 implicit

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 \(24.1\)](#)

Usage:

```
implicit obj Lire obj {
    [ seuil_convergence_implicit 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_implicit** *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 than 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* (8.11) 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.

24.2 piso

Description: Piso (Pressure Implicit with Split Operator) - method to solve N_S.

See also: simpler (24.3) implicite (24) simple (24.2)

Usage:

```

piso obj Lire obj {
    [ seuil_convergence_implicit 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 than `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* (8.11) 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.

24.3 simple

Description: SIMPLE type algorithm

See also: `piso` (24.1)

Usage:

```
simple obj Lire obj {
    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 than `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* (8.11) 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.

24.4 simplr

Description: Simpler method for incompressible systems.

See also: `solveur_implicite_base` (24) `piso` (24.1)

Usage:

```
simplr obj Lire obj {
    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 advised 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* (8.11): 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.

24.5 solveur_lineaire_std

Description: not_set

See also: *solveur_implicite_base* (24)

Usage:

```
solveur_lineaire_std obj Lire obj {
    [ solveur solveur_sys_base ]
}
```

where

- **solveur** *solveur_sys_base* (8.11)

25 source_base

Description: Basic class of source terms introduced in the equation.

See also: *objet_u* (31) *source_generique* (25.18) *boussinesq_temperature* (25.3) *boussinesq_concentration* (25.2) *dirac* (25.7) *puissance_thermique* (25.16) *source_qdm_lambdaup* (25.20) *source_th_tdivu* (25.24) *source_robin* (25.21) *source_robin_scalaire* (25.22) *canal_perio* (25.4) *source_constituant* (25.17) *source_transport_k_eps* (25.25) *acceleration* (25.1) *coriolis* (25.5) *source_qdm* (25.19) *perte_charge_singuliere* (25.15.2) *perte_charge_directionnelle* (25.11) *perte_charge_isotrope* (25.12) *perte_charge_anisotrope* (25.9) *perte_charge_circulaire* (25.10) *darcy* (25.6) *forchheimer* (25.8) *perte_charge_reguliere* (25.13)

Usage:

25.1 Source_Transport_K_Eps_anisotherme

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transportation equation. By default, these constants are set to: $C1_eps=1.44$ $C2_eps=1.92$ $C3_eps=1.0$

See also: *source_transport_k_eps* (25.25)

Usage:

```
Source_Transport_K_Eps_anisotherme obj Lire obj {
    [ c3_eps float ]
    [ c1_eps float ]
    [ c2_eps float ]
}
```

where

- **c3_eps** *float*: Third constant.
- **c1_eps** *float* for inheritance: First constant.
- **c2_eps** *float* for inheritance: Second constant.

25.2 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 \(25\)](#)

Usage:

```
acceleration obj Lire obj {
    [ 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): Keyword for the velocity of the referential R' into the R referential ($d\mathbf{OO'}/dt$ 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): Keyword for the acceleration of the referential R' into the R referential ($d^2\mathbf{OO'}/dt^2$ term [m.s-2]). *field_base* is a time dependant field (eg: `Champ_Fonc_t`).
- **omega** *champ_base* (14): 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): 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): 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.

25.3 boussinesq_concentration

Description: Class to describe a source term that couples the movement quantity equation and constituent transportation equation with the Boussinesq hypothesis.

See also: [source_base \(25\)](#)

Usage:

```
boussinesq_concentration obj Lire obj {
    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.

25.4 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 \(25\)](#)

Usage:

```
boussinesq_temperature obj Lire obj {
```

```

    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.

25.5 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 restarting 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 \(25\)](#)

Usage:

```
canal_perio obj Lire obj {
```



```

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 height 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 slightly changed to verify incompressibility.

25.6 coriolis

Description: Keyword for a Coriolis term in hydraulic equation. Warning: Only available in VDF.

See also: [source_base \(25\)](#)

Usage:
coriolis **omega**
 where

- **omega** *str*: Value of omega.

25.7 darcy

Description: Class for calculation in a porous media with source term of Darcy $-\nu/K \cdot V$. 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 added for porosity (porosite).

See also: [source_base \(25\)](#)

Usage:
darcy **bloc**
 where

- **bloc** *bloc_lecture* ([2.37](#)): Description.

25.8 dirac

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: [source_base \(25\)](#)

Usage:
dirac **position** **ch**
 where

- **position** *n x1 x2 ... xn*

- **ch** *champ_base* (14): 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.

25.9 forchheimer

Description: Class to add the source term of Forchheimer $-C_f/\sqrt{K} \cdot V^2$ in the Navier Stokes equations. We must precise a permeability model : constant or Ergun's law. Moreover we can give the constant C_f : by default its value is 1. Forchheimer source term is available also for quasi compressible calculation. A new keyword is added for porosity (*porosite*).

See also: *source_base* (25)

Usage:

forchheimer bloc

where

- **bloc** *bloc_lecture* (2.37): Description.

25.10 perte_charge_anisotrope

Description: Anisotropic pressure loss.

See also: *source_base* (25)

Usage:

perte_charge_anisotrope obj Lire obj {

```

    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* (14.1): Hydraulic diameter value.
- **direction** *champ_don_base* (14.1): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

25.11 perte_charge_circulaire

Description: New pressure loss.

See also: *source_base* (25)

Usage:

perte_charge_circulaire obj Lire obj {

```

    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 $f(\text{Re}_{\text{tot}}, \text{Re}_{\text{long}}, t, x, y, z)$ for loss coefficient in the longitudinal direction
- **lambda_ortho** *str*: function: Function $f(\text{Re}_{\text{tot}}, \text{Re}_{\text{ortho}}, t, x, y, z)$ for loss coefficient in transverse direction
- **diam_hydr** *champ_don_base* (14.1): Hydraulic diameter value.
- **diam_hydr_ortho** *champ_don_base* (14.1): Transverse hydraulic diameter value.
- **direction** *champ_don_base* (14.1): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

25.12 perte_charge_directionnelle

Description: Directional pressure loss.

See also: [source_base \(25\)](#)

Usage:

perte_charge_directionnelle obj Lire obj {

```

    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: $64/\text{Re}$).
- **diam_hydr** *champ_don_base* (14.1): Hydraulic diameter value.
- **direction** *champ_don_base* (14.1): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

25.13 perte_charge_isotrope

Description: Isotropic pressure loss.

See also: [source_base \(25\)](#)

Usage:

perte_charge_isotrope obj Lire obj {

```

    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* (14.1): Hydraulic diameter value.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

25.14 perte_charge_reguliere

Description: Source term modelling the presence of a bundle of tubes in a flow.

See also: [source_base](#) (25)

Usage:

perte_charge_reguliere spec zone_name

where

- **spec** *spec_pdc_base* (25.14): 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.

25.15 spec_pdc_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](#) (30) [longitudinale](#) (25.15) [transversale](#) (25.15.1)

Usage:

spec_pdc_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.

25.15.1 longitudinale

Description: Class to define the pressure loss in the direction of the tube bundle.

See also: [spec_pdc_base](#) (25.14)

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.

25.15.2 transversale

Description: Class to define the pressure loss in the direction perpendicular to the tube bundle.

See also: `spec_pdcrr_base` (25.14)

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.

25.16 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` (25)

Usage:

perte_charge_singuliere **dir** **coeff** **bloc_definition_surface**

where

- **dir** *str* into ['kx', 'ky', 'kz']: KX, KY or KZ designate directional pressure loss coefficients for respectively X, Y or Z direction.
- **coeff** *float*: Value of friction coefficient (KX, KY, KZ).
- **bloc_definition_surface** *bloc_lecture* (2.37): Surface definition block : In VDF, the surface area definition syntax is identical to that used to define sides (edges) in the Block, for example { X = x0 y0 <= Y <= y1 } for a line perpendicular to the Ox axis in a two-dimensional domain, or { Y = y0 x0 <= X <= x1 z0 <= Z <= z1 } for a surface perpendicular to the Oy axis in a 3D domain.
example : `sources { Perte_Charge_Singuliere KX 0.5 { X = 1. 0. <= Y <= 1. } }`

VEF : the surface area definition syntax relies on sub-areas definition (see 4.3.22). First value (X=0.35 in the example below, in regard to KX keyword) allows to determine the faces of elements in sub-area for which the pressure loss is applied.

example : `sources { Perte_Charge_Singuliere KX 0.5 { 0.35 sous_zone_toto } }`

Observations :

- If the surface area is not included in the calculation domain or if (in VDF) it is not perpendicular to the space direction in accordance with which the pressure loss is being calculated, Trio-U exists in error.
- The surface area may be diminished at only one side if a sudden shrinking or widening occurs.

25.17 **puissance_thermique**

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: [source_base \(25\)](#)

Usage:

puissance_thermique ch

where

- **ch** *champ_base* (14): 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. It is a power per volume unit (in a porous media, it is a power per fluid volume unit).

25.18 **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 \(25\)](#)

Usage:

source_constituant ch

where

- **ch** *champ_base* (14): Field type.

25.19 **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 \(25\)](#)

Usage:

source_generique champ

where

- **champ** *champ_generique_base* (6): the source field

25.20 **source_qdm**

Description: Momentum source term in the Navier Stokes equation.

See also: [source_base \(25\)](#)

Usage:

source_qdm ch

where

- **ch** *champ_base* (14): Field type.

25.21 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: $du'/dt = -\lambda u' + \text{grad } P'$ 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 \(25\)](#)

Usage:

source_qdm_lambdaup obj Lire obj {

```
    lambda float  
    [ lambda_min float]  
    [ lambda_max float]  
    [ ubar_umprim_cible float]
```

}

where

- **lambda** *float*: value of λ
- **lambda_min** *float*: value of λ_{\min}
- **lambda_max** *float*: value of λ_{\max}
- **ubar_umprim_cible** *float*: value of $\bar{u}_{\text{umprim_cible}}$

25.22 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 τ_w , u_{τ} and Reynolds_{τ} 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 \(25\)](#)

Usage:

source_robin bords

where

- **bords** *vect_nom* ([2.97](#))

25.23 source_robin_scalaire

Description: This source term should be used when a `Paroi_decalee_Robin` boundary condition is set in 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 I th 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 \(25\)](#)

Usage:

source_robin_scalaire bords

where

- **bords** *listdeuxmots_sacc* ([25.23](#))

25.24 listdeuxmots_sacc

Description: List of groups of two words (without accodances).

See also: listobj ([29.2](#))

Usage:

n object1 object2

list of *deuxmots* ([4.19](#))

25.25 source_th_tdivu

Description: This term source is dedicated for any scalar (called T) transportation. Coupled with upwind (amont) or muscl scheme, this term gives for final expression of convection : $\text{div}(\mathbf{U} \cdot \mathbf{T}) - \mathbf{T} \cdot \text{div}(\mathbf{U}) = \mathbf{U} \cdot \text{grad}(\mathbf{T})$. 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 ([25](#))

Usage:

source_th_tdivu

25.26 source_transport_k_eps

Description: Keyword to alter the source term constants in the standard k-eps model epsilon transportation equation. By default, these constants are set to: C1_eps=1.44 C2_eps=1.92

See also: source_base ([25](#)) Source_Transport_K_Eps_anisotherme ([25](#)) source_transport_k_eps_aniso_concen ([25.26](#)) source_transport_k_eps_aniso_therm_concen ([25.27](#))

Usage:

source_transport_k_eps obj Lire obj {

[**c1_eps** *float*]

[**c2_eps** *float*]

}

where

- **c1_eps** *float*: First constant.
- **c2_eps** *float*: Second constant.

25.27 source_transport_k_eps_aniso_concen

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transportation equation. By default, these constants are set to: C1_eps=1.44 C2_eps=1.92 C3_eps=1.0

See also: source_transport_k_eps ([25.25](#))

Usage:

source_transport_k_eps_aniso_concen obj Lire obj {

[**c3_eps** *float*]

[**c1_eps** *float*]


```

    [ c2_eps float]
}
where

```

- **c3_eps** *float*: Third constant.
- **c1_eps** *float* for inheritance: First constant.
- **c2_eps** *float* for inheritance: Second constant.

25.28 source_transport_k_eps_aniso_therm_concen

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transportation equation. By default, these constants are set to: C1_eps=1.44 C2_eps=1.92 C3_eps=1.0

See also: [source_transport_k_eps \(25.25\)](#)

Usage:

source_transport_k_eps_aniso_therm_concen obj Lire obj {

```

    [ c3_eps float]
    [ c1_eps float]
    [ c2_eps float]
}

```

where

- **c3_eps** *float*: Third constant.
- **c1_eps** *float* for inheritance: First constant.
- **c2_eps** *float* for inheritance: Second constant.

26 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 Lire (Read) interpreter 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 Associer (Associate) `nom_sous_zone nom_domaine` instruction; this instruction must always be preceded by the read instruction.

See also: [objet_u \(31\)](#)

Usage:

sous_zone obj Lire obj {

```

    [ 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 Lire keyword.
- **rectangle** *bloc_origine_cotes* (26): The sub-area will include all the domain elements whose centre of gravity is within the Rectangle (in dimension 2).
- **segment** *bloc_origine_cotes* (26)
- **boite** *bloc_origine_cotes* (26): 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* (26.1): The sub-area will include domain items whose number is between *n1* and *n2* (where $n1 \leq n2$).
- **polynomes** *bloc_lecture* (2.37): A REPENDRE
- **couronne** *bloc_couronne* (26.2): In 2D case, to create a couronne.
- **tube** *bloc_tube* (26.3): 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 Lire keyword.

26.1 bloc_origine_cotes

Description: Class to create a rectangle (or a box).

See also: *objet_lecture* (30)

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)*: Co-ordinates 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.

26.2 deuxentiers

Description: Two integers.

See also: *objet_lecture* (30)

Usage:

```

int1 int2
where

```

- **int1** *int*: First integer.
- **int2** *int*: Second integer.

26.3 bloc_couronne

Description: Class to create a couronne (2D).

See also: [objet_lecture \(30\)](#)

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.

26.4 bloc_tube

Description: Class to create a tube (3D).

See also: [objet_lecture \(30\)](#)

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.

27 turbulence_paroi_base

Description: Basic class for wall laws for NAVIER STOKES equations.

See also: [objet_u \(31\)](#) [loi_standard_hydr_old \(27.2\)](#) [loi_standard_hydr \(27.1\)](#) [paroi_tble \(27.4\)](#) [negligeable \(27.3\)](#) [utau_imp \(27.8\)](#)

Usage:

27.1 loi_expert_hydr

Description: This keyword is similar to the previous keyword `Loi_standard_hydr` but has several additional options into brackets.

See also: [loi_standard_hydr \(27.1\)](#)

Usage:

loi_expert_hydr obj Lire obj {

```
[ u_star_impose float]
[ methode_calcul_face_keps_impose str into ['toutes_les_faces_accrochees', 'que_les_faces_des-
_elts_dirichlet']]
[ kappa float]
[ Erugu float]
[ A_plus float]
```

}

where

- **u_star_impose** float: The value of the friction velocity (u^*) is not calculated but given by the user.
- **methode_calcul_face_keps_impose** str into ['toutes_les_faces_accrochees', 'que_les_faces_des_elts_dirichlet']: The available options select the algorithm to apply K and Eps boundaries condition (the algorithms differ according to the faces).
toutes_les_faces_accrochees : Default option in 2D (the algorithm is the same than the algorithm used in Loi_standard_hydr)
que_les_faces_des_elts_dirichlet : Default option in 3D (another algorithm where less faces are concerned when applying K-Eps boundary condition).
- **kappa** float: The value can be changed from the default one (0.415)
- **Erugu** float: The value of E can be changed from the default one for a smooth wall (9.11). It is also possible to change the value for one boundary wall only with paroi_rugueuse keyword/
- **A_plus** float: The value can be changed from the default one (26.0)

27.2 loi_standard_hydr

Description: Keyword for the logarithmic wall law for a hydraulic problem. Loi_standard_hydr refers to first cell rank eddy-viscosity defined from continuous analytical functions, whereas Loi_standard_hydr_3couches from functions separately defined for each sub-layer

See also: turbulence_paro_base (27) loi_expert_hydr (27)

Usage:

loi_standard_hydr

27.3 loi_standard_hydr_old

Description: not_set

See also: turbulence_paro_base (27)

Usage:

loi_standard_hydr_old

27.4 negligable

Description: Keyword to suppress the calculation of a law of the wall with a turbulence model. The wall stress is directly calculated with the derivative of the velocity, in the direction perpendicular to the wall ($\tau_{\tan}/\rho = \nu \, dU/dy$).

Warning: This keyword is not available for k-epsilon models. In that case you must choose a wall law.

See also: [turbulence_paro_base \(27\)](#)

Usage:

negligeable

27.5 paroi_tble

Description: Keyword for the Thin Boundary Layer Equation wall-model (a more complete description of the model can be found into this PDF file). The wall shear stress is evaluated thanks to boundary layer equations applied in a one-dimensional fine grid in the near-wall region.

See also: [turbulence_paro_base \(27\)](#)

Usage:

paroi_tble obj Lire obj {

```
[ n int]  
[ facteur float]  
[ modele_visco str]  
[ stats twofloat]  
[ sonde_tble liste_sonde_tble]  
[ restart ]  
[ stationnaire entierfloat]  
[ lambda str]  
[ mu str]  
[ sans_source_boussinesq ]  
[ alpha float]  
[ kappa float]
```

}

where

- **n** *int*: Number of nodes in the TBLE grid (mandatory option).
- **facteur** *float*: Stretching ratio for the TBLE grid (to refine, the TBLE facteur must be greater than 1).
- **modele_visco** *str*: File name containing the description of the eddy viscosity model.
- **stats** *twofloat* (27.5): Statistics of the TBLE velocity and turbulent viscosity profiles. 2 values are required : the starting time and ending time of the statistics computation.
- **sonde_tble** *liste_sonde_tble* (27.6)
- **restart**
- **stationnaire** *entierfloat* (27.7.1)
- **lambda** *str*
- **mu** *str*
- **sans_source_boussinesq**
- **alpha** *float*
- **kappa** *float*

27.6 twofloat

Description: two reals.

See also: [objet_lecture \(30\)](#)

Usage:

a b

where

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

27.7 liste_sonde_tble

Description: not_set

See also: listobj ([29.2](#))

Usage:

n object1 object2

list of *sonde_tble* ([27.7](#))

27.7.1 sonde_tble

Description: not_set

See also: objet_lecture ([30](#))

Usage:

name point

where

- **name** *str*
- **point** *un_point* ([2.7.2](#))

27.8 entierfloat

Description: An integer and a real.

See also: objet_lecture ([30](#))

Usage:

the_int the_float

where

- **the_int** *int*: Integer.
- **the_float** *float*: Real.

27.9 utau_imp

Description: Keyword to impose the friction velocity on the wall with a turbulence model for thermohydraulic problems. There are two possibilities to use this keyword :

1 - we can impose directly the value of the friction velocity u_{star} .

2 - we can also give the friction coefficient and hydraulic diameter. So, TRUST determines the friction velocity by : $u_{star} = U \cdot \sqrt{\lambda_c / 8}$.

See also: turbulence_paro_base ([27](#))

Usage:

utau_imp obj Lire obj {

```

    [ u_tau champ_base]
    [ lambda_c str]
    [ diam_hydr champ_base]
}
where

```

- **u_tau** *champ_base* (14): Field type.
- **lambda_c** *str*: The friction coefficient. It can be function of the spatial coordinates x,y,z, the Reynolds number Re, and the hydraulic diameter.
- **diam_hydr** *champ_base* (14): The hydraulic diameter.

28 turbulence_paroι_scalaire_base

Description: Basic class for wall laws for energy equation.

See also: objet_u (31) loi_standard_hydr_scalaire (28.3) loi_analytique_scalaire (28) paroi_tble_scal (28.5) loi_paroι_nu_impose (28.2) negligeable_scalaire (28.4)

Usage:

28.1 loi_analytique_scalaire

Description: not_set

See also: turbulence_paroι_scalaire_base (28)

Usage:

loi_analytique_scalaire

28.2 loi_expert_scalaire

Description: Keyword similar to keyword Loi_standard_hydr_scalaire but with additional option.

See also: loi_standard_hydr_scalaire (28.3)

Usage:

```

loi_expert_scalaire obj Lire obj {
    [ prdt_sur_kappa float]
    [ calcul_ldp_en_flux_impose int into [0, 1]]
}
where

```

- **prdt_sur_kappa** *float*: This option is to change the default value of 2.12 in the scalable wall function.
- **calcul_ldp_en_flux_impose** *int into [0, 1]*: By default (value set to 0), the law of the wall is not applied for a wall with a Neumann condition. With value set to 1, the law is applied even on a wall with Neumann condition.

28.3 loi_paroι_nu_impose

Description: Keyword to impose Nusselt numbers on the wall for the thermohydraulic problems. To use this option, it is necessary to give in the data file the value of the hydraulic diameter and the expression of the Nusselt number.

See also: turbulence_paroι_scalaire_base (28)

Usage:

loi_paroι_nu_impose obj Lire obj {

nusselt *str*

diam_hydr *champ_base*

}

where

- **nusselt** *str*: The Nusselt number. This expression can be a function of x, y, z, Re (Reynolds number), Pr (Prandtl number).
- **diam_hydr** *champ_base* (14): The hydraulic diameter.

28.4 loi_standard_hydr_scalaire

Description: Keyword for the law of the wall.

See also: turbulence_paroι_scalaire_base (28) loi_expert_scalaire (28.1)

Usage:

loi_standard_hydr_scalaire

28.5 negligeable_scalaire

Description: Keyword to suppress the calculation of a law of the wall with a turbulence model for thermohydraulic problems. The wall stress is directly calculated with the derivative of the velocity, in the direction perpendicular to the wall.

See also: turbulence_paroι_scalaire_base (28)

Usage:

negligeable_scalaire

28.6 paroi_tble_scal

Description: Keyword for the Thin Boundary Layer Equation thermal wall-model.

See also: turbulence_paroι_scalaire_base (28)

Usage:

paroi_tble_scal obj Lire obj {

[**n** *int*]

[**facteur** *float*]

[**modele_visco** *str*]

[**nb_comp** *int*]

[**stats** *fourfloat*]


```

[ sonde_tble liste_sonde_tble]
[ prandtl float]
}

```

where

- **n** *int*: Number of nodes in the TBLE grid (mandatory option).
- **facteur** *float*: Stretching ratio for the TBLE grid (to refine, the TBLE facteur must be greater than 1).
- **modele_visco** *str*: File name containing the description of the eddy viscosity model.
- **nb_comp** *int*: Number of component to solve in the fine grid (1 if 2D simulation (2D not available yet), 2 if 3D simulation).
- **stats** *fourfloat* (28.6): Statistics of the TBLE velocity and turbulent viscosity profiles. 4 values are required : the starting time of velocity averaging, the starting time of the RMS fluctuations, the ending time of the statistics computation and finally the print time period for the statistics.
- **sonde_tble** *liste_sonde_tble* (27.6)
- **prandtl** *float*

28.7 fourfloat

Description: Four reals.

See also: `objet_lecture` (30)

Usage:

a b c d

where

- **a** *float*: First real.
- **b** *float*: Second real.
- **c** *float*: Third real.
- **d** *float*: Fourth real.

29 listobj_impl

Description: `not_set`

See also: `objet_u` (31) `listobj` (29.2)

Usage:

29.1 list_un_pb

Description: pour les groupes

See also: `listobj` (29.2)

Usage:

{ `object1` , `object2` }

list of `un_pb` (29.1) separated with ,

29.2 un_pb

Description: pour les groupes

See also: [objet_lecture \(30\)](#)

Usage:

mot

where

- **mot** *str*: la chaîne

29.3 listobj

Description: List of objects.

See also: [listobj_impl \(29\)](#) [champs_a_post \(3.2.17\)](#) [list_stat_post \(3.2.20\)](#) [listpoints \(3.2.6\)](#) [sondes \(3.2.2\)](#) [listchamp_generique \(6.2\)](#) [list_nom_virgule \(6.1\)](#) [definition_champs \(3.2\)](#) [post_processings \(3.2.28\)](#) [liste_post \(3.4.3\)](#) [liste_post_ok \(3.3.1\)](#) [condlims \(4.3.1\)](#) [sources \(4.4.1\)](#) [vect_nom \(2.97\)](#) [list_nom \(2.82\)](#) [list_bord \(2.46.3\)](#) [list_bloc_mailler \(2.45\)](#) [list_un_pb \(29\)](#) [list_list_nom \(3.7\)](#) [ecrire_fichier_xyz_valeur_param \(4.5\)](#) [pp \(4.15\)](#) [listdeuxmots_sacc \(25.23\)](#) [liste_sonde_tble \(27.6\)](#) [listeqn \(3.9\)](#) [list_info_med \(3.31\)](#) [listsous_zone_valeur \(4.9.11\)](#) [reactions \(7\)](#)

Usage:

30 objet_lecture

Description: Auxiliary class for reading.

See also: [objet_u \(31\)](#) [bloc_lecture \(2.37\)](#) [deuxmots \(4.19\)](#) [format_file \(3.5.3\)](#) [deuxentiers \(26.1\)](#) [floatfloat \(4.20\)](#) [entierfloat \(27.7.1\)](#) [champ_a_post \(3.2.18\)](#) [champs_posts \(3.2.16\)](#) [stat_post_deriv \(3.2.21\)](#) [stats_posts \(3.2.19\)](#) [stats_serie_posts \(3.2.27\)](#) [sonde_base \(3.2.4\)](#) [un_point \(2.7.2\)](#) [sonde \(3.2.3\)](#) [definition_champ \(3.2.1\)](#) [postraitement_base \(3.4.1\)](#) [un_postraitement \(3.3\)](#) [type_un_post \(3.5.1\)](#) [type_postraitement_ft_lata \(3.5.2\)](#) [un_postraitement_spec \(3.5\)](#) [nom_postraitement \(3.4\)](#) [condinit \(4.3\)](#) [condinits \(4.2.9\)](#) [condlimlu \(4.4\)](#) [mailler_base \(2.46\)](#) [bloc_pave \(2.46.2\)](#) [defbord \(2.46.6\)](#) [bord_base \(2.46.4\)](#) [parametre_equation_base \(4.6.2\)](#) [un_pb \(29.1\)](#) [bords_ecrire \(4.6.1\)](#) [ecrire_fichier_xyz_valeur_item \(4.6\)](#) [convection_deriv \(4.9\)](#) [bloc_convection \(4.8\)](#) [diffusion_deriv \(4.2\)](#) [op_implicit \(4.2.8\)](#) [bloc_diffusion \(4.1\)](#) [traitement_particulier_base \(4.22\)](#) [traitement_particulier \(4.21\)](#) [penalisation_l2_ftd_lec \(4.16\)](#) [dt_impr_ustar_mean_only \(4.25\)](#) [modele_turbulence_hyd_deriv \(4.24\)](#) [paroi_ft_disc_deriv \(30\)](#) [bloc_sutherland \(18.4\)](#) [form_a_nb_points \(4.25.3\)](#) [modele_fonction_bas_reynolds_base \(4.25.10\)](#) [fourfloat \(28.6\)](#) [twofloat \(27.5\)](#) [sonde_tble \(27.7\)](#) [remove_elem_bloc \(2.71\)](#) [lecture_bloc_moment_base \(2.6\)](#) [bloc_origine_cotes \(26\)](#) [bloc_couronne \(26.2\)](#) [bloc_tube \(26.3\)](#) [bloc_lecture_poro \(2.55\)](#) [bloc_lec_champ_init_canal_sinal \(14.11\)](#) [fonction_champ_reprise \(14.7\)](#) [bloc_decouper \(2.52\)](#) [troisf \(2.31\)](#) [spec_pdc_base \(25.14\)](#) [format_lata_to_med \(2.41\)](#) [info_med \(3.32\)](#) [methode_transport_deriv \(30.1.1\)](#) [bloc_ef \(4.9.8\)](#) [sous_zone_valeur \(4.9.12\)](#) [bloc_diffusion_standard \(4.2.6\)](#) [reaction \(7.1\)](#)

Usage:

30.1 paroi_ft_disc_deriv

Description: not_set

See also: [objet_lecture \(30\)](#) [symetrie \(30.1\)](#)

Usage:

paroi_ft_disc_deriv

30.1.1 symetrie

Description: Symetrie condition in the case of two-phase flows

See also: `paroi_ft_disc_deriv` ([30](#))

Usage:

symetrie

30.2 methode_transport_deriv

Description: Basic class for method of transport of interface.

See also: `objet_lecture` ([30](#)) `loi_horaire` ([30.2](#))

Usage:

methode_transport_deriv

30.2.1 loi_horaire

Description: `not_set`

See also: `methode_transport_deriv` ([30.1.1](#))

Usage:

loi_horaire nom_loi

where

- **nom_loi** *str*

31 index

Index

/*, 139

#, 158

1D, 120

3D, 120

A_plus, 252

acceleration, 239

alias, 108, 110

alpha, 103, 104, 253

alpha_0, 202

alpha_1, 202

alpha_a, 202

alpha_sous_zone, 104

amont_sous_zone, 104

ampli_bruit, 177

ampli_sin, 177

ascii, 14, 49

avec_certains_bords, 24

avec_certains_bords_pour_extraire_surface, 23

avec_les_bords, 24

beta_co, 193, 194

beta_th, 193, 194

binaire, 18, 41

boite, 250

bord, 16, 119, 241

bords_a_decouper, 18

boundaries, 125

boundary_xmax, 35

boundary_xmin, 35

boundary_ymax, 35

boundary_ymin, 35

boundary_zmax, 35

boundary_zmin, 35

btd, 107

c0, 240

c1_eps, 239, 248, 249

c2_eps, 239, 248, 249

c3_eps, 238, 249

calc_spectre, 120

calcul_ldp_en_flux_impose, 255

canalx, 132

centre_rotation, 239

champ_med, 29

changement_de_base_p1bulle, 172

cl_presson_sommet_faible, 173

cmu, 135

coeff, 241

coefficient_diffusion, 192

coefficients_activites, 150

compo, 145

condition_elements, 23, 24

condition_faces, 24

condition_geometrique, 18

conduction, 70

conservation_Ec, 120

constante_modele_micro_melange, 149

constante_taux_reaction, 150

contre_energie_activation, 150

contre_reaction, 150

controle_residu, 153, 235–238

convection, 100, 107, 108, 110–115, 117, 122, 124, 137, 138

convection_diffusion_chaleur_qc, 84, 85

convection_diffusion_chaleur_turbulent_qc, 88, 89

convection_diffusion_concentration, 72, 73, 79, 80

convection_diffusion_concentration_turbulent, 74, 75, 81, 82

convection_diffusion_temperature, 78–80, 86

convection_diffusion_temperature_turbulent, 81, 82, 87, 90

correction_visco_turb_pour_controle_pas_de_temps, 125, 127, 129–131, 133–135

correction_visco_turb_pour_controle_pas_de_temps-_parametre, 125, 127, 129–131, 133–135

corriger_partition, 199

couronne, 250

Cp, 191

cp, 166, 167, 173, 192–195

crank, 99

critere_absolu, 25

cs, 130

Cv, 191

cw, 128

d, 181, 183

debit, 166, 167

debit_impose, 241

debut_stat, 119

definition_champs, 55, 66

delta, 165

derivee_rotation, 192

dh, 166, 167

diag, 153

diam_hydr, 242–244, 255, 256

diam_hydr_ortho, 243

diffusion, 93, 100, 107, 108, 110–115, 117, 122, 124, 137, 138

diffusion_implicite, 203, 205, 207, 209, 210, 212, 214, 215, 217, 218, 221, 223, 225, 227,

230, 232, 234
 dir , 166, 167
 dir_flow , 177
 dir_wall , 177
 direction , 17, 25–27, 119, 242, 243
 dmax , 132
 domain , 35
 domaine , 16, 18, 23–27, 40, 56, 66, 143, 145, 199
 domaine_final , 17, 25
 domaine_grossier , 18
 domaine_init , 17, 25
 domaines , 41
 domeadt , 239
 dt_impr , 125, 166, 167, 203, 205, 207, 209, 210, 212, 213, 215, 217, 218, 220, 223, 225, 227, 229, 232, 233
 dt_impr_moy_spat , 119
 dt_impr_moy_temp , 119
 dt_impr_nusselt , 196, 197
 dt_impr_ustar , 125, 127, 129–131, 133–135
 dt_impr_ustar_mean_only , 125, 127, 129–131, 133–135
 dt_max , 203, 205, 207, 208, 210, 212, 213, 215, 216, 218, 220, 222, 225, 227, 229, 231, 233
 dt_min , 203, 205, 207, 208, 210, 212, 213, 215, 216, 218, 220, 222, 225, 227, 229, 231, 233
 dt_projection , 116, 122, 123, 137
 dt_sauv , 203, 205, 207, 209, 210, 212, 213, 215, 217, 218, 220, 223, 225, 227, 229, 232, 233
 dt_start , 203, 205, 207, 209, 210, 212, 213, 215, 217, 218, 220, 223, 225, 227, 229, 232, 233
 Ec , 120
 Ec_dans_repere_fixe , 120
 ecrire_decoupage , 38
 ecrire_fichier_xyz_valeur , 93, 100, 107, 108, 110–115, 117, 122, 124, 137, 138
 ecrire_fichier_xyz_valeur_bin , 93, 100, 107, 109–115, 117, 122, 124, 137, 138
 ecrire_frontiere , 41
 ecrire_lata , 38
 emissivite_pour_rayonnement_entre_deux_plaques , 168
 _quasi_infinies , 168
 energie_activation , 150
 enthalpie_reaction , 150
 epaisseur , 23, 25
 eps_min , 125, 127, 129–131, 133–135
 equation_frequence_resolue , 100
 equation_non_resolue , 93, 100, 101, 108–114, 116, 117, 122, 124, 137, 138
 equations_scalaires_passifs , 68, 73, 75, 80, 82, 85, 86, 89, 90
 Erugu , 252
 erugu , 170
 espece , 111, 112
 espece_en_competition_micro_melange , 149
 exposant_beta , 150
 expression , 148
 facon_init , 120
 facsec , 203, 205, 207, 208, 210, 212, 213, 215, 216, 218, 220, 223, 225, 227, 229, 231, 233
 facsec_max , 205, 207, 220, 222, 224, 226, 229, 231
 facteur , 106, 107, 253, 257
 facteurs , 31
 fichier , 56, 66, 132, 198, 250
 fichier_ecriture_K_Eps , 132
 fichier_matrice , 49
 fichier_post , 17
 fichier_secmem , 49
 fichier_solution , 49
 fichier_solveur , 49
 fichier_solveur_non_recree , 153
 fichier_sortie , 29
 file_coord_x , 35
 file_coord_y , 35
 file_coord_z , 35
 fin_stat , 119
 fonction , 45, 131
 fonction_filtre , 36
 fonction_sous_zone , 250
 format , 40, 56, 66
 format_post , 36
 formate , 39
 formulation_a_nb_points , 127, 128, 130–132, 134
 frequence_recalc , 153
 function_coord_x , 35
 function_coord_y , 35
 function_coord_z , 35
 gamma , 191
 genere_fichier_solveur , 49
 ghost_thickness , 35
 groupes , 68
 h , 177, 241
 hexa_old , 25
 impr , 49, 150, 152, 153, 158
 impr_diffusion_implicit , 203, 205, 207, 209, 211, 212, 214, 215, 217, 219, 221, 223, 225, 228, 230, 232, 234
 indice , 193–195
 info , 95
 init_Ec , 120
 initial_value , 178, 184
 interfaces , 56, 66
 intervalle , 250

inverse_condition_element , 23
 joints_non_postraites , 41
 k , 194
 k_min , 125, 127, 129, 130, 132–135
 kappa , 193–195, 252, 253
 kmetis , 199
 lambda , 166, 167, 173, 192–195, 242, 243, 247, 253
 lambda_c , 255
 lambda_max , 247
 lambda_min , 247
 lambda_ortho , 242, 243
 larg_joint , 38
 liste , 45, 250
 liste_cas , 21
 liste_de_postraitements , 55, 69–81, 83–90, 92
 liste_postraitements , 55, 69–81, 83–90, 92
 localisation , 36, 144, 148
 loi_etat , 194
 longueur_boite , 120
 longueur_maille , 127, 128, 130, 131, 133, 134
 longueurs , 31
 main , 39
 masse_molaire , 108, 110, 173
 max_iter_implicit , 220, 222, 224, 227, 229, 231
 methode , 29, 143, 145, 147, 148
 methode_calcul_face_keps_impose , 252
 methode_calcul_pression_initiale , 116, 121, 123, 136
 min_dir_flow , 177
 min_dir_wall , 177
 mode_calcul_convection , 100, 107
 modele_fonc_bas_reynolds , 135
 modele_micro_melange , 149
 modele_turbulence , 107, 109, 112, 114, 123, 136
 modele_visco , 253, 257
 modif_div_face_dirichlet , 173
 moyenne_convergee , 146
 mu , 166, 167, 173, 193–195, 253
 n , 167, 194, 253, 257
 name_of_initial_zones , 14
 name_of_new_zones , 14
 navier_stokes_qc , 84, 85
 navier_stokes_standard , 71–73, 78–80, 86
 navier_stokes_turbulent , 74–76, 81, 82, 87, 90
 navier_stokes_turbulent_qc , 88, 89
 nb_comp , 178, 183, 184, 257
 nb_corrections_max , 235, 236
 nb_it_max , 153, 235–238
 nb_nodes , 35
 nb_parts , 198–200
 nb_parts_geom , 18
 nb_parts_naif , 18
 nb_parts_tot , 38
 nb_pas_dt_max , 203, 205, 207, 208, 210, 212, 213, 215, 217, 218, 220, 223, 225, 227, 229, 232, 233
 nb_points_par_phase , 119
 nb_procs , 21
 nb_test , 49
 nb_tranche , 29
 nb_tranches , 25–27
 nb_var , 131
 new_jacobian , 95
 niter_avg , 205, 207
 niter_max , 204, 207
 niter_max_diffusion_implicit , 99, 203, 205, 207, 209, 211, 212, 214, 215, 217, 219, 221, 223, 225, 228, 230, 232, 234
 niter_min , 204, 207
 no_check_disk_space , 204, 206, 208, 209, 211, 212, 214, 216, 217, 219, 221, 223, 226, 228, 230, 232, 234
 no_conv_subiteration_diffusion_implicit , 203, 206, 208, 209, 211, 212, 214, 216, 217, 219, 221, 223, 226, 228, 230, 232, 234
 no_error_if_not_converged_diffusion_implicit , 203, 206, 208, 209, 211, 212, 214, 216, 217, 219, 221, 223, 225, 228, 230, 232, 234
 no_qdm , 235–238
 nom , 178, 183, 184
 nom_bord , 25
 nom_cl_derriere , 27
 nom_cl_devant , 27
 nom_domaine , 36
 nom_fichier_post , 36
 nom_fichier_solveur , 153
 nom_fichier_sortie , 18
 nom_frontiere , 143
 nom_inconnue , 108, 109
 nom_pb , 36
 nom_source , 139–148
 nombre_de_noeuds , 31
 noms_champs , 36
 non_perio , 25
 normal_value , 183
 nu , 95, 166, 167
 nu_transp , 95
 numero , 145, 148
 numero_op , 141
 numero_source , 141
 nusselt , 256
 nut , 95
 nut_max , 125, 127, 129–131, 133–135
 nut_transp , 95
 old , 104
 omega , 177, 201, 204, 239

omega_relaxation_drho_dt , 195
 optimisation_sous_maillage , 144
 optimized , 152, 158
 option , 145, 239
 Origine , 31
 origine , 23
 p0 , 172
 p1 , 172
 p_imposee_aux_faces , 37
 pa , 172
 par_sous_zone , 17
 parametre_equation , 93, 101, 108–115, 117, 122, 124, 137, 138
 pas_de_solution_initiale , 49
 pb_champ , 146, 147
 pb_name , 39
 penalisation_l2_ftd , 113
 perio_x , 35
 perio_y , 35
 perio_z , 35
 periode , 120
 periode_calc_spectre , 120
 periode_sauvegarde_securite_en_heures , 203, 206, 208, 209, 211, 212, 214, 216, 217, 219, 221, 223, 226, 228, 230, 232, 234
 periodique , 39
 point1 , 23
 point2 , 23
 point3 , 23
 polynomes , 250
 position , 192
 prandtl_turbulent_fonction_nu_t_alpha , 196
 Prandtl , 191
 prandtl , 257
 prandtl_eps , 125, 127, 129, 130, 132–135
 prandtl_k , 125, 127, 129, 130, 132–135
 prdt , 196
 prdt_sur_kappa , 255
 precision_impr , 203, 205, 207, 209, 211, 212, 214, 216, 217, 219, 221, 223, 225, 228, 230, 232, 234
 precondition , 151, 158
 precondition0 , 202
 precondition1 , 202
 precondition_nul , 152, 158
 preconda , 202
 preconditionnement_diag , 99
 pression , 194
 probleme , 23, 24, 178, 184
 produits , 150
 projection_initiale , 116, 121, 123, 136
 projection_normale_bord , 25
 pulsation_w , 119
 quiet , 150, 152, 153, 158
 reactifs , 150
 reactions , 149
 rectangle , 250
 relax_pression , 236
 reorder , 39
 reprise , 55, 69–80, 82–89, 91, 92, 119
 reprise_correlation , 167, 168
 resolution_explicite , 100
 restart , 253
 restriction , 250
 resume_last_time , 55, 69–79, 81–88, 90–92
 rho , 166, 167, 192–195
 rho_constant_pour_debug , 191
 rotation , 192
 sans_passer_par_le2D , 25
 sans_solveur_masse , 141
 sans_source_boussinesq , 253
 sauvegarde , 55, 69–81, 83–90, 92
 sauvegarde_simple , 55, 69–80, 82–89, 91, 92
 Sc , 191
 scturb , 197
 segment , 250
 seuil , 152, 153, 158, 205, 207
 seuil_convergence_implicit , 99, 235–237
 seuil_convergence_solveur , 100, 235–237
 seuil_diffusion_implicit , 99, 203, 205, 207, 209, 211, 212, 214, 215, 217, 219, 221, 223, 225, 228, 230, 232, 234
 seuil_divU , 117, 122, 123, 137
 seuil_generation_solveur , 235–237
 seuil_statio , 203, 205, 207, 209, 210, 212, 214, 215, 217, 218, 221, 223, 225, 227, 230, 232, 233
 seuil_statio_relatif_deconseille , 203, 205, 207, 209, 210, 212, 214, 215, 217, 218, 221, 223, 225, 227, 230, 232, 234
 seuil_test_preliminaire_solveur , 235–237
 seuil_verification , 49
 seuil_verification_solveur , 235–237
 solveur , 49, 100, 220, 222, 224, 227, 229, 231, 235–238
 solveur0 , 151
 solveur1 , 151
 solveur_bar , 116, 121, 123, 136
 solveur_pression , 116, 121, 123, 136
 sonde_tble , 253, 257
 source , 139–148
 source_reference , 139–148
 sources , 93, 100, 107, 108, 110–115, 117, 122, 124, 137–149
 sources_reference , 139–148
 sous_zone , 23, 178, 184, 242–244
 sous_zones , 200
 splitting , 35

standard , 95
 stationnaire , 253
 statistiques , 56, 66
 statistiques_en_serie , 56, 66
 stats , 253, 257
 surface , 167
 surfacique , 40
 sutherland , 194
 symx , 31
 symy , 31
 symz , 31
 t0 , 240
 t_deb , 141–143, 146
 t_fin , 141–143, 146
 tanh , 31
 tanh_dilatation , 31
 tanh_taille_premiere_maille , 32
 tcpumax , 203, 205, 207, 208, 210, 212, 213, 215, 216, 218, 220, 222, 225, 227, 229, 231, 233
 tdivu , 104
 temps_debut_prise_en_compte_drho_dt , 194
 test , 104
 tinf , 166, 167
 tinit , 203, 205, 207, 208, 210, 211, 213, 215, 216, 218, 220, 222, 225, 227, 229, 231, 233
 tmax , 203, 205, 207, 208, 210, 212, 213, 215, 216, 218, 220, 222, 225, 227, 229, 231, 233
 traitement_coins , 37
 traitement_particulier , 117, 122, 124, 137
 traitement_pth , 194
 traitement_rho_gravite , 194
 tranches , 200
 transport_k_epsilon , 135
 triangle , 23
 trois_tetra , 25
 tsup , 166, 167
 tube , 250
 turbulence_paroie , 125, 127, 129–131, 133–135, 196, 197
 tuyauz , 132
 type , 145
 u , 181, 183
 u_star_impose , 252
 u_tau , 255
 ubar_umprim_cible , 247
 ucent , 177
 union , 250
 use_weights , 199
 val_Ec , 120
 verif_boussinesq , 240
 verif_dparoi , 132
 via_extraire_surface , 23
 vingt_tetra , 25
 vitesse , 192, 239
 volume , 166
 volumes_etendus , 104
 volumes_non_etendus , 104
 volumique , 40
 with_nu , 138
 xinf , 167
 xsup , 167
 zmax , 29
 zmin , 29
 acceleration, 239
 amount, 101
 amount_old, 101
 analyse_angle, 14
 associate, 14
 axi, 15
 bidim_axi, 15
 bloc_convection, 101
 bloc_couronne, 250
 bloc_decouper, 38
 bloc_diffusion, 94
 bloc_diffusion_standard, 95
 bloc_ef, 103
 bloc_lec_champ_init_canal_sinal, 176
 bloc_lecture, 28
 bloc_lecture_poro, 39
 bloc_origine_cotes, 250
 bloc_pave, 31
 bloc_sutherland, 195
 bloc_tube, 251
 bord, 32
 bord_base, 32
 bords_ecrire, 98
 boundary_field_inward, 182
 boundary_field_uniform_keps_from_ud, 183
 boussinesq_concentration, 239
 boussinesq_temperature, 240
 btd, 106
 calcul, 15
 calculer_moments, 15
 canal, 119
 canal_perio, 240
 centre, 102
 centre4, 102
 centre_de_gravite, 16
 centre_old, 102
 ch_front_input, 183
 ch_front_input_uniforme, 184
 champ_a_post, 61
 champ_base, 173
 champ_don_base, 173

champ_don_lu, 174
 champ_fonc_fonction, 174
 champ_fonc_fonction_txyz, 174
 champ_fonc_med, 175
 champ_fonc_reprise, 175
 champ_fonc_t, 176
 champ_fonc_tabule, 176
 champ_fonc_txyz, 180
 champ_fonc_xyz, 180
 champ_front_base, 182
 champ_front_bruite, 184
 champ_front_calc, 184
 champ_front_contact_vef, 185
 champ_front_debit, 185
 champ_front_fonc_pois_ipsn, 185
 champ_front_fonc_pois_tube, 186
 champ_front_fonc_txyz, 186
 champ_front_fonc_xyz, 186
 champ_front_fonction, 186
 champ_front_lu, 187
 champ_front_normal_vef, 187
 champ_front_pression_from_u, 187
 champ_front_recyclage, 188
 champ_front_tabule, 189
 champ_front_tangentiel_vef, 190
 champ_front_uniforme, 190
 champ_generique_base, 139
 champ_init_canal_sinal, 176
 champ_input_base, 177
 champ_input_p0, 178
 champ_ostwald, 178
 champ_post_de_champs_post, 139
 champ_post_extraction, 143
 champ_post_interpolation, 144
 champ_post_morceau_equation, 145
 champ_post_operateur_base, 140
 champ_post_operateur_divergence, 142
 champ_post_operateur_eqn, 140
 champ_post_operateur_gradient, 144
 champ_post_reduction_0d, 146
 champ_post_refchamp, 147
 champ_post_statistiques_base, 141
 champ_post_tparoi_vef, 147
 champ_post_transformation, 148
 champ_som_lu_vdf, 178
 champ_som_lu_vef, 179
 champ_tabule_temps, 179
 champ_uniforme_morceaux, 179
 champ_uniforme_morceaux_tabule_temps, 180
 Champ_front_fonc_txyz, 13
 champs_a_post, 60
 champs_posts, 60
 chimie, 149
 chmoy_faceperio, 120

Cholesky, 154–156
 cholesky, 150
 circle, 59
 circle_3, 60
 class_generic, 150
 combinaison, 130
 Concentration, 61, 64
 condinit, 97
 condinits, 96
 condlim_base, 159
 condlimlu, 97
 condlims, 97
 conduction, 93
 constituant, 192
 convection_deriv, 101
 convection_diffusion_chaleur_qc, 100
 convection_diffusion_chaleur_turbulent_qc, 107
 convection_diffusion_concentration, 108
 convection_diffusion_concentration_turbulent, 109
 convection_diffusion_fraction_massique_qc, 110
 convection_diffusion_fraction_massique_turbulent_qc, 111
 convection_diffusion_temperature, 112
 convection_diffusion_temperature_turbulent, 114
 coriolis, 241
 corps_postraitements, 55
 Correlation, 61
 correlation, 63, 141
 corriger_frontiere_periodique, 16
 create_domain_from_sous_zone, 17

 darcy, 241
 debog, 17
 decoupebord_pour_rayonnement, 18
 decouper_bord_coincident, 18
 defbord, 32
 defbord_2, 32
 defbord_3, 33
 definition_champ, 56
 definition_champs, 56
 deuxentiers, 250
 deuxmots, 117
 di_l2, 102
 diffusion_deriv, 94
 dilate, 19
 dimension, 19
 dirac, 241
 dirichlet, 159
 discretisation_base, 171
 discretiser_domaine, 19
 discretize, 19
 distance_parois, 20
 domain, 34
 domaine, 173

dt_calc, 150
 dt_fixe, 150
 dt_impr_ustar_mean_only, 125
 dt_min, 151
 dt_start, 151
 Dt_post, 61

 ec, 119
 ecart_type, 63, 142
 Ecart_type, 61, 64
 ecrire, 53
 ecrire_champ_med, 20
 ecrire_fichier_bin, 53
 ecrire_fichier_formatte, 20
 ecrire_fichier_xyz_valeur_item, 98
 ecrire_fichier_xyz_valeur_param, 98
 ecrire_med, 54
 ecriturelecturespecial, 21
 ef, 102, 172
 ef_stab, 103
 end, 27
 entierfloat, 254
 entree_temperature_imposee_h, 159
 epsilon, 34
 eqn_base, 115
 espece, 173
 execute_parallel, 21
 export, 21
 extract_2d_from_3d, 22
 extract_2daxi_from_3d, 22
 extraire_domaine, 22
 extraire_plan, 23
 extraire_surface, 23
 extrudebord, 24
 extrudeparoi, 25
 extruder, 25
 extruder_en20, 26
 extruder_en3, 26

 fichier_decoupage, 198
 field_uniform_keps_from_ud, 181
 floatfloat, 118
 fluide_incompressible, 192
 fluide_ostwald, 193
 fluide_quasi_compressible, 194
 fonction_champ_reprise, 175
 forchheimer, 242
 form_a_nb_points, 127
 format_file, 67
 format_lata_to_med, 29
 fourfloat, 257
 frontiere_ouverte, 159
 frontiere_ouverte_concentration_imposee, 160
 frontiere_ouverte_fraction_massique_imposee, 160
 frontiere_ouverte_gradient_pression_impose, 160
 frontiere_ouverte_gradient_pression_impose_vef, 160
 frontiere_ouverte_gradient_pression_impose_vefprep1b, 161
 frontiere_ouverte_gradient_pression_libre_vef, 161
 frontiere_ouverte_gradient_pression_libre_vefprep1b, 161
 frontiere_ouverte_k_eps_impose, 161
 frontiere_ouverte_pression_imposee, 162
 frontiere_ouverte_pression_imposee_orlansky, 162
 frontiere_ouverte_pression_moyenne_imposee, 162
 frontiere_ouverte_rho_u_impose, 162
 frontiere_ouverte_temperature_imposee, 163
 frontiere_ouverte_vitesse_imposee, 163
 frontiere_ouverte_vitesse_imposee_sortie, 163

 gaz_parfait, 191
 gaz_reel_rhot, 190
 GCP, 154, 157
 gcp, 157
 gcp_ns, 151
 gen, 152
 generic, 105
 gmres, 152
 Gradient, 154

 IBICGSTAB, 154
 implicite, 234
 imprimer_flux, 27
 imprimer_flux_sum, 28
 info_med, 91
 init_par_partie, 181
 integrer_champ_med, 28
 Interface, 155
 internes, 33
 interpreter, 13

 k_epsilon, 134
 kquick, 105

 lata_to_med, 29
 lata_to_other, 29
 leap_frog, 209
 lecture_bloc_moment_base, 15
 lire_ideas, 30
 lire_tgrid, 42
 list_bloc_mailler, 30
 list_bord, 32
 list_info_med, 91
 list_list_nom, 68
 list_nom, 48
 list_nom_virgule, 139
 list_stat_post, 62

list_un_pb, 257
 listchamp_generique, 140
 listdeuxmots_sacc, 247
 liste_post, 66
 liste_post_ok, 65
 liste_sonde_tble, 254
 listeqn, 69
 listobj, 258
 listobj_impl, 257
 listpoints, 57
 listsous_zone_valeur, 104
 local, 156
 loi_analytique_scalaire, 255
 loi_etat_base, 190
 loi_expert_hydr, 251
 loi_expert_scalaire, 255
 loi_horaire, 191, 259
 loi_paroι_nu_impose, 255
 loi_standard_hydr, 252
 loi_standard_hydr_old, 252
 loi_standard_hydr_scalaire, 256
 longitudinale, 244
 longueur_melange, 132

 mailler, 30
 mailler_base, 30
 maillerparalle, 34
 melange_gaz_parfait, 190
 methode_transport_deriv, 259
 metis, 199
 milieu_base, 192
 mod_turb_hyd_ss_maille, 126
 modele_fonction_bas_reynolds_base, 135
 modele_turbulence_hyd_deriv, 124
 modele_turbulence_scal_base, 196
 modif_bord_to_raccord, 35
 mor_eqn, 93
 Moyenne, 61, 64
 moyenne, 63, 145
 moyenne_volumique, 36
 muscl, 105
 muscl3, 103
 muscl_new, 106
 muscl_old, 105

 N, 155
 navier_stokes_qc, 116
 navier_stokes_standard, 121
 navier_stokes_turbulent, 122
 navier_stokes_turbulent_qc, 136
 negligeable, 94, 106, 252
 negligeable_scalaire, 256
 neumann, 163
 nom, 197

nom_anonyme, 197
 nom_postraitement, 65
 NUL, 125
 NULL, 156
 numero_elem_sur_maitre, 58

 objet_lecture, 258
 op_implicite, 96
 optimal, 153
 option, 96
 option_vdf, 37
 orientefacesbord, 37
 orienter_simplexes, 43

 p1b, 94
 p1ncp1b, 94
 parametre_diffusion_implicite, 99
 parametre_equation_base, 98
 parametre_implicite, 99
 Paroi, 159
 paroi_adiabatique, 164
 paroi_contact, 164
 paroi_contact_fictif, 165
 paroi_couple, 165
 paroi_decalee_robin, 165
 paroi_defilante, 165
 paroi_echange_contact_correlation_vdf, 166
 paroi_echange_contact_correlation_vef, 167
 paroi_echange_contact_vdf, 168
 paroi_echange_externe_impose, 168
 paroi_echange_externe_impose_h, 168
 paroi_echange_global_impose, 169
 paroi_fixe, 169
 paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses-
 _sommets, 169
 paroi_flux_impose, 169
 paroi_ft_disc_deriv, 258
 paroi_knudsen_non_negligeable, 169
 paroi_rugueuse, 170
 paroi_tble, 253
 paroi_tble_scal, 256
 paroi_temperature_imposee, 170
 partition, 37, 199
 partitionneur_deriv, 198
 pave, 30
 pb_avec_passif, 68
 Pb_base, 54
 pb_conduction, 69
 pb_gen_base, 54
 pb_hydraulique, 70
 pb_hydraulique_concentration, 71
 pb_hydraulique_concentration_scalaires_passifs,
 72
 pb_hydraulique_concentration_turbulent, 73

pb_hydraulique_concentration_turbulent_scalaires_passifs, 74
 pb_hydraulique_turbulent, 76
 pb_post, 76
 pb_thermohydraulique, 77
 pb_thermohydraulique_concentration, 78
 pb_thermohydraulique_concentration_scalaires_passifs, 79
 pb_thermohydraulique_concentration_turbulent, 81
 pb_thermohydraulique_concentration_turbulent_scalaires_passifs, 82
 pb_thermohydraulique_qc, 83
 pb_thermohydraulique_qc_fraction_massique, 84
 pb_thermohydraulique_scalaires_passifs, 85
 pb_thermohydraulique_turbulent, 86
 pb_thermohydraulique_turbulent_qc, 87
 pb_thermohydraulique_turbulent_qc_fraction_massique, 88
 pb_thermohydraulique_turbulent_scalaires_passifs, 90
 pbc_med, 91
 penalisation_l2_ftd_lec, 113
 periodique, 170
 perte_charge_anisotrope, 242
 perte_charge_circulaire, 242
 perte_charge_directionnelle, 243
 perte_charge_isotrope, 243
 perte_charge_reguliere, 244
 perte_charge_singuliere, 245
 Petsc, 153, 156
 petsc, 153
 pilote_icoco, 39
 piso, 235
 plan, 59
 point, 58
 points, 57
 porosites, 39
 porosites_champ, 40
 position_like, 58
 post_processing, 65
 post_processings, 64
 postraitement_base, 65
 postraiter_domaine, 40
 pp, 113
 prandtl, 196
 precisiongeom, 41
 Precond, 153, 156
 precondition_base, 201
 precondition_local, 201
 precondition_solver, 201
 predefini, 146
 Pression, 61, 64
 Print, 155
 problem_read_generic, 91
 probleme_couple, 67
 puissance_thermique, 245
 quick, 106
 raccord, 33
 raffiner_anisotrope, 41
 raffiner_isotrope, 41
 Raffiner_isotrope_parallele, 14
 reaction, 149
 reactions, 149
 read, 41
 read_file, 42
 read_file_binary, 42
 read_med, 43
 read_unsupported_ascii_file_from_icem, 43
 redresser_hexaedres_vdf, 44
 regroupebord, 44
 remove_elem, 44
 remove_elem_bloc, 45
 remove_invalid_internal_boundaries, 45
 reordonner, 46
 reordonner_faces_periodiques, 45
 reorienter_tetraedres, 46
 reorienter_triangles, 46
 rotation, 47
 runge_kutta_ordre_3, 211
 runge_kutta_ordre_4_d3p, 212
 runge_kutta_rationnel_ordre_2, 214
 scatter, 47
 scattermed, 47
 Sch_CN_EX_iteratif, 204
 Sch_CN_iteratif, 206
 schema_adams_bashforth_order_2, 216
 schema_adams_bashforth_order_3, 217
 schema_adams_moulton_order_2, 219
 schema_adams_moulton_order_3, 221
 schema_backward_differentiation_order_2, 223
 schema_backward_differentiation_order_3, 226
 schema_implicite_base, 230
 schema_predictor_corrector, 232
 schema_temps_base, 202
 scheme_euler_explicit, 208
 scheme_euler_implicit, 228
 schmidt, 197
 segment, 59
 segmentpoints, 58
 simple, 236
 simplifier, 237
 solide, 195
 solve, 47
 Solver, 153, 154, 157

Solveur, 153, 156
 solveur_implicite_base, 234
 solveur_lineaire_std, 238
 solveur_sys_base, 158
 Solveur_pression, 153, 156
 sonde, 56
 sonde_base, 57
 sonde_tble, 254
 sondes, 56
 sortie_libre_temperature_imposee_h, 171
 source_base, 238
 source_constituant, 246
 source_generique, 246
 source_qdm, 246
 source_qdm_lambdaup, 246
 source_robin, 247
 source_robin_scalaire, 247
 source_th_tdivu, 248
 source_transport_k_eps, 248
 source_transport_k_eps_aniso_concen, 248
 source_transport_k_eps_aniso_therm_concen, 249
 Source_Transport_K_Eps_anisotherme, 238
 sources, 97
 sous_maille, 133
 sous_maille_smago, 129
 sous_maille_wale, 128
 sous_zone, 249
 sous_zone_valeur, 104
 sous_zones, 199
 Spai, 156
 spec_pdc_r_base, 244
 SSOR, 156, 157
 ssor, 201
 ssor_bloc, 201
 stab, 94
 standard, 95
 stat_post_deriv, 62
 Statistiques, 61, 64
 Statistiques_en_serie, 64
 stats_posts, 61
 stats_serie_posts, 64
 supg, 106
 supprime_bord, 48
 symetrie, 171, 259
 system, 48

 t_deb, 62
 t_fin, 63
 tayl_green, 181
 Temperature, 61, 64
 temperature, 118
 temperature_imposee_paroi, 171
 test_solveur, 48
 testeur, 49
 testeur_medcoupling, 49
 tetraedriser, 50
 tetraedriser_homogene, 50
 tetraedriser_homogene_compact, 50
 tetraedriser_homogene_fin, 50
 tetraedriser_par_prisme, 51
 thi, 120
 traitement_particulier, 118
 traitement_particulier_base, 118
 tranche, 200
 transformer, 51
 transport_k_epsilon, 137
 transversale, 244
 trianguler, 51
 trianguler_fin, 52
 trianguler_h, 52
 troisf, 26
 turbulence_paroi_base, 251
 turbulence_paroi_scalaire_base, 255
 twofloat, 253
 type, 61, 64, 155, 156
 type_postraitement_ft_lata, 67
 type_un_post, 66

 un_pb, 257
 un_point, 16
 un_postraitement, 64
 un_postraitement_spec, 66
 uniform_field, 182
 utau_imp, 254

 valeur_totale_sur_volume, 182
 vdf, 172
 vect_nom, 52
 vef, 172
 vefprep1b, 172
 verifier_qualite_raffinements, 52
 verifier_simplexes, 53
 verifiercoin, 53
 Vitesse, 61, 64
 volume, 59

 xyz, 13