

# **TrioCFD Reference Manual V1.7.8**

**Support team: [trust@cea.fr](mailto:trust@cea.fr)**

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

December 19, 2018

# Contents

<b>1</b>	<b>Syntax to define a mathematical function</b>	<b>15</b>
<b>2</b>	<b>Existing &amp; predefined fields names</b>	<b>16</b>
<b>3</b>	<b>interprete</b>	<b>17</b>
3.1	Raffiner_isotrope_parallele . . . . .	18
3.2	read_med . . . . .	18
3.3	lire_medfile . . . . .	19
3.4	analyse_angle . . . . .	19
3.5	associer . . . . .	20
3.6	associer_algo . . . . .	20
3.7	associer_pbm_g_pbf . . . . .	20
3.8	associer_pbm_g_pbglobal . . . . .	20
3.9	axi . . . . .	21
3.10	bidim_axi . . . . .	21
3.11	calculer_moments . . . . .	21
3.12	lecture_bloc_moment_base . . . . .	21
	3.12.1 calcul . . . . .	22
	3.12.2 centre_de_gravite . . . . .	22
	3.12.3 un_point . . . . .	22
3.13	corriger_frontiere_periodique . . . . .	22
3.14	create_domain_from_sous_zone . . . . .	23
3.15	debog . . . . .	23
3.16	{ . . . . .	24
3.17	decoupebord_pour_rayonnement . . . . .	24
3.18	decouper_bord_coincident . . . . .	25
3.19	dilate . . . . .	25
3.20	dimension . . . . .	25
3.21	disable_TU . . . . .	25
3.22	discretiser_domaine . . . . .	26
3.23	discretize . . . . .	26
3.24	distance_paro . . . . .	26
3.25	ecrire_champ_med . . . . .	27
3.26	ecrire_fichier_formatte . . . . .	27
3.27	ecriturelecturespecial . . . . .	27
3.28	execute_parallel . . . . .	27
3.29	export . . . . .	28
3.30	extract_2d_from_3d . . . . .	28
3.31	extract_2daxi_from_3d . . . . .	28
3.32	extraire_domaine . . . . .	29
3.33	extraire_plan . . . . .	29
3.34	extraire_surface . . . . .	30
3.35	extrudebord . . . . .	30
3.36	extrudeparoi . . . . .	31
3.37	extruder . . . . .	32
3.38	troisf . . . . .	32
3.39	extruder_en20 . . . . .	32
3.40	extruder_en3 . . . . .	33
3.41	end . . . . .	33
3.42	} . . . . .	34
3.43	imposer_vit_bords_ale . . . . .	34
3.44	bloc_lecture . . . . .	34

3.45	imprimer_flux	34
3.46	imprimer_flux_sum	35
3.47	integrer_champ_med	35
3.48	interprete_geometrique_base	35
3.49	lata_to_med	36
3.50	format_lata_to_med	36
3.51	lata_to_other	36
3.52	lire_ideas	36
3.53	mailler	37
3.54	list_bloc_mailler	37
3.54.1	mailler_base	37
3.54.2	pave	37
3.54.3	bloc_pave	38
3.54.4	list_bord	38
3.54.5	bord_base	39
3.54.6	bord	39
3.54.7	defbord	39
3.54.8	defbord_2	39
3.54.9	defbord_3	39
3.54.10	raccord	40
3.54.11	internes	40
3.54.12	epsilon	41
3.54.13	domain	41
3.55	maillerparallel	41
3.56	modif_bord_to_raccord	42
3.57	moyenne_volumique	42
3.58	nettoiepasnoeuds	44
3.59	option_vdf	44
3.60	orientefacesbord	44
3.61	partition	44
3.62	bloc_decouper	45
3.63	pilote_icoco	46
3.64	porosites	46
3.65	bloc_lecture_poro	47
3.66	porosites_champ	47
3.67	postraiter_domaine	47
3.68	precisiongeom	48
3.69	raffiner_anisotrope	48
3.70	raffiner_isotrope	49
3.71	read	50
3.72	read_file	50
3.73	read_file_binary	50
3.74	lire_tgrid	51
3.75	read_unsupported_ascii_file_from_icem	51
3.76	orienter_simplexes	51
3.77	redresser_hexaedres_vdf	51
3.78	refine_mesh	52
3.79	regroupebord	52
3.80	remove_elem	52
3.81	remove_elem_bloc	53
3.82	remove_invalid_internal_boundaries	53
3.83	reordonner_faces_periodiques	53
3.84	reorienter_tetraedres	54
3.85	reorienter_triangles	54

3.86	reordonner	54
3.87	rotation	54
3.88	scatter	55
3.89	scatterformatte	55
3.90	scattermed	55
3.91	solve	56
3.92	supprime_bord	56
3.93	list_nom	56
3.94	system	56
3.95	test_solveur	57
3.96	testeur	57
3.97	testeur_medcoupling	57
3.98	tetraedriser	58
3.99	tetraedriser_homogene	58
3.100	tetraedriser_homogene_compact	59
3.101	tetraedriser_homogene_fin	59
3.102	tetraedriser_par_prisme	60
3.103	transformer	61
3.104	triangler	61
3.105	triangler_fin	61
3.106	triangler_h	62
3.107	verifier_qualite_raffinements	62
3.108	vect_nom	63
3.109	verifier_simplexes	63
3.110	verifiercoin	63
3.111	verifiercoin_bloc	63
3.112	ecrire	64
3.113	ecrire_fichier_bin	64
3.114	ecrire_med	64
3.115	ecrire_medfile	65
<b>4</b>	<b>pb_gen_base</b>	<b>65</b>
4.1	Pb_base	65
4.2	corps_postraitement	66
4.2.1	definition_champs	67
4.2.2	definition_champ	67
4.2.3	sondes	67
4.2.4	sonde	67
4.2.5	sonde_base	68
4.2.6	points	68
4.2.7	listpoints	68
4.2.8	point	68
4.2.9	segmentpoints	69
4.2.10	numero_elem_sur_maitre	69
4.2.11	position_like	69
4.2.12	segment	69
4.2.13	plan	70
4.2.14	volume	70
4.2.15	circle	70
4.2.16	circle_3	71
4.2.17	champs_posts	71
4.2.18	champs_a_post	71
4.2.19	champ_a_post	71
4.2.20	stats_posts	72

4.2.21	list_stat_post	73
4.2.22	stat_post_deriv	73
4.2.23	t_deb	73
4.2.24	t_fin	73
4.2.25	moyenne	73
4.2.26	ecart_type	74
4.2.27	correlation	74
4.2.28	stats_serie_posts	74
4.3	post_processings	75
4.3.1	un_postraitement	75
4.4	liste_post_ok	75
4.4.1	nom_postraitement	76
4.4.2	postraitement_base	76
4.4.3	post_processing	76
4.4.4	postraitement_ft_lata	77
4.5	liste_post	77
4.5.1	un_postraitement_spec	77
4.5.2	type_un_post	77
4.5.3	type_postraitement_ft_lata	78
4.6	format_file	78
4.7	probleme_couple	78
4.8	list_list_nom	79
4.9	modele_rayo_semi_transp	79
4.10	eq_rayo_semi_transp	80
4.10.1	condlims	80
4.10.2	condlimlu	80
4.11	pb_avec_passif	81
4.12	listeqn	82
4.13	pb_conduction	82
4.14	pb_conduction_milieu_variable	83
4.15	pb_couple_rayo_semi_transp	84
4.16	pb_hydraulique	84
4.17	pb_hydraulique_concentration	85
4.18	pb_hydraulique_concentration_scalaires_passifs	86
4.19	pb_hydraulique_concentration_turbulent	87
4.20	pb_hydraulique_concentration_turbulent_scalaires_passifs	88
4.21	pb_hydraulique_turbulent	89
4.22	pb_mg	90
4.23	pb_phase_field	91
4.24	pb_post	92
4.25	pb_thermohydraulique	93
4.26	pb_thermohydraulique_concentration	94
4.27	pb_thermohydraulique_concentration_scalaires_passifs	95
4.28	pb_thermohydraulique_concentration_turbulent	96
4.29	pb_thermohydraulique_concentration_turbulent_scalaires_passifs	97
4.30	pb_thermohydraulique_qc	98
4.31	pb_thermohydraulique_qc_fraction_massique	99
4.32	pb_thermohydraulique_scalaires_passifs	100
4.33	pb_thermohydraulique_turbulent	101
4.34	pb_thermohydraulique_turbulent_qc	102
4.35	pb_thermohydraulique_turbulent_qc_fraction_massique	104
4.36	pb_thermohydraulique_turbulent_scalaires_passifs	105
4.37	pb_med	106
4.38	list_info_med	106

4.38.1	info_med	106
4.39	problem_read_generic	107
4.40	pb_couple_rayonnement	107
4.41	probleme_ft_disc_gen	108
<b>5</b>	<b>mor_eqn</b>	<b>109</b>
5.1	conduction	109
5.2	bloc_diffusion	110
5.2.1	diffusion_deriv	110
5.2.2	negligeable	110
5.2.3	p1b	111
5.2.4	p1ncp1b	111
5.2.5	stab	111
5.2.6	standard	112
5.2.7	bloc_diffusion_standard	112
5.2.8	option	112
5.2.9	op_implicite	113
5.3	condinits	113
5.3.1	condinit	113
5.4	sources	113
5.5	ecrire_fichier_xyz_valeur_param	114
5.5.1	ecrire_fichier_xyz_valeur_item	114
5.5.2	bords_ecrire	114
5.6	parametre_equation_base	114
5.6.1	parametre_diffusion_implicite	115
5.6.2	parametre_implicite	115
5.7	conduction_milieu_variable	116
5.8	bloc_convection	117
5.8.1	convection_deriv	117
5.8.2	amont	117
5.8.3	amont_old	117
5.8.4	centre	117
5.8.5	centre4	118
5.8.6	centre_old	118
5.8.7	di_l2	118
5.8.8	ef	118
5.8.9	bloc_ef	119
5.8.10	muscl3	119
5.8.11	ef_stab	119
5.8.12	listsous_zone_valeur	120
5.8.13	sous_zone_valeur	120
5.8.14	generic	120
5.8.15	kquick	121
5.8.16	muscl	121
5.8.17	muscl_old	121
5.8.18	muscl_new	121
5.8.19	negligeable	122
5.8.20	quick	122
5.8.21	btd	122
5.8.22	supg	122
5.8.23	ale	123
5.9	convection_diffusion_chaleur_qc	123
5.10	convection_diffusion_chaleur_turbulent_qc	124
5.11	convection_diffusion_concentration	125

5.12	convection_diffusion_concentration_ft_disc	126
5.13	convection_diffusion_concentration_turbulent	128
5.14	convection_diffusion_fraction_massique_qc	129
5.15	convection_diffusion_fraction_massique_turbulent_qc	130
5.16	convection_diffusion_phase_field	131
5.17	convection_diffusion_temperature	132
5.18	pp	133
5.18.1	penalisation_l2_ftd_lec	133
5.19	convection_diffusion_temperature_ft_disc	134
5.20	objet_lecture_maintien_temperature	135
5.21	convection_diffusion_temperature_turbulent	135
5.22	eqn_base	136
5.23	navier_stokes_ft_disc	138
5.24	penalisation_forage	140
5.25	modele_turbulence_hyd_deriv	141
5.25.1	dt_impr_ustar_mean_only	141
5.25.2	NUL	142
5.25.3	mod_turb_hyd_ss_maille	142
5.25.4	form_a_nb_points	144
5.25.5	sous_maille_wale	144
5.25.6	sous_maille_smago	145
5.25.7	combinaison	146
5.25.8	longueur_melange	147
5.25.9	sous_maille	149
5.25.10	sous_maille_selectif_mod	150
5.25.11	deuxentiers	151
5.25.12	floatentier	151
5.25.13	sous_maille_selectif	151
5.25.14	sous_maille_1elt	153
5.25.15	sous_maille_1elt_selectif_mod	154
5.25.16	sous_maille_axi	155
5.25.17	sous_maille_smago_filtre	156
5.25.18	sous_maille_smago_dyn	157
5.25.19	mod_turb_hyd_rans	158
5.25.20	k_epsilon	159
5.25.21	modele_fonction_bas_reynolds_base	160
5.25.22	Lam_Bremhorst	160
5.25.23	standard_KEps	160
5.25.24	EASM_Baglietto	161
5.25.25	Launder_Sharma	161
5.25.26	Jones_Launder	161
5.26	deuxmots	162
5.27	floatfloat	162
5.28	traitement_particulier	162
5.28.1	traitement_particulier_base	162
5.28.2	temperature	162
5.28.3	canal	163
5.28.4	ec	163
5.28.5	thi	164
5.28.6	thi_thermo	165
5.28.7	chmoy_faceperio	165
5.28.8	profils_thermo	166
5.28.9	brech	166
5.28.10	ceg	166

5.28.11	ceg_areva	167
5.28.12	ceg_cea_jaea	167
5.29	navier_stokes_phase_field	167
5.30	navier_stokes_qc	169
5.31	navier_stokes_standard	171
5.32	navier_stokes_turbulent	173
5.33	navier_stokes_turbulent_qc	175
5.34	transport_interfaces_ft_disc	176
5.35	methode_transport_deriv	180
5.35.1	loi_horaire	180
5.35.2	vitesse_imposee	180
5.35.3	vitesse_interpolee	181
5.36	bloc_lecture_remaillage	181
5.37	parcours_interface	182
5.38	interpolation_champ_face_deriv	183
5.38.1	base	183
5.38.2	lineaire	183
5.39	transport_k_epsilon	183
5.40	transport_marqueur_ft	184
5.41	injection_marqueur	186
<b>6</b>	<b>algo_base</b>	<b>186</b>
6.1	algo_couple_1	187
<b>7</b>	<b>/*</b>	<b>187</b>
7.1	/*	187
<b>8</b>	<b>champ_generique_base</b>	<b>187</b>
8.1	champ_post_de_champs_post	187
8.2	list_nom_virgule	188
8.3	listchamp_generique	188
8.4	champ_post_operateur_base	188
8.5	champ_post_operateur_eqn	189
8.6	champ_post_statistiques_base	189
8.7	correlation	190
8.8	champ_post_operateur_divergence	190
8.9	ecart_type	191
8.10	champ_post_extraction	191
8.11	champ_post_operateur_gradient	192
8.12	champ_post_interpolation	193
8.13	champ_post_morceau_equation	193
8.14	moyenne	194
8.15	predefini	195
8.16	champ_post_reduction_0d	195
8.17	champ_post_refchamp	196
8.18	champ_post_tparoi_vef	196
8.19	champ_post_transformation	197
<b>9</b>	<b>chimie</b>	<b>198</b>
9.1	reactions	198
9.1.1	reaction	198



<b>10</b>	<b>class_generic</b>	<b>199</b>
10.1	cholesky	199
10.2	dt_calc	199
10.3	dt_fixe	199
10.4	dt_min	200
10.5	dt_start	200
10.6	gcp_ns	200
10.7	gen	201
10.8	gmres	201
10.9	optimal	202
10.10	petsc	202
10.11	gcp	206
10.12	solveur_sys_base	207
<b>11</b>	<b>#</b>	<b>207</b>
11.1	#	207
<b>12</b>	<b>condlim_base</b>	<b>208</b>
12.1	Neumann_homogene	208
12.2	Neumann_paro_i_adiabatique	208
12.3	Paroi	208
12.4	contact_vdf_vef	208
12.5	contact_vef_vdf	209
12.6	dirichlet	209
12.7	echange_contact_rayo_transp_vdf	209
12.8	echange_contact_vdf_ft_disc	210
12.9	echange_contact_vdf_ft_disc_solid	210
12.10	entree_temperature_imposee_h	210
12.11	flux_radiatif	211
12.12	flux_radiatif_vdf	211
12.13	flux_radiatif_vef	211
12.14	frontiere_ouverte	212
12.15	frontiere_ouverte_concentration_imposee	212
12.16	frontiere_ouverte_fraction_massique_imposee	212
12.17	frontiere_ouverte_gradient_pression_impose	212
12.18	frontiere_ouverte_gradient_pression_impose_vefprep1b	213
12.19	frontiere_ouverte_gradient_pression_libre_vef	213
12.20	frontiere_ouverte_gradient_pression_libre_vefprep1b	213
12.21	frontiere_ouverte_k_eps_impose	213
12.22	frontiere_ouverte_pression_imposee	213
12.23	frontiere_ouverte_pression_imposee_orlansky	214
12.24	frontiere_ouverte_pression_moyenne_imposee	214
12.25	frontiere_ouverte_rayo_semi_transp	214
12.26	frontiere_ouverte_rayo_transp	214
12.27	frontiere_ouverte_rayo_transp_vdf	215
12.28	frontiere_ouverte_rayo_transp_vef	215
12.29	frontiere_ouverte_rho_u_impose	215
12.30	frontiere_ouverte_temperature_imposee	216
12.31	frontiere_ouverte_temperature_imposee_rayo_semi_transp	216
12.32	frontiere_ouverte_temperature_imposee_rayo_transp	216
12.33	frontiere_ouverte_vitesse_imposee	216
12.34	frontiere_ouverte_vitesse_imposee_sortie	217
12.35	neumann	217
12.36	paroi_adiabatique	217

12.37	paroi_contact	217
12.38	paroi_contact_fictif	218
12.39	paroi_decalee_robin	218
12.40	paroi_defilante	219
12.41	paroi_echange_contact_correlation_vdf	219
12.42	paroi_echange_contact_correlation_vef	220
12.43	paroi_echange_contact_odvm_vdf	221
12.44	paroi_echange_contact_rayo_semi_transp_vdf	221
12.45	paroi_echange_contact_vdf	221
12.46	paroi_echange_contact_vdf_ft	222
12.47	paroi_echange_contact_vdf_zoom_fin	222
12.48	paroi_echange_contact_vdf_zoom_grossier	223
12.49	paroi_echange_externer_impose	223
12.50	paroi_echange_externer_impose_h	223
12.51	paroi_echange_externer_impose_rayo_semi_transp	224
12.52	paroi_echange_externer_impose_rayo_transp	224
12.53	paroi_echange_global_impose	224
12.54	paroi_fixe	224
12.55	paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets	225
12.56	paroi_flux_impose	225
12.57	paroi_flux_impose_rayo_semi_transp_vdf	225
12.58	paroi_flux_impose_rayo_semi_transp_vef	225
12.59	paroi_flux_impose_rayo_transp	226
12.60	paroi_ft_disc	226
12.61	paroi_ft_disc_deriv	226
12.61.1	symetrie	226
12.61.2	constant	226
12.62	paroi_knudsen_non_negligeable	227
12.63	paroi_rugueuse	227
12.64	paroi_temperature_imposee	227
12.65	paroi_temperature_imposee_rayo_semi_transp	228
12.66	paroi_temperature_imposee_rayo_transp	228
12.67	periodique	228
12.68	scalaire_impose_pari	228
12.69	sortie_libre_rho_variable	229
12.70	sortie_libre_temperature_imposee_h	229
12.71	symetrie	229
12.72	temperature_imposee_pari	229
<b>13</b>	<b>discretisation_base</b>	<b>230</b>
13.1	ef	230
13.2	vdf	230
13.3	vef	230
13.4	vefprep1b	230
<b>14</b>	<b>domaine</b>	<b>231</b>
14.1	domaine_ale	231
<b>15</b>	<b>espece</b>	<b>231</b>

<b>16 champ_base</b>	<b>232</b>
16.1 champ_base	232
16.2 Champ_Fonc_MEDfile	232
16.3 champ_don_base	232
16.4 champ_don_lu	232
16.5 champ_fonc_fonction	232
16.6 champ_fonc_fonction_txyz	233
16.7 champ_fonc_med	233
16.8 champ_fonc_reprise	234
16.9 fonction_champ_reprise	234
16.10champ_fonc_t	234
16.11champ_fonc_tabule	235
16.12champ_init_canal_sinal	235
16.13bloc_lec_champ_init_canal_sinal	235
16.14champ_input_base	236
16.15champ_input_p0	237
16.16champ_ostwald	237
16.17champ_som_lu_vdf	237
16.18champ_som_lu_vef	238
16.19champ_tabule_temps	238
16.20champ_uniforme_morceaux	238
16.21champ_uniforme_morceaux_tabule_temps	239
16.22champ_fonc_txyz	239
16.23champ_fonc_xyz	239
16.24field_uniform_keps_from_ud	239
16.25init_par_partie	240
16.26tayl_green	240
16.27uniform_field	240
16.28valeur_totale_sur_volume	241
 <b>17 champ_front_base</b>	 <b>241</b>
17.1 champ_front_base	241
17.2 Champ_front_debit_QC_VDF	241
17.3 boundary_field_inward	242
17.4 boundary_field_uniform_keps_from_ud	242
17.5 ch_front_input	242
17.6 ch_front_input_uniforme	243
17.7 champ_front_MED	243
17.8 champ_front_ale	243
17.9 champ_front_bruite	244
17.10champ_front_calc	244
17.11champ_front_contact_rayo_semi_transp_vef	244
17.12champ_front_contact_rayo_transp_vef	245
17.13champ_front_contact_vef	245
17.14champ_front_debit	245
17.15champ_front_fonc_pois_ipsn	246
17.16champ_front_fonc_pois_tube	246
17.17champ_front_fonc_t	246
17.18champ_front_fonc_txyz	246
17.19champ_front_fonc_xyz	247
17.20champ_front_fonction	247
17.21champ_front_lu	247
17.22champ_front_normal_vef	247
17.23champ_front_pression_from_u	248

17.24	champ_front_recyclage	248
17.25	champ_front_tabule	250
17.26	champ_front_tangentiel_vef	250
17.27	champ_front_uniforme	251
17.28	champ_front_vortex	251
17.29	champ_front_zoom	251
<b>18</b>	<b>loi_etat_base</b>	<b>251</b>
18.1	gaz_reel_rhot	252
18.2	melange_gaz_parfait	252
18.3	gaz_parfait	252
<b>19</b>	<b>loi_fermeture_base</b>	<b>253</b>
19.1	loi_fermeture_test	253
<b>20</b>	<b>loi_horaire</b>	<b>253</b>
<b>21</b>	<b>milieu_base</b>	<b>254</b>
21.1	constituant	254
21.2	fluide_incompressible	254
21.3	fluide_ostwald	255
21.4	fluide_quasi_compressible	256
21.5	bloc_sutherland	257
21.6	solide	257
21.7	solide_milieu_variable	257
<b>22</b>	<b>milieu_v2_base</b>	<b>258</b>
22.1	fluide_diphasique	258
<b>23</b>	<b>modele_rayonnement_base</b>	<b>258</b>
23.1	modele_rayonnement_milieu_transparent	258
<b>24</b>	<b>modele_turbulence_scal_base</b>	<b>260</b>
24.1	prandtl	260
24.2	schmidt	261
24.3	sous_maille_dyn	262
<b>25</b>	<b>nom</b>	<b>262</b>
25.1	nom_anonyme	262
<b>26</b>	<b>partitionneur_deriv</b>	<b>263</b>
26.1	fichier_decoupage	263
26.2	metis	263
26.3	partition	264
26.4	sous_zones	264
26.5	tranche	265
<b>27</b>	<b>precond_base</b>	<b>265</b>
27.1	precondsolv	266
27.2	ssor	266
27.3	ssor_bloc	266

<b>28</b>	<b>schema_temps_base</b>	<b>267</b>
28.1	implicit_euler_steady_scheme	268
28.2	Sch_CN_EX_iteratif	270
28.3	Sch_CN_iteratif	273
28.4	scheme_euler_explicit	275
28.5	leap_frog	276
28.6	rk3_ft	278
28.7	runge_kutta_ordre_3	280
28.8	runge_kutta_ordre_4_d3p	282
28.9	runge_kutta_rationnel_ordre_2	283
28.10	schema_adams_bashforth_order_2	285
28.11	schema_adams_bashforth_order_3	287
28.12	schema_adams_moulton_order_2	288
28.13	schema_adams_moulton_order_3	291
28.14	schema_backward_differentiation_order_2	293
28.15	schema_backward_differentiation_order_3	295
28.16	scheme_euler_implicit	298
28.17	schema_implicite_base	300
28.18	schema_phase_field	302
28.19	schema_predictor_corrector	304
<b>29</b>	<b>solveur_implicite_base</b>	<b>306</b>
29.1	implicit_steady	306
29.2	implicite	307
29.3	piso	308
29.4	simple	308
29.5	simpler	309
29.6	solveur_lineaire_std	310
<b>30</b>	<b>source_base</b>	<b>310</b>
30.1	Source_Transport_K_Eps_anisotherme	311
30.2	acceleration	311
30.3	boussinesq_concentration	312
30.4	boussinesq_temperature	312
30.5	canal_perio	313
30.6	coriolis	313
30.7	darcy	313
30.8	dirac	314
30.9	forchheimer	314
30.10	perte_charge_anisotrope	314
30.11	perte_charge_circulaire	315
30.12	perte_charge_directionnelle	315
30.13	perte_charge_isotrope	316
30.14	perte_charge_reguliere	316
30.15	spec_pdcr_base	316
30.15.1	longitudinale	317
30.15.2	transversale	317
30.16	perte_charge_singuliere	317
30.17	puissance_thermique	318
30.18	source_con_phase_field	318
30.19	source_constituant	319
30.20	flottabilite	319
30.21	source_generique	320
30.22	masse_ajoutee	320

30.23	source_qdm	320
30.24	source_qdm_lambdaup	320
30.25	source_qdm_phase_field	321
30.26	source_rayo_semi_transp	321
30.27	source_robin	321
30.28	source_robin_scalaire	321
30.29	listdeuxmots_sacc	322
30.30	source_th_tdivu	322
30.31	trainee	322
30.32	source_transport_k_eps	322
30.33	source_transport_k_eps_aniso_concen	323
30.34	source_transport_k_eps_aniso_therm_concen	323
<b>31</b>	<b>sous_zone</b>	<b>323</b>
31.1	bloc_origine_cotes	324
31.2	bloc_couronne	325
31.3	bloc_tube	325
<b>32</b>	<b>turbulence_paroι_base</b>	<b>325</b>
32.1	loi_ciofalo_hydr	326
32.2	loi_expert_hydr	326
32.3	loi_puissance_hydr	326
32.4	loi_standard_hydr	326
32.5	loi_standard_hydr_old	327
32.6	loi_ww_hydr	327
32.7	negligeable	327
32.8	paroi_tble	327
32.9	twofloat	328
32.10	liste_sonde_tble	328
32.10.1	sonde_tble	328
32.11	entierfloat	329
32.12	utau_imp	329
<b>33</b>	<b>turbulence_paroι_scalaire_base</b>	<b>329</b>
33.1	loi_WW_scalaire	330
33.2	loi_analytique_scalaire	330
33.3	loi_expert_scalaire	330
33.4	loi_odvm	330
33.5	loi_paroι_nu_impose	331
33.6	loi_standard_hydr_scalaire	331
33.7	negligeable_scalaire	331
33.8	paroi_tble_scal	332
33.9	fourfloat	332
<b>34</b>	<b>listobj_impl</b>	<b>332</b>
34.1	list_un_pb	333
34.2	un_pb	333
34.3	listobj	333
<b>35</b>	<b>objet_lecture</b>	<b>333</b>
<b>36</b>	<b>index</b>	<b>334</b>

## 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 : cosine function  
SIN : sine function  
TAN : tangent function  
ATAN : arctangent function  
EXP : exponential function  
LN : natural logarithm function  
SQRT : square root function  
INT : integer function  
ERF : error function  
RND(x) : random function (values between 0 and x)  
COSH : hyperbolic cosine function  
SINH : hyperbolic sine function  
TANH : hyperbolic tangent function  
ACOS : inverse cosine function  
ATANH : inverse hyperbolic tangent function  
NOT(x) : NOT x (returns 1 if x is false, 0 otherwise)  
x\_AND\_y : boolean logical operation AND (returns 1 if both x and y are true, else 0)  
x\_OR\_y : boolean logical operation OR (returns 1 if x or y is true, else 0)  
x\_GT\_y : greater than (returns 1 if  $x > y$ , else 0)  
x\_GE\_y : greater than or equal to (returns 1 if  $x \geq y$ , else 0)  
x\_LT\_y : less than (returns 1 if  $x < y$ , else 0)  
x\_LE\_y : less than or equal to (returns 1 if  $x \leq y$ , else 0)  
x\_MIN\_y : returns the smallest of x and y  
x\_MAX\_y : returns the largest of x and y  
x\_MOD\_y : modular division of x per y  
x\_EQ\_y : equal to (returns 1 if  $x == y$ , else 0)  
x\_NEQ\_y : not equal to (returns 1 if  $x != y$ , else 0)

You can also use the following operations:

+ : addition  
- : subtraction  
/ : division  
\* : multiplication  
% : modulo  
\$ : max  
^ : power  
< : less than  
> : greater than  
[ : less than or equal to  
] : greater than or equal to

You can also use the following constants:

Pi : pi value (3,1415...)

The variables which can be used are:

x,y,z : coordinates  
t : time

**Examples:**

Champ\_front\_fonc\_txyz 2 cos(y+x^2) t+ln(y)  
 Champ\_fonc\_xyz dom 2 tanh(4\*y)\*(0.95+0.1\*rand(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 Existing & predefined fields names

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

Physical values	Keyword for field_name	Unit
Velocity	Vitesse or Velocity	$m.s^{-1}$
Kinetic energy per elements ( $0.5\rho  u_i  ^2$ )	Energie_cinetique_elem	$kg.m^{-1}.s^{-2}$
Total kinetic energy ( $\frac{\sum_{i=1}^{nb\_elem} 0.5\rho  u_i  ^2 vol_i}{\sum_{i=1}^{nb\_elem} vol_i}$ )	Energie_cinetique_totale	$kg.m^{-1}.s^{-2}$
Vorticity	Vorticite	$s^{-1}$
Pressure in incompressible flow ( $P/\rho + gz$ ) For Front Tracking probleme ( $P + \rho gz$ )	Pression <sup>1</sup>	$Pa.m^3.kg^{-1}$ or $Pa$
Pressure in incompressible flow ( $P+\rho gz$ )	Pression_pa or Pressure	$Pa$
Pressure in compressible flow	Pression	$Pa$
Hydrostatic pressure ( $\rho gz$ )	Pression_hydrostatique	$Pa$
Totale pressure (when quasi compressible model is used)=Pth+P	Pression_tot	$Pa$
Pressure gradient ( $\nabla(P/\rho + gz)$ )	Gradient_pression	$m.s^{-2}$
Temperature	Temperature	$^{\circ}C$ or K
Phase temperature of a two phases flow	Temperature_EquationName	$^{\circ}C$ or K
Mass transfer rate between two phases	Temperature_mpoint	$kg.m^{-2}.s^{-1}$
Temperature variance	Variance_Temperature	$K^2$
Temperature dissipation rate	Taux_Dissipation_Temperature	$K^2.s^{-1}$
Temperature gradient	Gradient_temperature	$K.m^{-1}$
Heat exchange coefficient	H_echange_Tref <sup>2</sup>	$W.m^{-2}.K^{-1}$
Turbulent heat flux	Flux_Chaleur_Turbulente	$m.K.s^{-1}$
Turbulent viscosity	Viscosite_turbulente	$m^2.s^{-1}$
Turbulent dynamic viscosity (when quasi compressible model is used)	Viscosite_dynamique_turbulente	$kg.m.s^{-1}$

... continued on next page ...

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

<sup>2</sup>Tref indicates the value of a reference temperature and must be specified by the user. For example, H\_echange\_293 is the keyword to use for Tref=293K.



Physical values	Keyword for field_name	Unit
Turbulent kinetic energy	<b>K</b>	$m^2.s^{-2}$
Turbulent dissipation rate	<b>Eps</b>	$m^3.s^{-1}$
Turbulent quantities K and Epsilon	<b>K_Eps</b>	$(m^2.s^{-2}, m^3.s^{-1})$
Constituent concentration	<b>Concentration</b>	
Component velocity along X	<b>VitesseX</b>	$m.s^{-1}$
Component velocity along Y	<b>VitesseY</b>	$m.s^{-1}$
Component velocity along Z	<b>VitesseZ</b>	$m.s^{-1}$
Mass balance on each cell	<b>Divergence_U</b>	$m^3.s^{-1}$
Irradiancy	<b>Irradiance</b>	$W.m^{-2}$
Q-criteria	<b>Critere_Q</b>	$s^{-1}$
Distance to the wall $Y^+ = yU/\nu$ (only computed on boundaries of wall type)	<b>Y_plus</b>	dimensionless
Friction velocity	<b>U_star</b>	$m.s^{-1}$
Cell volumes	<b>Volume_maille</b>	$m^3$
Chemical potential	<b>Potentiel_Chimique_Generalise</b>	
Source term in non Galilean referential	<b>Acceleration_terme_source</b>	$m.s^{-2}$
Stability time steps	<b>Pas_de_temps</b>	S
Listing of boundary fluxes	<b>Flux_bords</b>	cf each *.out file
Volumetric porosity	<b>Porosite_volumique</b>	dimensionless
Distance to the wall	<b>Distance_Paroi</b> <sup>3</sup>	$m$
Volumic thermal power	<b>Puissance_volumique</b>	$W.m^{-3}$
Local shear strain rate defined as $\sqrt{(2S_{ij}S_{ij})}$	<b>Taux_cisaillement</b>	$s^{-1}$
Cell Courant number (VDF only)	<b>Courant_maille</b>	dimensionless
Cell Reynolds number (VDF only)	<b>Reynolds_maille</b>	dimensionless

### 3 interprete

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

See also: [objet\\_u \(36\)](#) [read \(3.71\)](#) [associate \(3.5\)](#) [discretize \(3.23\)](#) [mailler \(3.53\)](#) [maillerparallel \(3.55\)](#) [ecrire\\_fichier\\_bin \(3.113\)](#) [ecrire \(3.112\)](#) [read\\_file \(3.72\)](#) [lire\\_tgrid \(3.74\)](#) [solve \(3.91\)](#) [execute\\_parallel \(3.28\)](#) [end \(3.41\)](#) [dimension \(3.20\)](#) [bidim\\_axi \(3.10\)](#) [axi \(3.9\)](#) [transformer \(3.103\)](#) [rotation \(3.87\)](#) [dilate \(3.19\)](#) [testeur \(3.96\)](#) [test\\_solveur \(3.95\)](#) [postraiter\\_domaine \(3.67\)](#) [modif\\_bord\\_to\\_raccord \(3.56\)](#) [remove\\_elem \(3.80\)](#) [regroupebord \(3.79\)](#) [supprime\\_bord \(3.92\)](#) [calculer\\_moments \(3.11\)](#) [imprimer\\_flux \(3.45\)](#) [decouper\\_bord\\_coincident \(3.18\)](#) [raffiner\\_anisotrope \(3.69\)](#) [raffiner\\_isotrope \(3.70\)](#) [triangler \(3.104\)](#) [tetraedriser \(3.98\)](#) [orientefacesbord \(3.60\)](#) [reorienter\\_tetraedres \(3.84\)](#) [reorienter\\_triangles \(3.85\)](#) [verifiercoin \(3.110\)](#) [porosites \(3.64\)](#) [porosites\\_champ \(3.66\)](#) [discretiser\\_domaine \(3.22\)](#) [{ \(3.16\) } \(3.42\)](#) [export \(3.29\)](#) [debog \(3.15\)](#) [pilote\\_icoco \(3.63\)](#) [moyenne\\_volumique \(3.57\)](#) [ecrire\\_champ\\_med \(3.25\)](#) [read\\_med \(3.2\)](#) [lire\\_ideas \(3.52\)](#) [ecrire\\_med \(3.114\)](#) [system \(3.94\)](#) [redresser\\_hexaedres\\_vdf \(3.77\)](#) [analyse\\_angle \(3.4\)](#) [remove\\_invalid\\_internal\\_boundaries \(3.82\)](#) [reordonner \(3.86\)](#) [precisiongeom \(3.68\)](#) [nettoiepasnoeuds \(3.58\)](#) [scatter \(3.88\)](#) [partition \(3.61\)](#) [reordonner\\_faces\\_periodiques \(3.83\)](#) [corriger\\_frontiere\\_periodique \(3.13\)](#) [distance\\_paroi \(3.24\)](#) [extruder \(3.37\)](#) [extract\\_2d\\_from\\_3d \(3.30\)](#) [extruder\\_en20 \(3.39\)](#) [extrudeparoi \(3.36\)](#) [ecriturelecturespecial \(3.27\)](#) [lata\\_to\\_med \(3.49\)](#) [lata\\_to\\_other \(3.51\)](#) [decoupebord\\_pour\\_rayonnement \(3.17\)](#) [extraire\\_plan \(3.33\)](#) [extraire\\_domaine \(3.32\)](#) [extraire\\_surface \(3.34\)](#) [integrer\\_champ\\_med \(3.47\)](#) [orienter\\_simplexes](#)

<sup>3</sup>distance\_paroi is a field which can be used only if the mixing length model (see 2.15.1.2) is used in the data file.

(3.76) `verifier_simplexes` (3.109) `verifier_qualite_raffinements` (3.107) `testeur_medcoupling` (3.97) `Raffiner_isotrope_parallele` (3.1) `option_vdf` (3.59) `interprete_geometrique_base` (3.48) `extrudebord` (3.35) `disable_TU` (3.21) `refine_mesh` (3.78) `imposer_vit_bords_ale` (3.43)

Usage:

**interprete**

### 3.1 Raffiner\_isotrope\_parallele

Description: Refine parallel mesh in parallel

See also: `interprete` (3)

Usage:

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

where

- **name\_of\_initial\_zones** *str*: name of initial Zones
- **name\_of\_new\_zones** *str*: name of new Zones
- **ascii** : writing Zones in ascii format

### 3.2 read\_med

Synonymous: **lire\_med**

Description: Keyword to read MED mesh files where `domain_name` corresponds to the domain name, `filename.med` corresponds to the file (written in format MED) containing the mesh named `mesh_name`.

Note about naming boundaries: When reading `filename.med`, TRUST will detect boundaries between domain (Raccord) when the name of the boundary begins by `type_raccord_`. For example, a boundary named `type_raccord_wall` in `filename.med` will be considered by TRUST as a boundary named `wall` between two domains.

NB: To read several domains from a mesh issued from a MED file, use `Read_Med` to read the mesh then use `Create_domain_from_sous_zone` keyword.

NB: If the MED file contains one or several subzone defined as a group of volumes, then `Read_MED` will read it and will create two files `domain_name_ssz.geo` and `domain_name_ssz_par.geo` defining the subzones for sequential and/or parallel calculations. These subzones will be read in sequential in the datafile by including (after `Read_Med` keyword) something like:

```
Read_Med ....
```

```
Read_file domain_name_ssz.geo ;
```

During the parallel calculation, you will include something:

```
Scatter { ... }
```

```
Read_file domain_name_ssz_par.geo ;
```

See also: `interprete` (3) `lire_medfile` (3.3)

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 short\_family\_names is useful to suppress FAM\_-\*\_ from the boundary names of the MED meshes.
- **nom\_dom** *str*: corresponds to the domain name
- **nom\_dom\_med** *str*: name of the mesh in med file
- **file** *str*: corresponds to the file (written in format MED) containing the mesh

### 3.3 lire\_medfile

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

See also: read\_med (3.2)

Usage:

**lire\_medfile** [ **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 short\_family\_names is useful to suppress FAM\_-\*\_ from the boundary names of the MED meshes.
- **nom\_dom** *str*: corresponds to the domain name
- **nom\_dom\_med** *str*: name of the mesh in med file
- **file** *str*: corresponds to the file (written in format MED) containing the mesh

### 3.4 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: interpret (3)

Usage:

**analyse\_angle** **domain\_name** **nb\_histo**

where

- **domain\_name** *str*: Name of domain to resequence.
- **nb\_histo** *int*

### 3.5 associate

Synonymous: **associer**

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 with error because it cannot execute the Associate (Associer) instruction. For example, to calculate water flow in a pipe, a `Pb_Hydraulique` type object needs to be defined. But also a `Domaine` type object to represent the pipe, a `Scheme_euler_explicit` type object for time discretization, a discretization type object (VDF or VEF) and a `Fluide_Incompressible` type object which will contain the water properties. These objects must then all be associated with the problem.

See also: [interpret](#) (3) [associer\\_pbmng\\_pbgglobal](#) (3.8) [associer\\_pbmng\\_pbfin](#) (3.7) [associer\\_algo](#) (3.6)

Usage:

**associate objet\_1 objet\_2**

where

- **objet\_1** *str*: `Objet_1`
- **objet\_2** *str*: `Objet_2`

### 3.6 associer\_algo

Description: This interpreter allows an algorithm to be associated with multi-grid problem.

See also: [associate](#) (3.5)

Usage:

**associer\_algo objet\_1 objet\_2**

where

- **objet\_1** *str*: `Objet_1`
- **objet\_2** *str*: `Objet_2`

### 3.7 associer\_pbmng\_pbfin

Description: This interpreter allows a local problem to be associated with multi-grid problem.

See also: [associate](#) (3.5)

Usage:

**associer\_pbmng\_pbfin objet\_1 objet\_2**

where

- **objet\_1** *str*: `Objet_1`
- **objet\_2** *str*: `Objet_2`

### 3.8 associer\_pbmng\_pbgglobal

Description: This interpreter allows a global problem to be associated with multi-grid problem.

See also: [associate](#) (3.5)

Usage:

**associer\_pbmng\_pbgglobal objet\_1 objet\_2**

where

- **objet\_1** *str*: Objet\_1
- **objet\_2** *str*: Objet\_2

### 3.9 axi

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

See also: [interpret \(3\)](#)

Usage:

**axi**

### 3.10 bidim\_axi

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

See also: [interpret \(3\)](#)

Usage:

**bidim\_axi**

### 3.11 calculer\_moments

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

See also: [interpret \(3\)](#)

Usage:

**calculer\_moments nom\_dom mot**

where

- **nom\_dom** *str*: Name of domain.
- **mot** *lecture\_bloc\_moment\_base* ([3.12](#)): Keyword.

### 3.12 lecture\_bloc\_moment\_base

Description: Auxiliary class to compute and print the moments.

See also: [objet\\_lecture \(35\)](#) [calcul \(3.12.1\)](#) [centre\\_de\\_gravite \(3.12.2\)](#)

Usage:

### 3.12.1 calcul

Description: The centre of gravity will be calculated.

See also: (3.12)

Usage:

**calcul**

### 3.12.2 centre\_de\_gravite

Description: To specify the centre of gravity.

See also: (3.12)

Usage:

**centre\_de\_gravite point**

where

- **point** *un\_point* (3.12.3): A centre of gravity.

### 3.12.3 un\_point

Description: A point.

See also: objet\_lecture (35)

Usage:

**pos**

where

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

## 3.13 corriger\_frontiere\_periodique

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

See also: `interprete` (3)

Usage:

**corriger\_frontiere\_periodique {**

**domaine** *str*

**bord** *str*

[ **direction** *n x1 x2 ... xn*]

[ **fichier\_post** *str*]

**}**

where

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

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

### 3.14 create\_domain\_from\_sous\_zone

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

See also: `interprete_geometrique_base` (3.48)

Usage:

**create\_domain\_from\_sous\_zone** {

**domaine\_final** *str*  
**par\_sous\_zone** *str*  
**domaine\_init** *str*

}

where

- **domaine\_final** *str*: new domain in which faces are stored
- **par\_sous\_zone** *str*: a sub-area allowing to choose the elements
- **domaine\_init** *str*: initial domain

### 3.15 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, an integer or an 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 both codes are less than a given value (error). If not, it prints Ok else show the differences and the lines where it occurred.

See also: `interprete` (3)

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 sequential run, and 1 for the parallel run.

### 3.16 {

Description: Block's beginning.

See also: [interprete \(3\)](#)

Usage:

{

### 3.17 decoupebord\_pour\_rayonnement

Description: To subdivide the external boundary of a domain into several parts (may be useful for better accuracy when using radiation model in transparent medium). To specify the boundaries of the fine\_domain\_name domain to be splitted. These boundaries will be cut according the coarse mesh defined by either the keyword **domaine\_grossier** (each boundary face of the coarse mesh coarse\_domain\_name will be used to group boundary faces of the fine mesh to define a new boundary), either by the keyword **nb\_parts\_naif** (each boundary of the fine mesh is splitted into a partition with nx\*ny\*nz elements), either by a geometric condition given by a formulae with the keyword **condition\_geometrique**. If used, the coarse\_domain\_name domain should have the same boundaries name of the fine\_domain\_name domain.

A mesh file (ASCII format, except if binaire option is specified) named by default newgeom (or specified by the **nom\_fichier\_sortie** keyword) will be created and will contain the fine\_domain\_name domain with the splitted boundaries named boundary\_name

See also: [interprete \(3\)](#)

Usage:

```
decoupebord_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*



### 3.18 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](#) (3)

Usage:

**decouper\_bord\_coincident domain\_name bord**

where

- **domain\_name** *str*: Name of domain.
- **bord** *str*: `connectivity_failed_boundary_name`

### 3.19 dilate

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

See also: [interpret](#) (3)

Usage:

**dilate domain\_name alpha**

where

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

### 3.20 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](#) (3)

Usage:

**dimension dim**

where

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

### 3.21 disable\_TU

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

See also: [interpret](#) (3)

Usage:

**disable\_TU**

### 3.22 discretiser\_domaine

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

See also: [interpret \(3\)](#)

Usage:

**discretiser\_domaine** **domain\_name**

where

- **domain\_name** *str*: Name of the domain.

### 3.23 discretize

Synonymous: **discretiser**

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

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

See also: [interpret \(3\)](#)

Usage:

**discretize** **problem\_name** **dis**

where

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

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

See also: [interpret \(3\)](#)

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).

### 3.25 `ecrire_champ_med`

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

See also: `interpret` (3)

Usage:

**`ecrire_champ_med nom_dom nom_chp file`**  
where

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

### 3.26 `ecrire_fichier_formatte`

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

See also: `ecrire_fichier_bin` (3.113)

Usage:

**`ecrire_fichier_formatte name_obj filename`**  
where

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

### 3.27 `ecriturelecturespecial`

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

See also: `interpret` (3)

Usage:

**`ecriturelecturespecial type`**  
where

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

### 3.28 `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` (3)

Usage:

**`execute_parallel {`**  
**`liste_cas`** *n word1 word2 ... wordn*

```
[ nb_procs  n n1 n2 ... nn]
}
```

where

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

### 3.29 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: [interpret](#) (3)

Usage:

**export**

### 3.30 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: [interpret](#) (3) [extract\\_2daxi\\_from\\_3d](#) (3.31)

Usage:

**extract\_2d\_from\_3d dom3D bord dom2D**

where

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

### 3.31 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](#) (3.30)

Usage:

**extract\_2daxi\_from\_3d dom3D bord dom2D**

where

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

### 3.32 extraire\_domaine

Description: Keyword to create a new domain built with the domain elements of the pb\_name problem verifying the two conditions given by Condition\_elements. The problem pb\_name should have been discretized.

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

See also: [interpret](#) (3)

Usage:

```
extraire_domaine {  
    domaine str  
    probleme str  
    [ condition_elements str ]  
    [ sous_zone str ]  
}
```

where

- **domaine** *str*: Domain in which faces are saved
- **probleme** *str*: Problem from which faces should be extracted
- **condition\_elements** *str*
- **sous\_zone** *str*

### 3.33 extraire\_plan

Description: This keyword extracts a plane mesh named domain\_name (this domain should have been declared before) from the mesh of the pb\_name problem. The plane can be either a triangle (defined by the keywords Origine, Point1, Point2 and Triangle), either a regular quadrangle (with keywords Origine, Point1 and Point2), or either a generalized quadrangle (with keywords Origine, Point1, Point2, Point3). The keyword Epaisseur specifies the thickness of volume around the plane which contains the faces of the extracted mesh. The keyword via\_extraire\_surface will create a plan and use Extraire\_surface algorithm. Inverse\_condition\_element keyword then will be used in the case where the plane is a boundary not well oriented, and avec\_certain\_bords\_pour\_extraire\_surface is the option related to the Extraire\_surface option named avec\_certain\_bords.

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

See also: [interpret](#) (3)

Usage:

```
extraire_plan {  
    domaine str  
    probleme str  
    epaisseur float  
    origine n x1 x2 ... xn  
    point1 n x1 x2 ... xn  
    point2 n x1 x2 ... xn  
    [ point3 n x1 x2 ... xn ]  
    [ triangle ]  
    [ via_extraire_surface ]  
    [ inverse_condition_element ]  
    [ avec_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\_certains\_bords\_pour\_extraire\_surface** *n word1 word2 ... wordn*

### 3.34 extraire\_surface

Description: This keyword extracts a surface mesh named domain\_name (this domain should have been declared before) from the mesh of the pb\_name problem. The surface mesh is defined by one or two conditions. The first condition is about elements with Condition\_elements. For example: Condition\_elements  $x*x+y*y+z*z<1$

Will define a surface mesh with external faces of the mesh elements inside the sphere of radius 1 located at (0,0,0). The second condition Condition\_faces is useful to give a restriction.

By default, the faces from the boundaries are not added to the surface mesh excepted if option avec\_les\_bords is given (all the boundaries are added), or if the option avec\_certains\_bords is used to add only some boundaries.

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

See also: [interprete \(3\)](#)

Usage:

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

where

- **domaine** *str*: Domain in which faces are saved
- **probleme** *str*: Problem from which faces should be extracted
- **condition\_elements** *str*
- **condition\_faces** *str*
- **avec\_les\_bords**
- **avec\_certains\_bords** *n word1 word2 ... wordn*

### 3.35 extrudebord

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

Warning: If the initial domain is a tetrahedral mesh, the boundary will be moved in the XY plane then

extrusion will be applied (you should maybe use the Transformer keyword on the final domain to have the domain you really want). You can use the keyword `Ecrire_Fichier_Meshtv` to generate a meshtv file to visualize your initial and final meshes.

This keyword can be used for example to create a periodic box extracted from a boundary of a tetrahedral or a hexaedral mesh. This periodic box may be used then to engender turbulent inlet flow condition for the main domain.

Note that `ExtrudeBord` in VEF generates 3 or 14 tetrahedra from extruded prisms.

See also: [interpret \(3\)](#)

Usage:

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

}

where

- **domaine\_init** *str*: Initial domain with hexaedras or tetrahedras.
- **direction** *x1 x2 (x3)*: Directions for the extrusion.
- **nb\_tranches** *int*: Number of elements in the extrusion direction.
- **domaine\_final** *str*: Extruded domain.
- **nom\_bord** *str*: Name of the boundary of the initial domain where extrusion will be applied.
- **hexa\_old** : Old algorithm for boundary extrusion from a hexahedral mesh.
- **trois\_tetra** : To extrude in 3 tetrahedras instead of 14 tetrahedras.
- **vingt\_tetra** : To extrude in 20 tetrahedras instead of 14 tetrahedras.
- **sans\_passer\_par\_le2d** *int*: Only for non-regression

### 3.36 extrudeparoi

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

See also: [interpret \(3\)](#)

Usage:

```
extrudeparoi {
    domaine str
    nom_bord str
    [ epaisseur n x1 x2 ... xn ]
    [ critere_absolu int ]
    [ projection_normale_bord ]
```

}

where

- **domaine** *str*: Name of the domain.
- **nom\_bord** *str*: Name of the (no-slip) boundary for creation of prismatic layers.
- **epaisseur** *n x1 x2 ... xn: n r1 r2 .... rn* : (relative or absolute) width for each layer.
- **critere\_absolu** *int*: relative (0, the default) or absolute (1) width for each layer.
- **projection\_normale\_bord** : keyword to project layers on the same plane that contiguous boundaries. default values are : epaisseur\_relative 1 0.5 projection\_normale\_bord 1

### 3.37 extruder

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

See also: [interprete \(3\)](#) [extruder\\_en3 \(3.40\)](#)

Usage:

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

- **domaine** *str*: Name of the domain.
- **direction** *troisf* ([3.38](#)): Direction of the extrude operation.
- **nb\_tranches** *int*: Number of elements in the extrusion direction.

### 3.38 troisf

Description: Auxiliary class to extrude.

See also: [objet\\_lecture \(35\)](#)

Usage:

```
lx ly lz
where
```

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

### 3.39 extruder\_en20

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

See also: [interprete \(3\)](#)

Usage:

```
extruder_en20 {
    domaine str
```



```

    [ direction troisf]
    nb_tranches int
}

```

where

- **domaine** *str*: Name of the domain.
- **direction** *troisf* (3.38): 0 Direction of the extrude operation.
- **nb\_tranches** *int*: Number of elements in the extrusion direction.

### 3.40 extruder\_en3

Description: Class to create a 3D tetrahedral/hexahedral mesh (a prism is cut in 3) from a 2D triangular/quadrangular mesh. The names of the boundaries (by default, *devant* (front) and *derriere* (back)) may be edited by the keyword **nom\_cl\_devant** and **nom\_cl\_derriere**. If NULL is written for **nom\_cl**, then no boundary condition is generated at this place.

Recommendation : to ensure conformity between meshes (in case of fluid/solid coupling) it is recommended to extrude all the domains at the same time.

See also: **extruder** (3.37)

Usage:

```

extruder_en3 {
    domaine n word1 word2 ... wordn
    [ nom_cl_devant str]
    [ nom_cl_derriere str]
    direction troisf
    nb_tranches int
}

```

where

- **domaine** *n word1 word2 ... wordn*: List of the domains
- **nom\_cl\_devant** *str*: New name of the first boundary.
- **nom\_cl\_derriere** *str*: New name of the second boundary.
- **direction** *troisf* (3.38) for inheritance: Direction of the extrude operation.
- **nb\_tranches** *int* for inheritance: Number of elements in the extrusion direction.

### 3.41 end

Synonymous: **fin**

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

See also: **interpret** (3)

Usage:

**end**

### 3.42 }

Description: Block's end.

See also: [interpret](#) (3)

Usage:  
}

### 3.43 **imposer\_vit\_bords\_ale**

Description: not\_set

See also: [interpret](#) (3)

Usage:  
**imposer\_vit\_bords\_ale** **dom** **bloc**  
where

- **dom** *str*: Name of domain.
- **bloc** *bloc\_lecture* (3.44): Description.

### 3.44 **bloc\_lecture**

Description: to read between two braces

See also: [objet\\_lecture](#) (35)

Usage:  
**bloc\_lecture**  
where

- **bloc\_lecture** *str*

### 3.45 **imprimer\_flux**

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

See also: [interpret](#) (3) [imprimer\\_flux\\_sum](#) (3.46)

Usage:  
**imprimer\_flux** **domain\_name** **noms\_bord**  
where

- **domain\_name** *str*: Name of the domain.
- **noms\_bord** *bloc\_lecture* (3.44): List of boundaries, for ex: { Bord1 Bord2 }

### 3.46 imprimer\_flux\_sum

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

See also: imprimer\_flux ([3.45](#))

Usage:

**imprimer\_flux\_sum** **domain\_name** **noms\_bord**

where

- **domain\_name** *str*: Name of the domain.
- **noms\_bord** *bloc\_lecture* ([3.44](#)): List of boundaries, for ex: { Bord1 Bord2 }

### 3.47 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  $z=zmin$  and  $z=zmax$  on nb\_tranche surfaces. The output file indicates first the flow rate for the whole surface and then lists for each tranche : the height z, the surface average value, the surface area and the flow rate. For the debit\_total method, only one tranche is considered.

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

See also: interprete ([3](#))

Usage:

```
integrer_champ_med {  
    champ_med str  
    methode str into ['integrale_en_z', 'debit_total']  
    [ zmin float]  
    [ zmax float]  
    [ nb_tranche int]  
    [ fichier_sortie str]  
}
```

where

- **champ\_med** *str*
- **methode** *str* into ['integrale\_en\_z', 'debit\_total']: to choose between the integral following z or over the entire height (debit\_total corresponds to  $zmin=-D_{MAXFLOAT}$ ,  $ZMax=D_{MAXFLOAT}$ , nb\_tranche=1)
- **zmin** *float*
- **zmax** *float*
- **nb\_tranche** *int*
- **fichier\_sortie** *str*: name of the output file, by default: integrale.

### 3.48 interprete\_geometrique\_base

Description: Class for interpreting a data file

See also: interprete ([3](#)) create\_domain\_from\_sous\_zone ([3.14](#))

Usage:

**interprete\_geometrique\_base**

### 3.49 lata\_to\_med

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

See also: [interpret](#) (3)

Usage:

**lata\_to\_med** [ **format** ] **file** **file\_med**

where

- **format** *format\_lata\_to\_med* (3.50): generated file post\_med.data use format (MED or LATA or LML keyword).
- **file** *str*: LATA file to convert to the new format.
- **file\_med** *str*: Name of the MED file.

### 3.50 format\_lata\_to\_med

Description: not\_set

See also: [objet\\_lecture](#) (35)

Usage:

**mot** [ **format** ]

where

- **mot** *str* into ['format\_post\_sup']
- **format** *str* into ['lml', 'lata', 'lata\_v1', 'lata\_v2', 'med']: generated file post\_med.data use format (MED or LATA or LML keyword).

### 3.51 lata\_to\_other

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

See also: [interpret](#) (3)

Usage:

**lata\_to\_other** [ **format** ] **file** **file\_post**

where

- **format** *str* into ['lml', 'lata', 'lata\_v1', 'lata\_v2', 'med']: Results format (MED or LATA or LML keyword).
- **file** *str*: LATA file to convert to the new format.
- **file\_post** *str*: Name of file post.

### 3.52 lire\_ideas

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

See also: [interpret](#) (3)

Usage:

**lire\_ideas** **nom\_dom** **file**

where

- **nom\_dom** *str*: Name of domain.
- **file** *str*: Name of file.

### 3.53 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 \(3\)](#)

Usage:

**mailler domaine bloc**

where

- **domaine** *str*: Name of domain.
- **bloc** *list\_bloc\_mailler* ([3.54](#)): Instructions to mesh.

### 3.54 list\_bloc\_mailler

Description: List of block mesh.

See also: [listobj \(34.3\)](#)

Usage:

{ *object1* , *object2* .... }

list of *mailler\_base* ([3.54.1](#)) separated with ,

#### 3.54.1 mailler\_base

Description: Basic class to mesh.

See also: [objet\\_lecture \(35\)](#) [pave \(3.54.2\)](#) [epsilon \(3.54.12\)](#) [domain \(3.54.13\)](#)

Usage:

#### 3.54.2 pave

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

See also: [mailler\\_base \(3.54.1\)](#)

Usage:

**pave name bloc list\_bord**

where

- **name** *str*: Name of the pave (block).
- **bloc** *bloc\_pave* ([3.54.3](#)): Definition of the pave (block).
- **list\_bord** *list\_bord* ([3.54.4](#)): Domain boundaries definition.

### 3.54.3 bloc\_pave

Description: Class to create a pave.

See also: [objet\\_lecture \(35\)](#)

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 coordinate system).
- **longueurs** *x1 x2 (x3)*: Keyword to define the block dimensions, that is to say knowing the origin, length along the axes.
- **nombre\_de\_noeuds** *n1 n2 (n3)*: Keyword to define the discretization (nodenummer) in each direction.
- **facteurs** *x1 x2 (x3)*: Keyword to define stretching factors for mesh discretization in each direction. This is a real number which must be positive (by default 1.0). A stretching factor other than 1 allows refinement on one edge in one direction.
- **symx**: Keyword to define a block mesh that is symmetrical with respect to the YZ plane (respectively Y-axis in 2D) passing through the block centre.
- **symy**: Keyword to define a block mesh that is symmetrical with respect to the XZ plane (respectively X-axis in 2D) passing through the block centre.
- **symz**: Keyword defining a block mesh that is symmetrical with respect to the XY plane passing through the block centre.
- **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.

### 3.54.4 list\_bord

Description: The block sides.

See also: [listobj \(34.3\)](#)

Usage:

```
{ object1 object2 .... }  
list of bord_base (3.54.5)
```

### 3.54.5 bord\_base

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

See also: [objet\\_lecture \(35\)](#) [bord \(3.54.6\)](#) [raccord \(3.54.10\)](#) [internes \(3.54.11\)](#)

Usage:

### 3.54.6 bord

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

See also: [bord\\_base \(3.54.5\)](#)

Usage:

**bord nom defbord**

where

- **nom** *str*: Name of block side.
- **defbord** *defbord (3.54.7)*: Definition of block side.

### 3.54.7 defbord

Description: Class to define an edge.

See also: [objet\\_lecture \(35\)](#) [defbord\\_2 \(3.54.8\)](#) [defbord\\_3 \(3.54.9\)](#)

Usage:

### 3.54.8 defbord\_2

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

See also: [\(3.54.7\)](#)

Usage:

**dir eq pos pos2\_min inf1 dir2 inf2 pos2\_max**

where

- **dir** *str into ['X', 'Y']*: Edge is perpendicular to this direction.
- **eq** *str into ['=']*: Equality sign.
- **pos** *float*: Position value.
- **pos2\_min** *float*: Minimal value.
- **inf1** *str into ['<=']*: Less than or equal to sign.
- **dir2** *str into ['X', 'Y']*: Edge is parallel to this direction.
- **inf2** *str into ['<=']*: Less than or equal to sign.
- **pos2\_max** *float*: Maximal value.

### 3.54.9 defbord\_3

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

See also: [\(3.54.7\)](#)

Usage:

**dir eq pos pos2\_min inf1 dir2 inf2 pos2\_max pos3\_min inf3 dir3 inf4 pos3\_max**  
 where

- **dir** *str* into ['X', 'Y', 'Z']: Edge is perpendicular to this direction.
- **eq** *str* into ['=']: Equality sign.
- **pos** *float*: Position value.
- **pos2\_min** *float*: Minimal value.
- **inf1** *str* into ['<=']: Less than or equal to sign.
- **dir2** *str* into ['X', 'Y']: Edge is parallel to this direction.
- **inf2** *str* into ['<=']: Less than or equal to sign.
- **pos2\_max** *float*: Maximal value.
- **pos3\_min** *float*: Minimal value.
- **inf3** *str* into ['<=']: Less than or equal to sign.
- **dir3** *str* into ['Y', 'Z']: Edge is parallel to this direction.
- **inf4** *str* into ['<=']: Less than or equal to sign.
- **pos3\_max** *float*: Maximal value.

### 3.54.10 raccord

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

See also: `bord_base` ([3.54.5](#))

Usage:

**raccord type1 type2 nom defbord**  
 where

- **type1** *str* into ['local', 'distant']: Contact type.
- **type2** *str* into ['homogene']: Contact type.
- **nom** *str*: Name of block side.
- **defbord** *defbord* ([3.54.7](#)): Definition of block side.

### 3.54.11 internes

Description: To indicate that the block has a set of internal faces (these faces will be duplicated automatically by the program and will be processed in a manner similar to edge faces).

Two boundaries with the same boundary conditions may have the same name (whether or not they belong to the same block).

The keyword Internes (Internal) must be used to execute a calculation with plates, followed by the equation of the surface area covered by the plates.

See also: `bord_base` ([3.54.5](#))

Usage:

**internes nom defbord**  
 where

- **nom** *str*: Name of block side.
- **defbord** *defbord* ([3.54.7](#)): Definition of block side.



### 3.54.12 epsilon

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

See also: `mailler_base` ([3.54.1](#))

Usage:

**epsilon** **eps**

where

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

### 3.54.13 domain

Description: Class to reuse a domain.

See also: `mailler_base` ([3.54.1](#))

Usage:

**domain** **domain\_name**

where

- **domain\_name** *str*: Name of domain.

## 3.55 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` ([3](#))

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*

### 3.56 `modif_bord_to_raccord`

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

See also: [interprete \(3\)](#)

Usage:

```
modif_bord_to_raccord domaine nom_bord
```

where

- **domaine** *str*: Name of domain
- **nom\_bord** *str*: Name of the boundary to transform.

### 3.57 `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: [interpret \(3\)](#)

Usage:

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

where

- **nom\_pb** *str*: name of the problem where the source fields will be searched.
- **nom\_domaine** *str*: name of the destination domain (for example, it can be a coarser mesh, but for optimal performance in parallel, the domain should be split with the same algorithm as the computation mesh, eg, same *tranche* parameters for example)
- **noms\_champs** *n word1 word2 ... wordn*: name of the source fields (these fields must be accessible from the *postraitement*) N *source\_field1 source\_field2 ... source\_fieldN*
- **nom\_fichier\_post** *str*: indicates the filename where the result is written
- **format\_post** *str*: gives the fileformat for the result (by default : *lata*)
- **localisation** *str* into [ 'elem', 'som' ]: indicates where the convolution product should be computed: either on the elements or on the nodes of the destination domain.
- **fonction\_filtre** *bloc\_lecture* (3.44): to specify the given filter

```
Fonction_filtre {  
    type filter_type  
    demie-largeur l  
    [ omega w ]  
    [ expression string ]  
}
```

type filter\_type : This parameter specifies the filtering function. Valid filter\_type are:

Boite is a box filter,  $f(x, y, z) = (abs(x) < l) * (abs(y) < l) * (abs(z) < l) / (8l^3)$

Chapeau is a hat filter (product of hat filters in each direction) centered on the origin, the half-width of the filter being l and its integral being 1.

Quadra is a 2nd order filter.

Gaussienne is a normalized gaussian filter of standard deviation sigma in each direction (all field elements outside a cubic box defined by *clipping\_half\_width* are ignored, hence, taking *clipping\_half\_width*=2.5\*sigma yields an integral of 0.99 for a uniform unity field).

Parser allows a user defined function of the x,y,z variables. All elements outside a cubic box defined by *clipping\_half\_width* are ignored. The parser is much slower than the equivalent c++ coded function...

demie-largeur l : This parameter specifies the half width of the filter

[ omega w ] : This parameter must be given for the gaussienne filter. It defines the standard deviation of the gaussian filter.

[ expression string ] : This parameter must be given for the parser filter type. This expression will be interpreted by the math parser with the predefined variables x, y and z.

### 3.58 nettoiepasnoeuds

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

See also: [interpret \(3\)](#)

Usage:

**nettoiepasnoeuds** **domain\_name**  
where

- **domain\_name** *str*: Name of domain.

### 3.59 option\_vdf

Description: Class of VDF options.

See also: [interpret \(3\)](#)

Usage:

**option\_vdf** {  
    [ **traitement\_coins** *str* into ['oui', 'non']]  
    [ **p\_imposee\_aux\_faces** *str* into ['oui', 'non']]  
}

where

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

### 3.60 orientefacesbord

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

See also: [interpret \(3\)](#)

Usage:

**orientefacesbord** **domain\_name**  
where

- **domain\_name** *str*: Name of domain.

### 3.61 partition

Synonymous: **decouper**

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

See also: [interpret \(3\)](#)

Usage:

**partition domaine bloc\_decouper**  
where

- **domaine** *str*: Name of the domain to be cut.
- **bloc\_decouper** *bloc\_decouper* ([3.62](#)): Description how to cut a domain.

### 3.62 bloc\_decouper

Description: Auxiliary class to cut a domain.

See also: [objet\\_lecture \(35\)](#)

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\_toolpartitionneur** *partitionneur\_deriv* ([26](#)): 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 disk (you might just want to generate a 'ecrire\_decoupage' or 'ecrire\_lata').
- **ecrire\_decoupage** *str*: After having called the partitionning algorithm, the resulting partition is written on disk in the specified filename. See also *partitionneur Fichier\_Decoupage*. This keyword is useful to change the partition numbers: first, you write the partition into a file with the option *ecrire\_decoupage*. This file contains the zone number for each element's mesh. Then you can easily permute zone numbers in this file. Then read the new partition to create the .Zones files with the *Fichier\_Decoupage* keyword.
- **ecrire\_lata** *str*
- **nb\_parts\_tot** *int*: Keyword to generates N .Zone files, instead of the default number M obtained after the partitionning algorithm. N must be greater or equal to M. This option might be used to perform coupled parallel computations. Supplemental empty zones from M to N-1 are created. This

keyword is used when you want to run a parallel calculation on several domains with for example, 2 processors on a first domain and 10 on the second domain because the first domain is very small compare to second one. You will write Nb\_parts 2 and Nb\_parts\_tot 10 for the first domain and Nb\_parts 10 for the second domain.

- **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.

### 3.63 pilote\_icoco

Description: not\_set

See also: interpret (3)

Usage:

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

where

- **pb\_name** *str*
- **main** *str*

### 3.64 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 (3)

Usage:

```
porosites pb sous_zone bloc
where
```

- **pb** *str*: Name of the problem to which the sub-area is attached.
- **sous\_zone** *str*: Name of the sub-area to which porosity are allocated.
- **bloc** *bloc\_lecture\_poro* (3.65): Surface and volume porosity values.

### 3.65 bloc\_lecture\_poro

Description: Surface and volume porosity values.

See also: objet\_lecture (35)

Usage:

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

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

### 3.66 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 Discretize should have already been used to read the object.

See also: interpret (3)

Usage:

```
porosites_champ pb ch  
where
```

- **pb** *str*: Name of the problem to which the sub-area is attached.
- **ch** *champ\_base* (16.1): field used to define the porosity field

### 3.67 postraiter\_domaine

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

See also: interpret (3)

Usage:

```
postraiter_domaine {  
    format str into ['lml', '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', '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* (3.44): Names of domains : { name1 name2 }
- **joints\_non\_postraites** *int into [0, 1]*: The joints\_non\_postraites (1 by default) will not write the boundaries between the partitioned mesh.
- **binaire** *int into [0, 1]*: Binary (binaire 1) or ASCII (binaire 0) may be used. By default, it is 0 for LATA and only ASCII is available for LML and only binary is available for MED.
- **ecrire\_frontiere** *int into [0, 1]*: This option will write (if set to 1, the default) or not (if set to 0) the boundaries as fields into the file (it is useful to not add the boundaries when writing a domain extracted from another domain)

### 3.68 precisiongeom

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

See also: [interpret](#) (3)

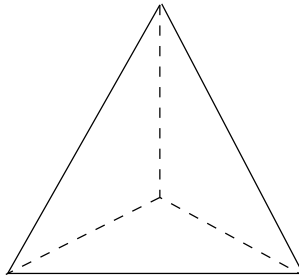
Usage:

**precisiongeom precision**  
where

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

### 3.69 raffiner\_anisotrope

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



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

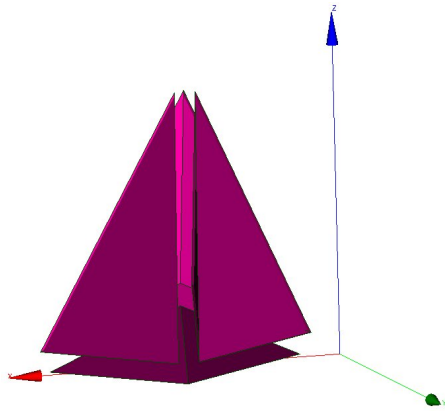
See also: [interpret](#) (3)

Usage:

**raffiner\_anisotrope domain\_name**  
where

- **domain\_name** *str*: Name of domain.

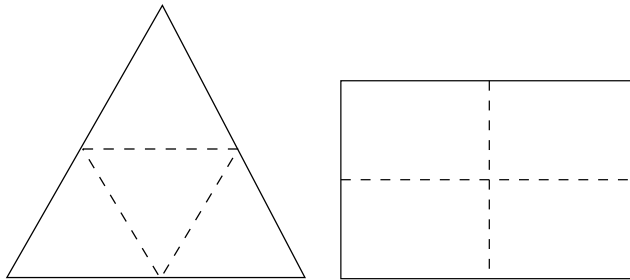




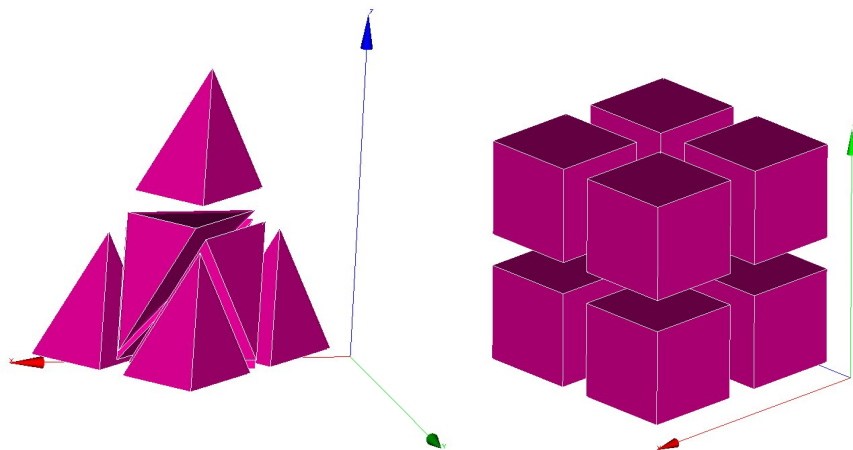
### 3.70 raffiner\_isotrope

Synonymous: **raffiner\_simplexes**

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



For 3D elements:



See also: [interpret \(3\)](#)

Usage:

**raffiner\_isotrope domain\_name**

where

- **domain\_name** *str*: Name of domain.

### 3.71 read

Synonymous: **lire**

Description: Interpreter to read the a\_object object defined between the braces.

See also: [interpret](#) (3)

Usage:

**read a\_object bloc**

where

- **a\_object** *str*: Object to be read.
- **bloc** *str*: Definition of the object.

### 3.72 read\_file

Synonymous: **lire\_fichier**

Description: Keyword to read the object name\_obj contained in the file filename.

This is notably used when the calculation domain has already been meshed and the mesh contains the file filename, simply write read\_file dom filename (where dom is the name of the meshed domain).

If the filename is ;, is to execute a data set given in the file of name name\_obj (a space must be entered between the semi-colon and the file name).

See also: [interpret](#) (3) [read\\_unsupported\\_ascii\\_file\\_from\\_icem](#) (3.75) [read\\_file\\_binary](#) (3.73)

Usage:

**read\_file name\_obj filename**

where

- **name\_obj** *str*: Name of the object to be read.
- **filename** *str*: Name of the file.

### 3.73 read\_file\_binary

Synonymous: **lire\_fichier\_bin**

Description: Keyword to read an object name\_obj in the unformatted type file filename.

See also: [read\\_file](#) (3.72)

Usage:

**read\_file\_binary name\_obj filename**

where

- **name\_obj** *str*: Name of the object to be read.
- **filename** *str*: Name of the file.

### 3.74 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 \(3\)](#)

Usage:

**lire\_tgrid dom filename**

where

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

### 3.75 read\_unsupported\_ascii\_file\_from\_icem

Description: not\_set

See also: [read\\_file \(3.72\)](#)

Usage:

**read\_unsupported\_ascii\_file\_from\_icem name\_obj filename**

where

- **name\_obj** *str*: Name of the object to be read.
- **filename** *str*: Name of the file.

### 3.76 orienter\_simplexes

Synonymous: **rectify\_mesh**

Description: Keyword to refine a mesh

See also: [interpret \(3\)](#)

Usage:

**orienter\_simplexes domain\_name**

where

- **domain\_name** *str*: Name of domain.

### 3.77 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 \(3\)](#)

Usage:

**redresser\_hexaedres\_vdf domain\_name**

where

- **domain\_name** *str*: Name of domain to resequence.

### 3.78 refine\_mesh

Description: not\_set

See also: interpret (3)

Usage:

**refine\_mesh** **domaine**

where

- **domaine** *str*

### 3.79 regroupebord

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

See also: interpret (3)

Usage:

**regroupebord** **domaine** **new\_bord** **bords**

where

- **domaine** *str*: Name of domain
- **new\_bord** *str*: Name of the new boundary
- **bords** *bloc\_lecture* (3.44): { Bound1 Bound2 }

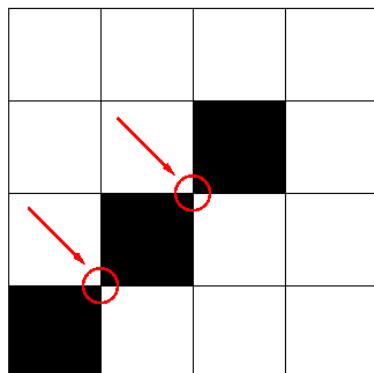
### 3.80 remove\_elem

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  $(x,y)=(0.5,0.5)$ :

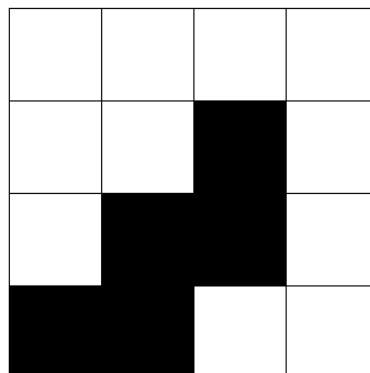
Remove\_elem dom { fonction  $0.2 * 0.2 - (x - 0.5)^2 - (y - 0.5)^2 > 0$  }

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 \(3\)](#)

Usage:

**remove\_elem domaine bloc**

where

- **domaine** *str*: Name of domain
- **bloc** *remove\_elem\_bloc* ([3.81](#))

### 3.81 remove\_elem\_bloc

Description: not\_set

See also: [objet\\_lecture \(35\)](#)

Usage:

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

where

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

### 3.82 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 \(3\)](#)

Usage:

**remove\_invalid\_internal\_boundaries domain\_name**

where

- **domain\_name** *str*: Name of domain.

### 3.83 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 \(3\)](#)

Usage:

**reordonner\_faces\_periodiques domaine nom\_bord\_perio**

where

- **domaine** *str*: Name of domain.
- **nom\_bord\_perio** *str*: boundary\_name.

### 3.84 reorienter\_tetraedres

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

See also: [interpret \(3\)](#)

Usage:

**reorienter\_tetraedres domain\_name**

where

- **domain\_name** *str*: Name of domain.

### 3.85 reorienter\_triangles

Description: not\_set

See also: [interpret \(3\)](#)

Usage:

**reorienter\_triangles domain\_name**

where

- **domain\_name** *str*: Name of domain.

### 3.86 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:

Read\_file dom fichier.geom

Reordonner dom

Observations: This keyword is redundant when the mesh that is read is correctly sequenced in the TRUST sense. This significant mesh operation may take some time... The message returned by TRUST is not explicit when the Reordonner (Resequencing) keyword is required but not included in the data set...

See also: [interpret \(3\)](#)

Usage:

**reordonner domain\_name**

where

- **domain\_name** *str*: Name of domain to resequence.

### 3.87 rotation

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

See also: [interpret \(3\)](#)

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)

### 3.88 scatter

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

See also: [interpret](#) (3) [scatterformatte](#) (3.89) [scattermed](#) (3.90)

Usage:

**scatter file domaine**

where

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

### 3.89 scatterformatte

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

See also: [scatter](#) (3.88)

Usage:

**scatterformatte file domaine**

where

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

### 3.90 scattermed

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

See also: [scatter](#) (3.88)

Usage:

**scattermed file domaine**

where

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

### 3.91 solve

Synonymous: **resoudre**

Description: Interpreter to start calculation with TRUST.

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

See also: [interpret \(3\)](#)

Usage:

**solve pb**

where

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

### 3.92 supprime\_bord

Description: Keyword to remove boundaries (named Boundary\_name1 Boundary\_name2 ) of the domain named domain\_name.

See also: [interpret \(3\)](#)

Usage:

**supprime\_bord domaine bords**

where

- **domaine** *str*: Name of domain
- **bords** *list\_nom* ([3.93](#)): { Boundary\_name1 Boundary\_name2 }

### 3.93 list\_nom

Description: List of name.

See also: [listobj \(34.3\)](#)

Usage:

{ object1 object2 .... }

list of *nom\_anonyme* ([25.1](#))

### 3.94 system

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

See also: [interpret \(3\)](#)

Usage:

**system cmd**

where

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



### 3.95 test\_solveur

Description: To test several solvers

See also: [interprete \(3\)](#)

Usage:

```
test_solveur {  
    [ fichier_secmem  str]  
    [ fichier_matrice str]  
    [ fichier_solution str]  
    [ nb_test  int]  
    [ impr  ]  
    [ solveur  solveur_sys_base]  
    [ fichier_solveur str]  
    [ genere_fichier_solveur float]  
    [ seuil_verification float]  
    [ pas_de_solution_initiale ]  
    [ ascii  ]  
}
```

where

- **fichier\_secmem** *str*: Filename containing the second member B
- **fichier\_matrice** *str*: Filename containing the matrix A
- **fichier\_solution** *str*: Filename containing the solution x
- **nb\_test** *int*: Number of tests to measure the time resolution (one preconditionnement)
- **impr** : To print the convergence solver
- **solveur** *solveur\_sys\_base* ([10.12](#)): 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

### 3.96 testeur

Description: not\_set

See also: [interprete \(3\)](#)

Usage:

```
testeur data  
where
```

- **data** *bloc\_lecture* ([3.44](#))

### 3.97 testeur\_medcoupling

Description: not\_set

See also: [interprete \(3\)](#)

Usage:

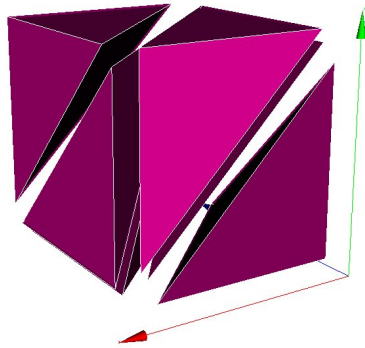
**testeur\_medcoupling pb\_name field\_name**

where

- **pb\_name** *str*: Name of domain.
- **field\_name** *str*: Name of domain.

### 3.98 tetraedriser

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



See also: [interpret \(3\)](#) [tetraedriser\\_homogene \(3.99\)](#) [tetraedriser\\_homogene\\_fin \(3.101\)](#) [tetraedriser\\_homogene\\_compact \(3.100\)](#) [tetraedriser\\_par\\_prisme \(3.102\)](#)

Usage:

**tetraedriser domain\_name**

where

- **domain\_name** *str*: Name of domain.

### 3.99 tetraedriser\_homogene

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

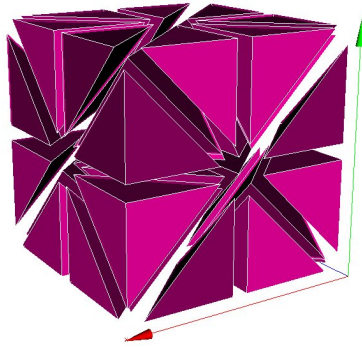
See also: [tetraedriser \(3.98\)](#)

Usage:

**tetraedriser\_homogene domain\_name**

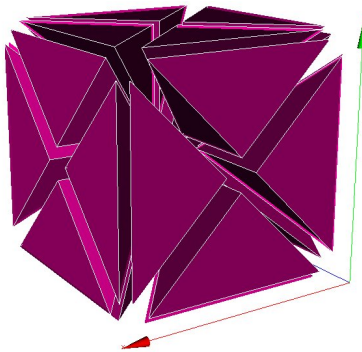
where

- **domain\_name** *str*: Name of domain.



### 3.100 **tetraedriser\_homogene\_compact**

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



See also: `tetraedriser` ([3.98](#))

Usage:

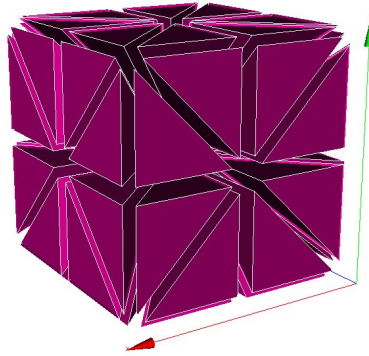
**tetraedriser\_homogene\_compact** **domain\_name**  
where

- **domain\_name** *str*: Name of domain.

### 3.101 **tetraedriser\_homogene\_fin**

Description: `Tetraedriser_homogene_fin` is the recommended option to tetrahedralise blocks. As an extension (subdivision) of `Tetraedriser_homogene_compact`, this last one cut each initial block in 48 tetrahedra (against 24, previously). This cutting ensures :

- a correct cutting in the corners (in respect to pressure discretization PreP1B),
- a better isotropy of elements than with `Tetraedriser_homogene_compact`,
- a better alignment of summits (this could have a benefit effect on calculation near walls since first elements in contact with it are all contained in the same constant thickness and ii/ by the way, a 3D cartesian grid based on summits can be engendered and used to realise spectral analysis in HIT for instance). Initial block is divided in 48 tetrahedra:



See also: [tetraedriser \(3.98\)](#)

Usage:

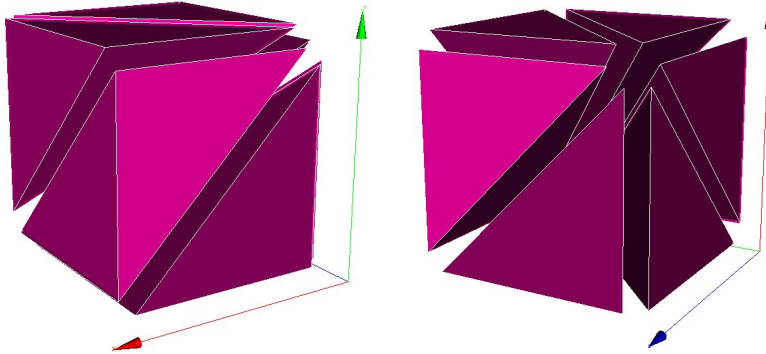
**tetraedriser\_homogene\_fin** **domain\_name**

where

- **domain\_name** *str*: Name of domain.

### 3.102 **tetraedriser\_par\_prisme**

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



Initial block is divided in 6 prisms.

See also: [tetraedriser \(3.98\)](#)

Usage:

**tetraedriser\_par\_prisme** **domain\_name**

where

- **domain\_name** *str*: Name of domain.

### 3.103 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 [\(3\)](#)

Usage:

**transformer domain\_name formule**

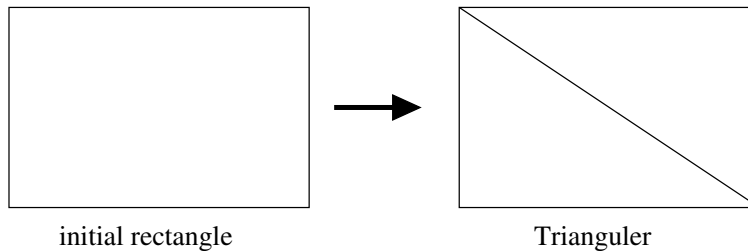
where

- **domain\_name** *str*: Name of domain.
- **formule** *word1 word2 (word3)*: Function\_for\_x Function\_for\_y

*Function\_forz*

### 3.104 trianguler

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



See also: interpret [\(3\)](#) [trianguler\\_h \(3.106\)](#) [trianguler\\_fin \(3.105\)](#)

Usage:

**trianguler domain\_name**

where

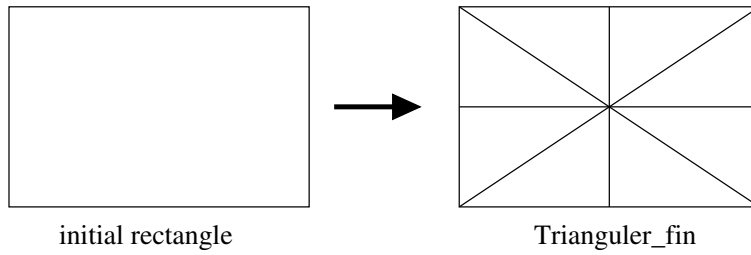
- **domain\_name** *str*: Name of domain.

### 3.105 trianguler\_fin

Description: Trianguler\_fin is the recommended option to triangulate rectangles.

As an extension (subdivision) of Trianguler\_h option, this one cut each initial rectangle in 8 triangles (against 4, previously). This cutting ensures :

- a correct cutting in the corners (in respect to pressure discretization PreP1B).
- a better isotropy of elements than with Trianguler\_h option.
- a better alignment of summits (this could have a benefit effect on calculation near walls since first elements in contact with it are all contained in the same constant thickness, and, by this way, a 2D cartesian grid based on summits can be engendered and used to realize statistical analysis in plane channel configuration for instance). Principle:



See also: [triangler](#) (3.104)

Usage:

**triangler\_fin** **domain\_name**

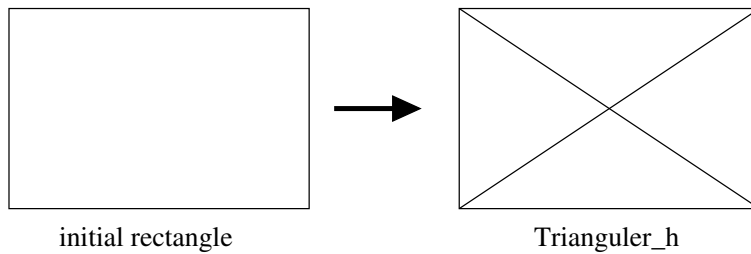
where

- **domain\_name** *str*: Name of domain.

### 3.106 **triangler\_h**

Description: To achieve a triangular mesh from a mesh comprising rectangles (4 triangles per rectangle).

Should be used in VEF discretization. Principle:



See also: [triangler](#) (3.104)

Usage:

**triangler\_h** **domain\_name**

where

- **domain\_name** *str*: Name of domain.

### 3.107 **verifier\_qualite\_raffinements**

Description: not\_set

See also: [interpret](#) (3)

Usage:

**verifier\_qualite\_raffinements** **domain\_names**

where

- **domain\_names** *vect\_nom* (3.108)

### 3.108 vect\_nom

Description: Vect of name.

See also: listobj (34.3)

Usage:

n object1 object2 ....

list of *nom\_anonyme* (25.1)

### 3.109 verifier\_simplexes

Description: Keyword to raffine a simplexes

See also: interpret (3)

Usage:

**verifier\_simplexes** **domain\_name**

where

- **domain\_name** *str*: Name of domain.

### 3.110 verifiercoin

Description: This keyword subdivides inconsistent 2D/3D cells used with VEFPreP1B discretization. Must be used before the mesh is discretized. The Read\_file option can be used only if the file.decoupage\_som was previously created by TRUST. This option, only in 2D, reverses the common face at two cells (at least one is inconsistent), through the nodes opposed. In 3D, the option has no effect.

The expert\_only option deactivates, into the VEFPreP1B divergence operator, the test of inconsistent cells.

See also: interpret (3)

Usage:

**verifiercoin** **domain\_name** **bloc**

where

- **domain\_name** *str*: Name of the domaine
- **bloc** *verifiercoin\_bloc* (3.111)

### 3.111 verifiercoin\_bloc

Description: not\_set

See also: objet\_lecture (35)

Usage:

{

[ **Lire\_fichier**|**Read\_file** *str*]

[ **expert\_only** ]

}  
where

- **Lire\_fichier|Read\_file** *str*: name of the \*.decoupage\_som file
- **expert\_only** : to not check the mesh

### 3.112 ecrire

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

See also: interpreté (3)

Usage:

**ecrire name\_obj**  
where

- **name\_obj** *str*: Name of the object to be written.

### 3.113 ecrire\_fichier\_bin

Synonymous: **ecrire\_fichier**

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

See also: interpreté (3) ecrire\_fichier\_formatte (3.26)

Usage:

**ecrire\_fichier\_bin name\_obj filename**  
where

- **name\_obj** *str*: Name of the object to be written.
- **filename** *str*: Name of the file.

### 3.114 ecrire\_med

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

See also: interpreté (3) ecrire\_medfile (3.115)

Usage:

**ecrire\_med nom\_dom file**  
where

- **nom\_dom** *str*: Name of domain.
- **file** *str*: Name of file.



### 3.115 ecrire\_medfile

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

See also: `ecrire_med` (3.114)

Usage:

**ecrire\_medfile** **nom\_dom** **file**

where

- **nom\_dom** *str*: Name of domain.
- **file** *str*: Name of file.

## 4 pb\_gen\_base

Description: Basic class for problems.

See also: `objet_u` (36) `Pb_base` (4.1) `probleme_couple` (4.7) `pbc_med` (4.37) `pb_mg` (4.22)

Usage:

### 4.1 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 Discretize should have already been used to read the object.

See also: `pb_gen_base` (4) `pb_thermohydraulique` (4.25) `pb_hydraulique` (4.16) `pb_hydraulique_turbulent` (4.21) `pb_thermohydraulique_turbulent` (4.33) `pb_conduction` (4.13) `pb_thermohydraulique_qc` (4.30) `pb_thermohydraulique_turbulent_qc` (4.34) `pb_hydraulique_concentration` (4.17) `pb_hydraulique_concentration_turbulent` (4.19) `pb_thermohydraulique_concentration` (4.26) `pb_thermohydraulique_concentration_turbulent` (4.28) `pb_avec_passif` (4.11) `pb_post` (4.24) `problem_read_generic` (4.39) `pb_conduction_milieu_variable` (4.14) `pb_phase_field` (4.23) `modele_rayo_semi_transp` (4.9)

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* (4.2): One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3): List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4): This

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

## 4.2 corps\_postraitement

Description: not\_set

See also: post\_processing (4.4.3)

Usage:

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

where

- **definition\_champs** *definition\_champs* (4.2.1) for inheritance: Keyword to create new or more complex field for advanced postprocessing.
- **Probes|sondes** *sondes* (4.2.3) for inheritance: Probe.
- **domaine** *str* for inheritance: This optional parameter specifies the domain on which the data should be interpolated before it is written in the output file. The default is to write the data on the domain of the current problem (no interpolation).
- **format** *str* into ['lml', 'lata', 'lata\_v1', 'lata\_v2', 'med', 'med\_major'] for inheritance: This optional parameter specifies the format of the output file. The basename used for the output file is the basename of the data file. For the fmt parameter, choices are lml or lata. A short description of each format can be found below. The default value is lml.
- **fields|champs** *champs\_posts* (4.2.17) for inheritance: Field's write mode.
- **statistiques** *stats\_posts* (4.2.20) for inheritance: Statistics between two points fixed : start of integration time and end of integration time.

- **fichier** *str* for inheritance: Name of file.
- **statistiques\_en\_serie** *stats\_serie\_posts* (4.2.28) for inheritance: Statistics between two points not fixed : on period of integration.
- **interfaces** *champs\_posts* (4.2.17) for inheritance: Keyword to read all the characteristics of the interfaces. Different kind of interfaces exist as well as different interface initialisations.

#### 4.2.1 definition\_champs

Description: List of definition champ

See also: listobj (34.3)

Usage:

{ object1 object2 .... }

list of *definition\_champ* (4.2.2)

#### 4.2.2 definition\_champ

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

See also: objet\_lecture (35)

Usage:

**name** **champ\_generique**

where

- **name** *str*: The name of the new created field.
- **champ\_generique** *champ\_generique\_base* (8)

#### 4.2.3 sondes

Description: List of probes.

See also: listobj (34.3)

Usage:

{ object1 object2 .... }

list of *sonde* (4.2.4)

#### 4.2.4 sonde

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

See also: objet\_lecture (35)

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* (4.2.5): Type of probe.

#### 4.2.5 sonde\_base

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

See also: objet\_lecture (35) points (4.2.6) numero\_elem\_sur\_maitre (4.2.10) position\_like (4.2.11) segment (4.2.12) plan (4.2.13) volume (4.2.14) circle (4.2.15) circle\_3 (4.2.16)

Usage:

**sonde\_base**

#### 4.2.6 points

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

See also: sonde\_base (4.2.5) point (4.2.8) segmentpoints (4.2.9)

Usage:

**points points**

where

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

#### 4.2.7 listpoints

Description: Points.

See also: listobj (34.3)

Usage:

n object1 object2 ....

list of *un\_point* (3.12.3)

#### 4.2.8 point

Description: Point as class-daughter of Points.

See also: points (4.2.6)

Usage:

**point points**

where

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

#### 4.2.9 segmentpoints

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

See also: `points` (4.2.6)

Usage:

**segmentpoints points**

where

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

#### 4.2.10 numero\_elem\_sur\_maitre

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

See also: `sonde_base` (4.2.5)

Usage:

**numero\_elem\_sur\_maitre numero**

where

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

#### 4.2.11 position\_like

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

See also: `sonde_base` (4.2.5)

Usage:

**position\_like autre\_sonde**

where

- **autre\_sonde** *str*: Name of the other probe.

#### 4.2.12 segment

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

See also: `sonde_base` (4.2.5)

Usage:

**segment nbr point\_deb point\_fin**

where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point\_deb** *un\_point* (3.12.3): First outer probe segment point.
- **point\_fin** *un\_point* (3.12.3): Second outer probe segment point.

#### 4.2.13 plan

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

See also: sonde\_base ([4.2.5](#))

Usage:

**plan nbr nbr2 point\_deb point\_fin point\_fin\_2**

where

- **nbr** *int*: Number of probes in the first direction.
- **nbr2** *int*: Number of probes in the second direction.
- **point\_deb** *un\_point* ([3.12.3](#)): First point defining the angle. This angle should be positive.
- **point\_fin** *un\_point* ([3.12.3](#)): Second point defining the angle. This angle should be positive.
- **point\_fin\_2** *un\_point* ([3.12.3](#)): Third point defining the angle. This angle should be positive.

#### 4.2.14 volume

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

See also: sonde\_base ([4.2.5](#))

Usage:

**volume nbr nbr2 nbr3 point\_deb point\_fin point\_fin\_2 point\_fin\_3**

where

- **nbr** *int*: Number of probes in the first direction.
- **nbr2** *int*: Number of probes in the second direction.
- **nbr3** *int*: Number of probes in the third direction.
- **point\_deb** *un\_point* ([3.12.3](#)): Point of origin.
- **point\_fin** *un\_point* ([3.12.3](#)): Point defining the first direction (from point of origin).
- **point\_fin\_2** *un\_point* ([3.12.3](#)): Point defining the second direction (from point of origin).
- **point\_fin\_3** *un\_point* ([3.12.3](#)): Point defining the third direction (from point of origin).

#### 4.2.15 circle

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

See also: sonde\_base ([4.2.5](#))

Usage:

**circle nbr point\_deb [ direction ] radius theta1 theta2**

where

- **nbr** *int*: Number of probes between theta1 and theta2 (angles given in degrees).
- **point\_deb** *un\_point* ([3.12.3](#)): Center of the circle.
- **direction** *int into [0, 1, 2]*: Axis normal to the circle plane (0:x axis, 1:y axis, 2:z axis).
- **radius** *float*: Radius of the circle.
- **theta1** *float*: First angle.
- **theta2** *float*: Second angle.

#### 4.2.16 circle\_3

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

See also: `sonde_base` ([4.2.5](#))

Usage:

**circle\_3** *nbr* **point\_deb** **direction** **radius** **theta1** **theta2**

where

- **nbr** *int*: Number of probes between `teta1` and `teta2` (angles given in degrees).
- **point\_deb** *un\_point* ([3.12.3](#)): Center of the circle.
- **direction** *int into [0, 1, 2]*: Axis normal to the circle plane (0:x axis, 1:y axis, 2:z axis).
- **radius** *float*: Radius of the circle.
- **theta1** *float*: First angle.
- **theta2** *float*: Second angle.

#### 4.2.17 champs\_posts

Description: Field's write mode.

See also: `objet_lecture` ([35](#))

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* ([4.2.18](#)): Post-processed fields.

#### 4.2.18 champs\_a\_post

Description: Fields to be post-processed.

See also: `listobj` ([34.3](#))

Usage:

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

list of *champ\_a\_post* ([4.2.19](#))

#### 4.2.19 champ\_a\_post

Description: Field to be post-processed.

See also: `objet_lecture` ([35](#))

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.

#### 4.2.20 stats\_posts

Description: Field's write mode.

**Dt\_post**: This keyword is used to set the calculated statistics write period.

*dts*: frequency value.

**t\_deb** value: Start of integration time

**t\_fin** value: End of integration time

*stat*: Set to **Moyenne (average)** to calculate the average of the field *nom\_champ* (field name) over time or **Ecart\_type (std deviation)** to calculate the standard deviation (statistic rms) of the field *nom\_champ* (*field\_name*) or **Correlation** to calculate the correlation between the two fields *nom\_champ* and *second\_nom\_champ*.

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

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

Example:

```
Statistiques Dt_post dtst {
    t_deb 0.1 t_fin 0.12
Moyenne Pression
Ecart_type Pression
Correlation Vitesse Vitesse }
```

It will write every **dt\_post** the mean, standard deviation and correlation value:

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

See also: objet\_lecture (35)

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* (4.2.21): Post-processed fields.



#### 4.2.21 list\_stat\_post

Description: Post-processing for statistics

See also: listobj ([34.3](#))

Usage:

{ object1 object2 .... }

list of *stat\_post\_deriv* ([4.2.22](#))

#### 4.2.22 stat\_post\_deriv

Description: not\_set

See also: objet\_lecture ([35](#)) t\_deb ([4.2.23](#)) t\_fin ([4.2.24](#)) moyenne ([4.2.25](#)) ecart\_type ([4.2.26](#)) correlation ([4.2.27](#))

Usage:

**stat\_post\_deriv**

#### 4.2.23 t\_deb

Description: not\_set

See also: stat\_post\_deriv ([4.2.22](#))

Usage:

**t\_deb val**

where

- **val** *float*

#### 4.2.24 t\_fin

Description: not\_set

See also: stat\_post\_deriv ([4.2.22](#))

Usage:

**t\_fin val**

where

- **val** *float*

#### 4.2.25 moyenne

Synonymous: **champ\_post\_statistiques\_moyenne**

Description: not\_set

See also: stat\_post\_deriv ([4.2.22](#))

Usage:

**moyenne field [ localisation ]**

where

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

#### 4.2.26 ecart\_type

Synonymous: **champ\_post\_statistiques\_ecart\_type**

Description: not\_set

See also: stat\_post\_deriv ([4.2.22](#))

Usage:

**ecart\_type** **field** [**localisation**]

where

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

#### 4.2.27 correlation

Synonymous: **champ\_post\_statistiques\_correlation**

Description: not\_set

See also: stat\_post\_deriv ([4.2.22](#))

Usage:

**correlation** **first\_field** **second\_field** [**localisation**]

where

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

#### 4.2.28 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 (velocity)**, **Pression (pressure)**, **Temperature**, **Concentration**,...

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

*Example*:

**Statistiques\_en\_serie Dt\_integr dtst {**

**Moyenne Pression**

}

Will calculate and write every `dtst` seconds the mean value:

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

See also: `objet_lecture` (35)

Usage:

**mot dt\_integr stat**

where

- **mot** *str* into [*'dt\_integr'*]: Keyword is used to set the statistics period of integration and write period.
- **dt\_integr** *float*: Average on `dt_integr` time interval is post-processed every `dt_integr` seconds.
- **stat** *list\_stat\_post* (4.2.21)

### 4.3 post\_processings

Synonymous: **postraitements**

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

See also: `listobj` (34.3)

Usage:

{ `object1 object2 ....` }

list of *un\_postraitement* (4.3.1)

#### 4.3.1 un\_postraitement

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

See also: `objet_lecture` (35)

Usage:

**nom post**

where

- **nom** *str*: Name of the post-processing.
- **post** *corps\_postraitement* (4.2): Definition of the post-processing.

### 4.4 liste\_post\_ok

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

See also: `listobj` (34.3)

Usage:

{ `object1 object2 ....` }

list of *nom\_postraitement* (4.4.1)

#### 4.4.1 nom\_postraitement

Description:

See also: `objet_lecture` (35)

Usage:

**nom post**

where

- **nom** *str*: Name of the post-processing.
- **post** *postraitement\_base* (4.4.2): the post

#### 4.4.2 postraitement\_base

Description: `not_set`

See also: `objet_lecture` (35) `post_processing` (4.4.3) `postraitement_ft_lata` (4.4.4)

Usage:

#### 4.4.3 post\_processing

Synonymous: **postraitement**

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

See also: `postraitement_base` (4.4.2) `corps_postraitement` (4.2)

Usage:

**post\_processing** {

```
[ definition_champs definition_champs]  
[ Probeslsondes sondes]  
[ domaine str]  
[ format str into ['lml', 'lata', 'lata_v1', 'lata_v2', 'med', 'med_major']]  
[ fieldslchamps champs_posts]  
[ statistiques stats_posts]  
[ fichier str]  
[ statistiques_en_serie stats_serie_posts]  
[ interfaces champs_posts]
```

}

where

- **definition\_champs** *definition\_champs* (4.2.1): Keyword to create new or more complex field for advanced postprocessing.
- **Probeslsondes** *sondes* (4.2.3): Probe.
- **domaine** *str*: This optional parameter specifies the domain on which the data should be interpolated before it is written in the output file. The default is to write the data on the domain of the current problem (no interpolation).
- **format** *str* into ['lml', 'lata', 'lata\_v1', 'lata\_v2', 'med', 'med\_major']: This optional parameter specifies the format of the output file. The basename used for the output file is the basename of the data file. For the `fmt` parameter, choices are `lml` or `lata`. A short description of each format can be found below. The default value is `lml`.

- **fieldslchamps** *champs\_posts* (4.2.17): Field's write mode.
- **statistiques** *stats\_posts* (4.2.20): Statistics between two points fixed : start of integration time and end of integration time.
- **fichier** *str*: Name of file.
- **statistiques\_en\_serie** *stats\_serie\_posts* (4.2.28): Statistics between two points not fixed : on period of integration.
- **interfaces** *champs\_posts* (4.2.17): Keyword to read all the characteristics of the interfaces. Different kind of interfaces exist as well as different interface initialisations.

#### 4.4.4 postraitement\_ft\_lata

Description: not\_set

See also: postraitement\_base (4.4.2)

Usage:

**postraitement\_ft\_lata** **bloc**

where

- **bloc** *str*

### 4.5 liste\_post

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

See also: listobj (34.3)

Usage:

{ object1 object2 .... }

list of *un\_postraitement\_spec* (4.5.1)

#### 4.5.1 un\_postraitement\_spec

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

See also: objet\_lecture (35)

Usage:

[ **type\_un\_post** ] [ **type\_postraitement\_ft\_lata** ]

where

- **type\_un\_post** *type\_un\_post* (4.5.2)
- **type\_postraitement\_ft\_lata** *type\_postraitement\_ft\_lata* (4.5.3)

#### 4.5.2 type\_un\_post

Description: not\_set

See also: objet\_lecture (35)

Usage:

**type\_post**

where

- **type** *str* into ['postraitement', 'post\_processing']
- **post** *un\_postraitement* (4.3.1)

#### 4.5.3 type\_postraitement\_ft\_lata

Description: not\_set

See also: objet\_lecture (35)

Usage:

**type nom bloc**

where

- **type** *str* into ['postraitement\_ft\_lata', 'postraitement\_lata']
- **nom** *str*: Name of the post-processing.
- **bloc** *str*

#### 4.6 format\_file

Description: File formatted.

See also: objet\_lecture (35)

Usage:

[ **format** ] **name\_file**

where

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

#### 4.7 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 Associate keyword or with the Read/groupes keywords. The difference is that in the first case, the four problems exchange values then calculate their timestep, rather in the second case, the same strategy is used for all the problems listed inside one group, but the second group of problem exchange values with the first group of problems after the first group did its timestep. So, the first case may then also be written like this:

Probleme\_Couple pbc

Read pbc { groupes { { pb1 , pb2 , pb3 , pb4 } } }

There is a physical environment per problem (however, the same physical environment could be common to several problems).

Each problem is resolved in a domain.

Warning : Presently, coupling requires coincident meshes. In case of non-coincident meshes, boundary condition 'paroi\_contact' in VEF returns error message (see paroi\_contact for correcting procedure).

See also: pb\_gen\_base (4) pb\_couple\_rayonnement (4.40) pb\_couple\_rayo\_semi\_transp (4.15)

Usage:

**probleme\_couple** obj Lire obj {

[ **groupes** *list\_list\_nom*]

}  
where

- **groupes** *list\_list\_nom* (4.8): { groupes { { pb1 , pb2 } , { pb3 , pb4 } } }

## 4.8 list\_list\_nom

Description: pour les groupes

See also: listobj (34.3)

Usage:

{ object1 , object2 .... }

list of *list\_un\_pb* (34.1) separated with ,

## 4.9 modele\_rayo\_semi\_transp

Description: Radiation model for semi transparent gas. The model should be associated to the coupling problem BEFORE the time scheme.

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

See also: Pb\_base (4.1)

Usage:

```
modele_rayo_semi_transp obj Lire obj {  
    [ eq_rayo_semi_transp eq_rayo_semi_transp]  
    [ Post_processing|postraitemment corps_postraitemment]  
    [ Post_processings|postraitemments post_processings]  
    [ liste_de_postraitemments liste_post_ok]  
    [ liste_postraitemments liste_post]  
    [ sauvegarde format_file]  
    [ sauvegarde_simple format_file]  
    [ reprise format_file]  
    [ resume_last_time format_file]  
}
```

where

- **eq\_rayo\_semi\_transp** *eq\_rayo\_semi\_transp* (4.10): Irradiancy G equation. Radiative flux equals  $-\text{grad}(G)/3/\kappa$ .
- **Post\_processing|postraitemment** *corps\_postraitemment* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitemments** *post\_processings* (4.3) for inheritance: List of Postraitemment objects (with name).
- **liste\_de\_postraitemments** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitemments** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.

- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the *name\_file* file (see the class *format\_file*). If *format\_reprise* is xyz, the *name\_file* file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N \leq P$ ) processors. Should the calculation be resumed, values for the *tinit* (see *schema\_temps\_base*) time fields are taken from the *name\_file* file. If there is no backup corresponding to this time in the *name\_file*, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the *name\_file* file, resume the calculation at the last time found in the file (*tinit* is set to last time of saved files).

## 4.10 eq\_rayo\_semi\_transp

Description: Irradiancy equation.

See also: *objet\_lecture* (35)

Usage:

```
{
    solveur solveur_sys_base
    [ boundary_conditions|conditions_limites condlims ]
}
```

where

- **solveur** *solveur\_sys\_base* (10.12): Solver of the irradiancy equation.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1): Boundary conditions.

### 4.10.1 condlims

Description: Boundary conditions.

See also: *listobj* (34.3)

Usage:

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

list of *condlimlu* (4.10.2)

### 4.10.2 condlimlu

Description: Boundary condition specified.

See also: *objet\_lecture* (35)

Usage:

```
bord cl
```

where

- **bord** *str*: Name of the edge where the boundary condition applies.
- **cl** *condlim\_base* (12): Boundary condition at the boundary called *bord* (edge).



## 4.11 pb\_avec\_passif

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

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

See also: Pb\_base (4.1) pb\_thermohydraulique\_concentration\_turbulent\_scalaires\_passifs (4.29) pb\_thermohydraulique\_concentration\_scalaires\_passifs (4.27) pb\_thermohydraulique\_turbulent\_scalaires\_passifs (4.36) pb\_thermohydraulique\_scalaires\_passifs (4.32) pb\_hydraulique\_concentration\_turbulent\_scalaires\_passifs (4.20) pb\_hydraulique\_concentration\_scalaires\_passifs (4.18) pb\_thermohydraulique\_qc\_fraction\_massique (4.31) pb\_thermohydraulique\_turbulent\_qc\_fraction\_massique (4.35)

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* (4.12): Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction\_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N \leq P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.

- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.12 listeqn

Description: List of equations.

See also: listobj (34.3)

Usage:

{ object1 object2 .... }

list of *eqn\_base* (5.22)

## 4.13 pb\_conduction

Description: Resolution of the heat equation.

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

See also: Pb\_base (4.1)

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* (5.1): Heat equation.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.

- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < > P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

#### 4.14 pb\_conduction\_milieu\_variable

Description: Resolution of the heat equation.

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

See also: Pb\_base (4.1)

Usage:

```
pb_conduction_milieu_variable obj Lire obj {
    [ conduction_milieu_variable conduction_milieu_variable]
    [ 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\_milieu\_variable** *conduction\_milieu\_variable* (5.7): Heat equation.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on

P processors, whereas the previous calculation has been run on N ( $N \neq P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.

- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.15 pb\_couple\_rayo\_semi\_transp

Description: Problem coupling several other problems to which radiation coupling is added (for semi transparent gas).

You have to associate a modele\_rayo\_semi\_transp

You have to add a radiative term source in energy equation

Warning: Calculation with semi transparent gas model may lead to divergence when high temperature differences are used. Indeed, the calculation of the stability time step of the equation does not take in account the source term. In semi transparent gas model, energy equation source term depends strongly of temperature via irradiance and stability is not guaranteed by the calculated time step. Reducing the facsec of the time scheme is a good tip to reach convergence when divergence is encountered.

See also: probleme\_couple (4.7)

Usage:

**pb\_couple\_rayo\_semi\_transp** obj Lire obj {

[ **groupes** *list\_list\_nom*]

}

where

- **groupes** *list\_list\_nom* (4.8) for inheritance: { groupes { { pb1 , pb2 } , { pb3 , pb4 } } }

## 4.16 pb\_hydraulique

Description: Resolution of the Navier-Stokes equations.

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

See also: Pb\_base (4.1)

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* (5.31): Navier-Stokes equations.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.17 pb\_hydraulique\_concentration

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

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

See also: Pb\_base (4.1)

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* (5.31): Navier-Stokes equations.

- **convection\_diffusion\_concentration** *convection\_diffusion\_concentration* (5.11): Constituent transport vectorial equation (concentration diffusion convection).
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

#### 4.18 pb\_hydraulique\_concentration\_scalaires\_passifs

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

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

See also: pb\_avec\_passif (4.11)

Usage:

```
pb_hydraulique_concentration_scalaires_passifs 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* (5.31): Navier-Stokes equations.
- **convection\_diffusion\_concentration** *convection\_diffusion\_concentration* (5.11): Constituent transport equations (concentration diffusion convection).
- **equations\_scalaires\_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processing|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.19 pb\_hydraulique\_concentration\_turbulent

Description: Resolution of Navier-Stokes/multiple constituent transport equations, with turbulence modelling.

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

See also: Pb\_base (4.1)

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_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\_turbulent** *navier\_stokes\_turbulent* (5.32): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection\_diffusion\_concentration\_turbulent** *convection\_diffusion\_concentration\_turbulent* (5.13): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.20 pb\_hydraulique\_concentration\_turbulent\_scalaires\_passifs

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

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

See also: pb\_avec\_passif (4.11)

Usage:

```

pb_hydraulique_concentration_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* (5.32): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection\_diffusion\_concentration\_turbulent** *convection\_diffusion\_concentration\_turbulent* (5.13): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **equations\_scalaires\_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.21 pb\_hydraulique\_turbulent

Description: Resolution of Navier-Stokes equations with turbulence modelling.

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

See also: `Pb_base` (4.1)

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* (5.32): Navier-Stokes equations as well as the associated turbulence model equations.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.22 pb\_mg

Description: Multi-grid problem.

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

See also: `pb_gen_base` (4)

Usage:

**pb\_mg**

## 4.23 pb\_phase\_field

Description: Problem to solve local instantaneous incompressible-two-phase-flows. Complete description of the Phase Field model for incompressible and immiscible fluids can be found into this PDF: TRUST-  
\_ROOT/doc/TRUST/phase\_field\_non\_miscible\_manuel.pdf

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

See also: Pb\_base (4.1)

Usage:

**pb\_phase\_field** obj Lire obj {

```
[ navier_stokes_phase_field navier_stokes_phase_field]  
[ convection_diffusion_phase_field convection_diffusion_phase_field]  
[ 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\_phase\_field** *navier\_stokes\_phase\_field* (5.29): Navier Stokes equation for the Phase Field problem.
- **convection\_diffusion\_phase\_field** *convection\_diffusion\_phase\_field* (5.16): Cahn-Hilliard equation of the Phase Field problem. The unknown of this equation is the concentration C.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.

- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.24 pb\_post

Description: not\_set

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

See also: Pb\_base (4.1)

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

## 4.25 pb\_thermohydraulique

Description: Resolution of thermohydraulic problem.

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

See also: Pb\_base (4.1)

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* (5.31): Navier-Stokes equations.
- **convection\_diffusion\_temperature** *convection\_diffusion\_temperature* (5.17): Energy equation (temperature diffusion convection).
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.26 pb\_thermohydraulique\_concentration

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

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

See also: Pb\_base (4.1)

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* (5.31): Navier-Stokes equations.
- **convection\_diffusion\_concentration** *convection\_diffusion\_concentration* (5.11): Constituent transport equations (concentration diffusion convection).
- **convection\_diffusion\_temperature** *convection\_diffusion\_temperature* (5.17): Energy equation (temperature diffusion convection).
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.27 pb\_thermohydraulique\_concentration\_scalaires\_passifs

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

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

See also: pb\_avec\_passif (4.11)

Usage:

```
pb_thermohydraulique_concentration_scalaires_passifs 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* (5.31): Navier-Stokes equations.
- **convection\_diffusion\_concentration** *convection\_diffusion\_concentration* (5.11): Constituent transport equations (concentration diffusion convection).
- **convection\_diffusion\_temperature** *convection\_diffusion\_temperature* (5.17): Energy equations (temperature diffusion convection).
- **equations\_scalaires\_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction\_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* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file



created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.

- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.28 pb\_thermohydraulique\_concentration\_turbulent

Description: Resolution of Navier-Stokes/energy/multiple constituent transport equations, with turbulence modelling.

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

See also: Pb\_base (4.1)

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* (5.32): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection\_diffusion\_concentration\_turbulent** *convection\_diffusion\_concentration\_turbulent* (5.13): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection\_diffusion\_temperature\_turbulent** *convection\_diffusion\_temperature\_turbulent* (5.21): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.



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

## 4.29 pb\_thermohydraulique\_concentration\_turbulent\_scalaires\_passifs

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

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

See also: pb\_avec\_passif (4.11)

Usage:

```
pb_thermohydraulique_concentration_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* (5.32): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection\_diffusion\_concentration\_turbulent** *convection\_diffusion\_concentration\_turbulent* (5.13): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection\_diffusion\_temperature\_turbulent** *convection\_diffusion\_temperature\_turbulent* (5.21): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations\_scalaires\_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This

kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.

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

### 4.30 pb\_thermohydraulique\_qc

Description: Resolution of thermohydraulic problem under low 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 Discretize should have already been used to read the object.

See also: Pb\_base (4.1)

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* (5.30): Navier-Stokes equations under low Mach number.
- **convection\_diffusion\_chaleur\_qc** *convection\_diffusion\_chaleur\_qc* (5.9): Energy equation under low Mach number.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

### 4.31 pb\_thermohydraulique\_qc\_fraction\_massique

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

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

See also: pb\_avec\_passif (4.11)

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* (5.30): Navier-Stokes equations under low Mach number.
- **convection\_diffusion\_chaleur\_qc** *convection\_diffusion\_chaleur\_qc* (5.9): Energy equation under low Mach number.
- **equations\_scalaires\_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

### 4.32 pb\_thermohydraulique\_scalaires\_passifs

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

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

See also: pb\_avec\_passif (4.11)

Usage:

```

pb_thermohydraulique_scalaires_passifs 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* (5.31): Navier-Stokes equations.
- **convection\_diffusion\_temperature** *convection\_diffusion\_temperature* (5.17): Energy equations (temperature diffusion convection).
- **equations\_scalaires\_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

### 4.33 pb\_thermohydraulique\_turbulent

Description: Resolution of thermohydraulic problem, with turbulence modelling.

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

See also: `Pb_base` (4.1)

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* (5.32): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection\_diffusion\_temperature\_turbulent** *convection\_diffusion\_temperature\_turbulent* (5.21): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

#### 4.34 pb\_thermohydraulique\_turbulent\_qc

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

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

Keyword Discretize should have already been used to read the object.  
See also: Pb\_base (4.1)

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* (5.33): Navier-Stokes equations under low Mach number as well as the associated turbulence model equations.
- **convection\_diffusion\_chaleur\_turbulent\_qc** *convection\_diffusion\_chaleur\_turbulent\_qc* (5.10): Energy equation under low Mach number as well as the associated turbulence model equations.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).



### 4.35 pb\_thermohydraulique\_turbulent\_qc\_fraction\_massique

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

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

See also: pb\_avec\_passif (4.11)

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* (5.33): Navier-Stokes equations under low Mach number as well as the associated turbulence model equations.
- **convection\_diffusion\_chaleur\_turbulent\_qc** *convection\_diffusion\_chaleur\_turbulent\_qc* (5.10): Energy equation under low Mach number as well as the associated turbulence model equations.
- **equations\_scalaires\_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction\_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format\_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde\_simple** *format\_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on



P processors, whereas the previous calculation has been run on N ( $N \leq P$ ) processors. Should the calculation be resumed, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.

- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to resume a calculation based on the name\_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

### 4.36 pb\_thermohydraulique\_turbulent\_scalaires\_passifs

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

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

See also: pb\_avec\_passif (4.11)

Usage:

```
pb_thermohydraulique_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* (5.32): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection\_diffusion\_temperature\_turbulent** *convection\_diffusion\_temperature\_turbulent* (5.21): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations\_scalaires\_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post\_processing|postraitement** *corps\_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.

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

### 4.37 pbc\_med

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

See also: *pb\_gen\_base* (4)

Usage:

**pbc\_med list\_info\_med**

where

- **list\_info\_med** *list\_info\_med* (4.38)

### 4.38 list\_info\_med

Description: *not\_set*

See also: *listobj* (34.3)

Usage:

{ *object1* , *object2* .... }

list of *info\_med* (4.38.1) separated with ,

#### 4.38.1 info\_med

Description: *not\_set*

See also: *objet\_lecture* (35)

Usage:

**file\_med domaine pb\_post**

where

- **file\_med** *str*: Name of the MED file.
- **domaine** *str*: Name of domain.
- **pb\_post** *pb\_post* (4.24)

### 4.39 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 Associate keyword.

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

See also: `Pb_base` (4.1) `probleme_ft_disc_gen` (4.41)

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

### 4.40 pb\_couple\_rayonnement

Description: This keyword is used to define a problem coupling several other problems to which radiation coupling is added.

See also: `probleme_couple` (4.7)

Usage:

```
pb_couple_rayonnement obj Lire obj {  
    [ groupes list_list_nom ]  
}
```

where

- **groupes** *list\_list\_nom* (4.8) for inheritance: { groupes { { pb1 , pb2 } , { pb3 , pb4 } } }

#### 4.41 `probleme_ft_disc_gen`

Description: The generic Front-Tracking problem in the discontinuous version. It differs from the rest of the TRUST code : The problem does not state the number of equations that are enclosed in the problem. Two equations are compulsory : a momentum balance equation (alias Navier-Stokes equation) and an interface tracking equation. The list of equations to be solved is declared in the beginning of the data file. Another difference with more classical TRUST data file, lies in the fluids definition. The two-phase fluid (Fluide\_Diphasique) is made with two usual single-phase fluids (Fluide\_Incompressible). As the list of equations to be solved in the generic Front-Tracking 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 Discretize should have already been used to read the object.

See also: `problem_read_generic` (4.39)

Usage:

```
probleme_ft_disc_gen 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* (4.2) for inheritance: One post-processing (without name).
- **Post\_processings|postraitements** *post\_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste\_de\_postraitements** *liste\_post\_ok* (4.4) for inheritance: This
- **liste\_postraitements** *liste\_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.

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

## 5 mor\_eqn

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

See also: *objet\_u* (36) *eqn\_base* (5.22)

Usage:

### 5.1 conduction

Description: Heat equation.

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

See also: *eqn\_base* (5.22)

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* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)

- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.2 bloc\_diffusion

Description: not\_set

See also: objet\_lecture (35)

Usage:

**aco** [ **opérateur** ] [ **op\_implicite** ] **acof**

where

- **aco** *str* into [' ']: Opening curly bracket.
- **opérateur** *diffusion\_deriv* (5.2.1): if none is specified, the diffusive scheme used is a 2nd-order scheme.
- **op\_implicite** *op\_implicite* (5.2.9): To have diffusive implicitation, it use Uzawa algorithm. Very useful when viscosity has large variations.
- **acof** *str* into [' ']: Closing curly bracket.

### 5.2.1 diffusion\_deriv

Description: not\_set

See also: objet\_lecture (35) negligeable (5.2.2) p1b (5.2.3) p1ncp1b (5.2.4) stab (5.2.5) standard (5.2.6) option (5.2.8)

Usage:

**diffusion\_deriv**

### 5.2.2 negligeable

Description: the diffusivity will not taken in count

See also: diffusion\_deriv (5.2.1)

Usage:  
**negligeable**

### 5.2.3 p1b

Description: not\_set

See also: diffusion\_deriv (5.2.1)

Usage:  
**p1b**

### 5.2.4 p1ncp1b

Description: not\_set

See also: diffusion\_deriv (5.2.1)

Usage:

### 5.2.5 stab

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

See also: diffusion\_deriv (5.2.1)

Usage:

```
stab {  
    [ standard int]  
    [ info int]  
    [ new_jacobian int]  
    [ nu int]  
    [ nut int]  
    [ nu_transp int]  
    [ nut_transp int]  
}
```

where

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

### 5.2.6 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` ([5.2.1](#))

Usage:

**standard** [ **mot1** ] [ **bloc\_diffusion\_standard** ]

where

- **mot1** *str* into [*'defaut\_bar'*]: equivalent to `grad_Ubar 1 nu 1 nut 1 nu_transp 1 nut_transp 1 filtrer_resu 1`
- **bloc\_diffusion\_standard** *bloc\_diffusion\_standard* ([5.2.7](#))

### 5.2.7 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` ([35](#))

Usage:

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

where

- **mot1** *str* into [*'grad\_Ubar', 'nu', 'nut', 'nu\_transp', 'nut\_transp', 'filtrer\_resu'*]
- **val1** *int* into [*0, 1*]
- **mot2** *str* into [*'grad\_Ubar', 'nu', 'nut', 'nu\_transp', 'nut\_transp', 'filtrer\_resu'*]
- **val2** *int* into [*0, 1*]
- **mot3** *str* into [*'grad\_Ubar', 'nu', 'nut', 'nu\_transp', 'nut\_transp', 'filtrer\_resu'*]
- **val3** *int* into [*0, 1*]
- **mot4** *str* into [*'grad\_Ubar', 'nu', 'nut', 'nu\_transp', 'nut\_transp', 'filtrer\_resu'*]
- **val4** *int* into [*0, 1*]
- **mot5** *str* into [*'grad\_Ubar', 'nu', 'nut', 'nu\_transp', 'nut\_transp', 'filtrer\_resu'*]
- **val5** *int* into [*0, 1*]
- **mot6** *str* into [*'grad\_Ubar', 'nu', 'nut', 'nu\_transp', 'nut\_transp', 'filtrer\_resu'*]
- **val6** *int* into [*0, 1*]

### 5.2.8 option

Description: `not_set`

See also: `diffusion_deriv` ([5.2.1](#))

Usage:



## **option bloc\_lecture**

where

- **bloc\_lecture** *bloc\_lecture* ([3.44](#))

### **5.2.9 op\_implicite**

Description: not\_set

See also: *objet\_lecture* ([35](#))

Usage:

## **implicite mot solveur**

where

- **implicite** *str* into [*'implicite'*]
- **mot** *str* into [*'solveur'*]
- **solveur** *solveur\_sys\_base* ([10.12](#))

## **5.3 condinits**

Description: Initial conditions.

See also: *objet\_lecture* ([35](#))

Usage:

## **aco condinit acof**

where

- **aco** *str* into [*'{'*]: Opening curly bracket.
- **condinit** *condinit* ([5.3.1](#)): *CI*
- **acof** *str* into [*'}'*]: Closing curly bracket.

### **5.3.1 condinit**

Description: Initial condition.

See also: *objet\_lecture* ([35](#))

Usage:

## **nom ch**

where

- **nom** *str*: Name of initial condition field.
- **ch** *champ\_base* ([16.1](#)): Type field and the initial values.

## **5.4 sources**

Description: The sources.

See also: *listobj* ([34.3](#))

Usage:

{ *object1* , *object2* .... }

list of *source\_base* ([30](#)) separeted with ,

## 5.5 ecrire\_fichier\_xyz\_valeur\_param

Description: not\_set

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

See also: listobj ([34.3](#))

Usage:

n object1 , object2 ....

list of *ecrire\_fichier\_xyz\_valeur\_item* ([5.5.1](#)) separated with ,

### 5.5.1 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 ([35](#))

Usage:

**name dt\_ecrire\_fic [ bords ]**

where

- **name** *str*: Name of the field to write (*Champ\_Inc*, *Champ\_Fonc* or a post\_processed field).
- **dt\_ecrire\_fic** *float*: Time period for printing in the file.
- **bords** *bords\_ecrire* ([5.5.2](#)): to post-process only on some boundaries

### 5.5.2 bords\_ecrire

Description: not\_set

See also: objet\_lecture ([35](#))

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.

## 5.6 parametre\_equation\_base

Description: Basic class for parametre\_equation

See also: objet\_lecture ([35](#)) parametre\_diffusion\_implicite ([5.6.1](#)) parametre\_implicite ([5.6.2](#))

Usage:

### 5.6.1 parametre\_diffusion\_implicite

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

See also: `parametre_equation_base` (5.6)

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.

### 5.6.2 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` (5.6)

Usage:

```
parametre_implicite {  
    [ seuil_convergence_implicite float ]  
    [ seuil_convergence_solveur float ]  
    [ solveur solveur_sys_base ]  
    [ resolution_explicite ]  
    [ equation_non_resolue ]  
    [ equation_frequence_resolue str ]  
}  
where
```

- **seuil\_convergence\_implicite** *float*: Keyword to change for this equation only the value of `seuil_convergence_implicite` used in the implicit scheme.
- **seuil\_convergence\_solveur** *float*: Keyword to change for this equation only the value of `seuil_convergence_solveur` used in the implicit scheme
- **solveur** *solveur\_sys\_base* (10.12): Keyword to change for this equation only the solver used in the implicit scheme
- **resolution\_explicite** : To solve explicitly the equation whereas the scheme is an implicit scheme.

- **equation\_non\_resolue** : Keyword to specify that the equation is not solved.
- **equation\_frequence\_resolue** *str*: Keyword to specify that the equation is solved only every n time steps (n is an integer or given by a time-dependent function f(t)).

## 5.7 conduction\_milieu\_variable

Description: Heat equation.

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

See also: eqn\_base (5.22)

Usage:

```
conduction_milieu_variable 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* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) 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 equations are not solved between time t0 and t1.

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

## 5.8 bloc\_convection

Description: not\_set

See also: objet\_lecture (35)

Usage:

**aco operateur acof**

where

- **aco** *str* into [*'*]: Opening curly bracket.
- **operateur** *convection\_deriv* (5.8.1)
- **acof** *str* into [*'*]: Closing curly bracket.

### 5.8.1 convection\_deriv

Description: not\_set

See also: objet\_lecture (35) [amont](#) (5.8.2) [amont\\_old](#) (5.8.3) [centre](#) (5.8.4) [centre4](#) (5.8.5) [centre\\_old](#) (5.8.6) [di\\_l2](#) (5.8.7) [ef](#) (5.8.8) [muscl3](#) (5.8.10) [ef\\_stab](#) (5.8.11) [generic](#) (5.8.14) [kquick](#) (5.8.15) [muscl](#) (5.8.16) [muscl\\_old](#) (5.8.17) [muscl\\_new](#) (5.8.18) [negligeable](#) (5.8.19) [quick](#) (5.8.20) [btd](#) (5.8.21) [supg](#) (5.8.22) [ale](#) (5.8.23)

Usage:

**convection\_deriv**

### 5.8.2 amont

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

See also: [convection\\_deriv](#) (5.8.1)

Usage:

**amont**

### 5.8.3 amont\_old

Description: Only for VEF discretization, obsolete keyword, see [amont](#).

See also: [convection\\_deriv](#) (5.8.1)

Usage:

**amont\_old**

### 5.8.4 centre

Description: For VDF and VEF discretizations.

See also: [convection\\_deriv](#) (5.8.1)

Usage:  
**centre**

#### 5.8.5 centre4

Description: For VDF and VEF discretizations.

See also: convection\_deriv ([5.8.1](#))

Usage:  
**centre4**

#### 5.8.6 centre\_old

Description: Only for VEF discretization.

See also: convection\_deriv ([5.8.1](#))

Usage:  
**centre\_old**

#### 5.8.7 di\_l2

Description: Only for VEF discretization.

See also: convection\_deriv ([5.8.1](#))

Usage:  
**di\_l2**

#### 5.8.8 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 :  $\text{div}((u \cdot \text{grad } u_b, v_b) - (u \cdot \text{grad } v_b, u_b))$ , where  $v_b$  corresponds to the filtered reference test functions.

Remark:

This class requires to define a filtering operator : see solveur\_bar

See also: convection\_deriv ([5.8.1](#))

Usage:  
**ef [ mot1 ] [ bloc\_ef ]**  
where

- **mot1** str into [*default\_bar*]: equivalent to transportant\_bar 0 transporte\_bar 1 filtrer\_resu 1 antisym

- **bloc\_ef** *bloc\_ef* (5.8.9)

### 5.8.9 bloc\_ef

Description: not\_set

See also: objet\_lecture (35)

Usage:

**mot1 val1 mot2 val2 mot3 val3 mot4 val4**

where

- **mot1** *str* into ['transportant\_bar', 'transporte\_bar', 'filtrer\_resu', 'antisym']
- **val1** *int* into [0, 1]
- **mot2** *str* into ['transportant\_bar', 'transporte\_bar', 'filtrer\_resu', 'antisym']
- **val2** *int* into [0, 1]
- **mot3** *str* into ['transportant\_bar', 'transporte\_bar', 'filtrer\_resu', 'antisym']
- **val3** *int* into [0, 1]
- **mot4** *str* into ['transportant\_bar', 'transporte\_bar', 'filtrer\_resu', 'antisym']
- **val4** *int* into [0, 1]

### 5.8.10 muscl3

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

See also: convection\_deriv (5.8.1)

Usage:

**muscl3** {

    [ **alpha** *float*]

}

where

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

### 5.8.11 ef\_stab

Description: Keyword for a VEF convective scheme.

See also: convection\_deriv (5.8.1)

Usage:

**ef\_stab** {

    [ **alpha** *float*]

    [ **test** *int*]

    [ **tdivu** ]

    [ **old** ]

    [ **volumes\_etendus** ]

    [ **volumes\_non\_etendus** ]

    [ **amont\_sous\_zone** *str*]

```
[ alpha_sous_zone listsous_zone_valeur ]
}
```

where

- **alpha** *float*: To weight the scheme centering with the factor double (between 0 (full centered) and 1 (mix between upwind and centered), by default 1). For scalar equation, it is 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  $\text{div}(\text{TU}) - \text{TdivU} (= \text{UgradT})$ .
- **old** : To use old version of EF\_stab scheme (default no).
- **volumes\_etendus** : Option for the scheme to use the extended volumes (default, yes).
- **volumes\_non\_etendus** : Option for the scheme to not use the extended volumes (default, no).
- **amont\_sous\_zone** *str*: Option to degenerate EF\_stab scheme into Amont (upwind) scheme in the sub zone of name *sz\_name*. The sub zone may be located arbitrarily in the domain but the more often this option will be activated in a zone where EF\_stab scheme generates instabilities as for free outlet for example.
- **alpha\_sous\_zone** *listsous\_zone\_valeur* (5.8.12): Option to change locally the alpha value on N sub-zones named *sub\_zone\_name\_I*. Generally, it is used to prevent from a local divergence by increasing locally the alpha parameter.

#### 5.8.12 **listsous\_zone\_valeur**

Description: List of groups of two words.

See also: [listobj \(34.3\)](#)

Usage:

n object1 object2 ....

list of *sous\_zone\_valeur* (5.8.13)

#### 5.8.13 **sous\_zone\_valeur**

Description: Two words.

See also: [objet\\_lecture \(35\)](#)

Usage:

**sous\_zone valeur**

where

- **sous\_zone** *str*: sous zone
- **valeur** *float*: value

#### 5.8.14 **generic**

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

Examples:

```
convection { generic amont }
```

```
convection { generic muscl minmod 1 }
```



convection { generic muscl vanleer 2 }

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

See also: convection\_deriv ([5.8.1](#))

Usage:

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

where

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

#### 5.8.15 kquick

Description: Only for VEF discretization.

See also: convection\_deriv ([5.8.1](#))

Usage:

**kquick**

#### 5.8.16 muscl

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

See also: convection\_deriv ([5.8.1](#))

Usage:

**muscl**

#### 5.8.17 muscl\_old

Description: Only for VEF discretization.

See also: convection\_deriv ([5.8.1](#))

Usage:

**muscl\_old**

#### 5.8.18 muscl\_new

Description: Only for VEF discretization.

See also: convection\_deriv ([5.8.1](#))

Usage:

**muscl\_new**

### 5.8.19 **negligeable**

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

See also: `convection_deriv` ([5.8.1](#))

Usage:

**negligeable**

### 5.8.20 **quick**

Description: Only for VDF discretization.

See also: `convection_deriv` ([5.8.1](#))

Usage:

**quick**

### 5.8.21 **btd**

Description: Only for EF discretization.

See also: `convection_deriv` ([5.8.1](#))

Usage:

```
btd {  
    btd float  
    facteur float  
}
```

where

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

### 5.8.22 **supg**

Description: Only for EF discretization.

See also: `convection_deriv` ([5.8.1](#))

Usage:

```
supg {  
    facteur float  
}
```

where

- **facteur** *float*

### 5.8.23 ale

Description: a convective scheme for ALE method. Example: See the test case ALE\_membrane.

See also: convection\_deriv (5.8.1)

Usage:

**ale opconv**

where

- **opconv** *bloc\_convection* (5.8)

## 5.9 convection\_diffusion\_chaleur\_qc

Description: Energy equation under low Mach number.

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

See also: eqn\_base (5.22) convection\_diffusion\_chaleur\_turbulent\_qc (5.10)

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  
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* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.10 convection\_diffusion\_chaleur\_turbulent\_qc

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

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

See also: convection\_diffusion\_chaleur\_qc (5.9)

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* (24): 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):  $\rho u \cdot \text{grad} T = \text{div}(\rho u \cdot T) - T \text{div}(\rho u \cdot 1)$   
ancien:  $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \text{div}(u)$   
divuT\_moins\_Tdivu :  $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \text{div}(u \cdot 1)$
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.

- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.11 convection\_diffusion\_concentration

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

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

See also: eqn\_base (5.22) convection\_diffusion\_concentration\_turbulent (5.13) convection\_diffusion\_concentration\_ft\_disc (5.12) convection\_diffusion\_phase\_field (5.16)

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 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*
- **alias** *str*
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.12 convection\_diffusion\_concentration\_ft\_disc

Description: not\_set

Keyword Discretize should have already been used to read the object.  
See also: convection\_diffusion\_concentration (5.11)

Usage:

**convection\_diffusion\_concentration\_ft\_disc** obj Lire obj {

```
[ equation_interface  str]
phase  int into [0, 1]
[ option  str]
[ nom_inconnue  str]
[ masse_molaire  float]
[ alias  str]
[ convection  bloc_convection]
[ diffusion  bloc_diffusion]
[ initial_conditions|conditions_initiales  condinits]
```

```

[ boundary_conditions|conditions_limits 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

```

- **equation\_interface** *str*: this is the name of the interface tracking equation to watch. The scalar will not diffuse through the interface of this equation.
- **phase** *int* into  $[0, 1]$ : tells whether the scalar must be confined in phase 0 or in phase 1
- **option** *str*: Experimental features used to prevent the concentration to leak through the interface between phases due to numerical diffusion.  
RIEN: do nothing  
RAMASSE\_MIETTES\_SIMPLE: at each timestep, this algorithm takes all the mass located in the opposite phase and spreads it uniformly in the given phase.
- **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* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limit**s *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) 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 equations are not solved between time  $t_0$  and  $t_1$ .  

```

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

```

### 5.13 convection\_diffusion\_concentration\_turbulent

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

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

See also: convection\_diffusion\_concentration (5.11)

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* (24): Turbulence model to be used in the constituent transport 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 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* for inheritance
- **alias** *str* for inheritance
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.14 convection\_diffusion\_fraction\_massique\_qc

Description: not\_set

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

See also: eqn\_base (5.22)

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_limités 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* (15)
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limités** *condlims* (4.10.1) for inheritance: Boundary conditions.

- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)

- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.

Navier\_Sokes\_Standard

{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.15 convection\_diffusion\_fraction\_massique\_turbulent\_qc

Description: not\_set

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

See also: eqn\_base (5.22)

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* (24): Turbulence model to be used.
- **espece** *espece* (15)
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.

- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)

- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.16 convection\_diffusion\_phase\_field

Description: Cahn-Hilliard equation of the Phase Field problem. The unknown of this equation is the concentration C.

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

See also: convection\_diffusion\_concentration (5.11)

Usage:

**convection\_diffusion\_phase\_field** obj Lire obj {

```

    mu_1 float
    mu_2 float
    rho_1 float
    rho_2 float
    potentiel_chimique_generalise str into ['avec_energie_cinetique', 'sans_energie_cinetique']
    [ nom_inconnue str]
    [ masse_molaire float]
    [ alias str]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ initial_conditions|conditions_initiales condinits]
    [ boundary_conditions|conditions_limités 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

- **mu\_1** float: Dynamic viscosity of the first phase.
- **mu\_2** float: Dynamic viscosity of the second phase.
- **rho\_1** float: Density of the first phase.
- **rho\_2** float: Density of the second phase.
- **potentiel\_chimique\_generalise** *str into ['avec\_energie\_cinetique', 'sans\_energie\_cinetique']*: To define (chaîne set to avec\_energie\_cinetique) or not (chaîne set to sans\_energie\_cinetique) if the Cahn-Hilliard equation contains the kinetic energy term.

- **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* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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 : `pdbname_fieldname_[boundaryname]_time.dat`
- **ecrire\_fichier\_xyz\_valeur\_bin** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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 : `pdbname_fieldname_[boundaryname]_time.dat`
- **parametre\_equation** *parametre\_equation\_base* (5.6) 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 equations are not solved between time `t0` and `t1`.  
`Navier_Sokes_Standard`  
`{ equation_non_resolue (t>t0)*(t<t1) }`

## 5.17 convection\_diffusion\_temperature

Description: Energy equation (temperature diffusion convection).

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

See also: `eqn_base` (5.22) `convection_diffusion_temperature_ft_disc` (5.19)

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* (5.18): to activate or not (the default is Direct Forcing method) the Penalized Direct Forcing method to impose the specified temperature on the solid-fluid interface.
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limite** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) 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 equations are not solved between time t0 and t1.  
*Navier\_Sokes\_Standard*  
{ *equation\_non\_resolue* (t>t0)\*(t<t1) }

## 5.18 pp

Description: *not\_set*

See also: *listobj* (34.3)

Usage:

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

list of *penalisation\_l2\_ftd\_lec* (5.18.1)

### 5.18.1 penalisation\_l2\_ftd\_lec

Description: *not\_set*

See also: *objet\_lecture* (35)

Usage:

**bord** *val*

where

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

## 5.19 convection\_diffusion\_temperature\_ft\_disc

Description: not\_set

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

See also: convection\_diffusion\_temperature (5.17)

Usage:

**convection\_diffusion\_temperature\_ft\_disc** obj Lire obj {

```
[ equation_interface str]  
phase int into [0, 1]  
[ equation_navier_stokes str]  
[ stencil_width int]  
[ maintien_temperature objet_lecture_maintien_temperature]  
[ 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

- **equation\_interface** *str*: The name of the interface equation should be given.
- **phase** *int into [0, 1]*: Phase in which the temperature equation will be solved. The temperature, which may be postprocessed with the keyword temperature\_EquationName, in the other phase may be negative: the code only computes the temperature field in the specified phase. The other phase is supposed to physically stay at saturation temperature. The code uses a ghost fluid numerical method to work on a smooth temperature field at the interface. In the opposite phase (1-X) the temperature will therefore be extrapolated in the vicinity of the interface and have the opposite sign, saturation temperature is zero by convention).
- **equation\_navier\_stokes** *str*: The name of the Navier Stokes equation of the problem should be given.
- **stencil\_width** *int*: distance in mesh elements over which the temperature field should be extrapolated in the opposite phase.
- **maintien\_temperature** *objet\_lecture\_maintien\_temperature* (5.20): **maintien\_temperature SOUS\_ZONE\_NAME VALUE** : experimental, this acts as a dynamic source term that heats or cools the fluid to maintain the average temperature to VALUE within the specified region. At this time, this is done by multiplying the temperature within the SOUS\_ZONE by an appropriate uniform value at each timestep. This feature might be implemented in a separate source term in the future.

- **penalisation\_l2\_ftd** *pp* (5.18) for inheritance: to activate or not (the default is Direct Forcing method) the Penalized Direct Forcing method to impose the specified temperature on the solid-fluid interface.
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.20 objet\_lecture\_maintien\_temperature

Description: not\_set

See also: objet\_lecture (35)

Usage:

**sous\_zone temperature\_moyenne**

where

- **sous\_zone** *str*
- **temperature\_moyenne** *float*

## 5.21 convection\_diffusion\_temperature\_turbulent

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

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

See also: `eqn_base` (5.22)

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* (24): Turbulence model for the energy equation.
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) 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 equations are not solved between time `t0` and `t1`.  
`Navier_Sokes_Standard`  
`{ equation_non_resolue (t>t0)*(t<t1) }`

## 5.22 eqn\_base

Description: Basic class for equations.



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

See also: `mor_eqn` (5) `navier_stokes_standard` (5.31) `convection_diffusion_temperature` (5.17) `convection_diffusion_temperature_turbulent` (5.21) `conduction` (5.1) `convection_diffusion_chaleur_qc` (5.9) `transport_k_epsilon` (5.39) `convection_diffusion_concentration` (5.11) `convection_diffusion_fraction_massique_qc` (5.14) `convection_diffusion_fraction_massique_turbulent_qc` (5.15) `conduction_milieu_variable` (5.7) `transport_interfaces_ft_disc` (5.34) `transport_marqueur_ft` (5.40)

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* (5.8): Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2): Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3): Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1): Boundary conditions.
- **sources** *sources* (5.4): To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5): 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 : *pbname\_fieldname\_[boundaryname]\_time.dat*
- **ecrire\_fichier\_xyz\_valeur\_bin** *ecrire\_fichier\_xyz\_valeur\_param* (5.5): 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 : *pbname\_fieldname\_[boundaryname]\_time.dat*
- **parametre\_equation** *parametre\_equation\_base* (5.6): Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str*: The equation will not be solved while condition(t) is verified if `equation_non_resolue` keyword is used. Exemple: The Navier-Stokes equations are not solved between time  $t_0$  and  $t_1$ .  
`Navier_Sokes_Standard`  
{ `equation_non_resolue` (t>t0)\*(t<t1) }

## 5.23 navier\_stokes\_ft\_disc

Description: Two-phase momentum balance equation.

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

See also: `navier_stokes_turbulent` (5.32)

Usage:

```
navier_stokes_ft_disc obj Lire obj {  
    [ equation_interfaces_proprietes_fluide str]  
    [ equation_interfaces_vitesse_imposee str]  
    [ equations_interfaces_vitesse_imposee n word1 word2 ... wordn]  
    [ clipping_courbure_interface int]  
    [ terme_gravite str into ['rho_g', 'grad_i']]  
    [ equation_temperature_mpoint str]  
    [ matrice_pression_invariante ]  
    [ penalisation_forage penalisation_forage]  
    [ equation_temperature_mpoint_vapeur str]  
    [ mpoint_inactif_sur_qdm ]  
    [ mpoint_vapeur_inactif_sur_qdm ]  
    [ 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

- **equation\_interfaces\_proprietes\_fluide** *str*: This keyword is used for liquid-gas, liquid-vapor and fluid-fluid deformable interface, which transported at the Eulerian velocity. When this case is selected, the keyword sequence `Methode_transport vitesse_interpolee` is used in the block `Transport_Interfaces_FT_Disc` to define the velocity field for the displacement of the interface.
- **equation\_interfaces\_vitesse\_imposee** *str*: This keyword is used to specify the velocity field to be used when using an interface that mimics a solid interface moving with a given solid speed of displacement. When this case is selected, the keyword sequence `Methode_transport vitesse_imposee` in the `Transport_Interfaces_FT_Disc` block will define the velocity field for the displacement of the interface.
- **equations\_interfaces\_vitesse\_imposee** *n word1 word2 ... wordn*: This keyword is used to specify the velocity field to be used when using an interface that mimics a solid interface moving with a given solid speed of displacement. When this case is selected, the keyword sequence `Methode_transport vitesse_imposee` in the `Transport_Interfaces_FT_Disc` block will define the velocity field

for the displacement of the interface. If two or more solid interfaces are defined, then the keyword `equations_interfaces_vitesse_imposee` should be used.

- **clipping\_courbure\_interface** *int*: This keyword is used to numerically limit the values of curvature used in the momentum balance equation. Curvature is computed as usual, but values exceeding the clipping value are replaced by this threshold, before using the clipped curvature in the momentum balance. Each time a curvature value is clipped, a counter is increased by one unity and the value of the counter is written in the .err file at the end of the time step. This clipping allows not reducing drastically the time stepping when a geometrical singularity occurs in the interface mesh. However, physical phenomena may be concealed with the use of such a clipping.
- **terme\_gravite** *str into ['rho\_g', 'grad\_i']*: The `Terme_gravite` keyword changes the numerical scheme used for the gravity source term. The default is `grad_i`, which is designed to remove spurious currents around the interface. In this case, the pressure field does not contain the hydrostatic part but only a jump across the interface. This scheme seems not to work very well in vef. The `rho_g` option uses the more traditional source term, equal to  $\rho \cdot g$  in the volume. In this case, the hydrostatic pressure is visible in the pressure field and the boundary conditions in pressure must be set accordingly. This model produces spurious currents in the vicinity of the fluid-fluid interfaces and with the immersed boundary conditions.
- **equation\_temperature\_mpoint** *str*: The `equation_temperature_mpoint` should be used in the case of liquid-vapor flow with phase-change (see the TRUST\_ROOT/doc/TRUST/ft\_chgt\_phase.pdf written in French for more information about the model). The name of the temperature equation, defined with the `convection_diffusion_temperature_ft_disc` keyword, should be given.
- **matrice\_pression\_invariante**: This keyword is a shortcut to be used only when the flow is a single-phase one, with interface tracking only used for solid-fluid interfaces. In this peculiar case, the density of the fluid does not evolve during the computation and the pressure matrix does not need to be actuated at each time step.
- **penalisation\_forcage** *penalisation\_forcage* (5.24): This keyword is used to specify a strong formulation (value set to 0) 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 except some rare cases (see `Ecoulement_Neumann` test case for example) where the second one should be used despite of its slow convergence.
- **equation\_temperature\_mpoint\_vapeur** *str*
- **mpoint\_inactif\_sur\_qdm**
- **mpoint\_vapeur\_inactif\_sur\_qdm**
- **modele\_turbulence** *modele\_turbulence\_hyd\_deriv* (5.25) 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 `lapP=0` is solved with Neuman boundary conditions on pressure if any), `avec_sources` (`lapP=f` is solved with Neuman boundaries conditions and `f` integrating the source terms of the Navier-Stokes equations) and `avec_sources_et_operateurs` (`lapP=f` is solved as with the previous option `avec_sources` but `f` integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection\_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks  $\text{Div}U=0$ . By default, boolean equals 1.
- **solveur\_pression** *solveur\_sys\_base* (10.12) for inheritance: Linear pressure system resolution method.
- **solveur\_bar** *solveur\_sys\_base* (10.12) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and `Source_Qdm_lambdaup`). A file (`solveur.bar`) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt\_projection** *deuxmots* (5.26) for inheritance: nb value : This keyword checks every nb time-steps the equality of velocity divergence to zero. value is the criteria convergency for the solver used.

- **seuil\_divU** *floatfloat* (5.27) 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 (  $\text{lmax}(\text{DivU}) \cdot \text{dtl} < \text{value}$  )  
 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* (5.28) for inheritance: Keyword to post-process particular values.
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinit* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) 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 equations are not solved between time  $t_0$  and  $t_1$ .  
 Navier\_Sokes\_Standard  
 { equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.24 penalisation\_forcage

Description: penalisation\_forcage

See also: objet\_lecture (35)

Usage:

```
{
    [ pression_reference float]
```

```
[ domaine_flottant_fluide x1 x2 (x3) ]
}
```

where

- **pression\_reference** *float*
- **domaine\_flottant\_fluide** *x1 x2 (x3)*

## 5.25 modele\_turbulence\_hyd\_deriv

Description: Basic class for turbulence model for Navier-Stokes equations.

See also: objet\_lecture (35) NUL (5.25.2) mod\_turb\_hyd\_ss\_maille (5.25.3) mod\_turb\_hyd\_rans (5.25.19)

Usage:

```
modele_turbulence_hyd_deriv {
    [ 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 ]
}
```

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* (32): 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* (5.25.1): 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).

### 5.25.1 dt\_impr\_ustar\_mean\_only

Description: `not_set`

See also: objet\_lecture (35)

Usage:

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

where

- **dt\_impr** *float*
- **boundaries** *n word1 word2 ... wordn*

### 5.25.2 NUL

Description: not\_set

See also: modele\_turbulence\_hyd\_deriv (5.25)

Usage:

```
NUL [ correction_visco_turb_pour_controle_pas_de_temps ] [ correction_visco_turb_pour_controle-  
_pas_de_temps_parametre ] [ turbulence_paroi ] [ dt_impr_ustar ] [ dt_impr_ustar_mean_only ] [ nut_max ]  
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**i *turbulence\_paro*i\_base (32): 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* (5.25.1): 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).

### 5.25.3 mod\_turb\_hyd\_ss\_maille

Description: Class for sub-grid turbulence model for Navier-Stokes equations.

See also: modele\_turbulence\_hyd\_deriv (5.25) sous\_maille\_wale (5.25.5) sous\_maille\_smago (5.25.6) combinaison (5.25.7) longueur\_melange (5.25.8) sous\_maille (5.25.9) sous\_maille\_selectif\_mod (5.25.10)

sous\_maille\_selectif (5.25.13) sous\_maille\_elt (5.25.14) sous\_maille\_axi (5.25.16) sous\_maille\_smago\_filtre (5.25.17) sous\_maille\_smago\_dyn (5.25.18)

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_paroit turbulence_paroit_base ]
    [ dt_impr_ustar float ]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only ]
    [ nut_max float ]
}
```

where

- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (5.25.4): 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']*: 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* (32) 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* (5.25.1) 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).



#### 5.25.4 form\_a\_nb\_points

Description: 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.

See also: objet\_lecture (35)

Usage:

**nb dir1 dir2**

where

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

#### 5.25.5 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 (5.25.3)

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_paro turbulence_paro_base]  
[ dt_impr_ustar float]  
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]  
[ nut_max 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* (5.25.4) 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* (32) 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* (5.25.1) 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).

#### 5.25.6 sous\_maille\_smago

Description: Smagorinsky sub-grid turbulence model.

$Nut = Cs1 * Cs1 * l * \sqrt{2 * S * S}$

$K = Cs2 * Cs2 * l * l * 2 * S$

See also: `mod_turb_hyd_ss_maille` (5.25.3)

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]
}
```

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* (5.25.4) 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* (32) 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* (5.25.1) 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).

### 5.25.7 combinaison

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

See also: `mod_turb_hyd_ss_maille` (5.25.3)

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_paro  turbulence_paro_base]
    [ dt_impr_ustar  float]
    [ dt_impr_ustar_mean_only  dt_impr_ustar_mean_only]
    [ nut_max  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* (5.25.4) 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* (32) 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* (5.25.1) 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).

### 5.25.8 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 (5.25.3)

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_paroit turbulence_paroit_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max 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 resume 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 resuming a K-Epsilon calculation with the Champ\_Fonc\_Med keyword.
- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (5.25.4) 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* (32) 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* (5.25.1) 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).

### 5.25.9 sous\_maille

Description: Structure sub-grid function model.

See also: `mod_turb_hyd_ss_maille` (5.25.3)

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_parois turbulence_parois_base ]
    [ dt_impr_ustar float ]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only ]
    [ nut_max float ]
}
```

where

- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (5.25.4) 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* (32) 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* (5.25.1) 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).

#### 5.25.10 sous\_maille\_selectif\_mod

Description: Selective structure sub-grid function model (modified).

See also: `mod_turb_hyd_ss_maille` (5.25.3)

Usage:

```
sous_maille_selectif_mod {
    [ thi deuxentiers]
    [ canal floatentier]
    [ 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]
}
```

where

- **thi** *deuxentiers* (5.25.11): For homogeneous isotropic turbulence (THI), two integers  $k_i$  and  $k_c$  are needed in VDF (not in VEF).
- **canal** *floatentier* (5.25.12):  $h$  `dir_faces_paro`: For a channel flow, the half width  $h$  and the orientation of the wall `dir_faces_paro` are needed.
- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (5.25.4) 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* (32) 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* (5.25.1) 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).

#### 5.25.11 deuxentiers

Description: Two integers.

See also: `objet_lecture` (35)

Usage:

**int1 int2**

where

- **int1** *int*: First integer.
- **int2** *int*: Second integer.

#### 5.25.12 floatentier

Description: A real and an integer.

See also: `objet_lecture` (35)

Usage:

**the\_float the\_int**

where

- **the\_float** *float*: Real.
- **the\_int** *int*: Integer.

#### 5.25.13 sous\_maille\_selectif

Description: Selective structure sub-grid function model (a filter is applied to the structure function).

See also: `mod_turb_hyd_ss_maille` (5.25.3)



Usage:

```
sous_maille_selectif {  
    [ 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 ]  
}
```

where

- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (5.25.4) 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* (32) 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* (5.25.1) 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).



### 5.25.14 sous\_maille\_1elt

Description: Turbulence model sous\_maille\_1elt.

See also: mod\_turb\_hyd\_ss\_maille (5.25.3) sous\_maille\_1elt\_selectif\_mod (5.25.15)

Usage:

```
sous_maille_1elt {  
    [ 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 ]  
}
```

where

- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (5.25.4) 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* (32) 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* (5.25.1) 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).

### 5.25.15 sous\_maille\_1elt\_selectif\_mod

Description: Turbulence model sous\_maille\_1elt\_selectif\_mod.

See also: sous\_maille\_1elt (5.25.14)

Usage:

```
sous_maille_1elt_selectif_mod {
    [ 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]
}
```

where

- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (5.25.4) 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* (32) 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* (5.25.1) 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).

### 5.25.16 sous\_maille\_axi

Description: Structure sub-grid function turbulence model available in cylindrical co-ordinates.

See also: mod\_turb\_hyd\_ss\_maille (5.25.3)

Usage:

```
sous_maille_axi {
    [ 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 ]
}
```

where

- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (5.25.4) 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\_paroit** *turbulence\_paroit\_base* (32) 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* (5.25.1) 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).

### 5.25.17 sous\_maille\_smago\_filtre

Description: Smagorinsky sub-grid turbulence model should be used with low-filter.

See also: `mod_turb_hyd_ss_maille` (5.25.3)

Usage:

```
sous_maille_smago_filtre {
    [ 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_parois turbulence_parois_base ]
    [ dt_impr_ustar float ]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only ]
    [ nut_max float ]
}
```

where

- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (5.25.4) 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\_parois** *turbulence\_parois\_base* (32) 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* (5.25.1) 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).

### 5.25.18 sous\_maille\_smago\_dyn

Description: Dynamic Smagorinsky sub-grid turbulence model (available in VDF discretization only).

See also: `mod_turb_hyd_ss_maille` (5.25.3)

Usage:

```
sous_maille_smago_dyn {
    [ stabilise str into ['6_points', 'moy_euler', 'plans_paralleles']]
    [ nb_points int]
    [ 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]
}
```

where

- **stabilise** *str into ['6\_points', 'moy\_euler', 'plans\_paralleles']*
- **nb\_points** *int*
- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (5.25.4) 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* (32) 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* (5.25.1) 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).

### 5.25.19 mod\_turb\_hyd\_rans

Description: Class for RANS turbulence model for Navier-Stokes equations.

See also: `modele_turbulence_hyd_deriv` (5.25) `k_epsilon` (5.25.20)

Usage:

```
mod_turb_hyd_rans {
    [ eps_min float]
    [ eps_max float]
    [ k_min float]
    [ quiet ]
    [ 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]
}
```

where

- **eps\_min** *float*: Lower limitation of epsilon (default value 1.e-10).
- **eps\_max** *float*: Upper limitation of epsilon (default value 1.e+10).
- **k\_min** *float*: Lower limitation of k (default value 1.e-10).
- **quiet** : To disable printing of information about k and epsilon.
- **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* (32) 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* (5.25.1) 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).

### 5.25.20 k\_epsilon

Description: Turbulence model (k-eps).

See also: `mod_turb_hyd_rans` (5.25.19)

Usage:

```
k_epsilon {
    transport_k_epsilon transport_k_epsilon
    [ modele_fonc_bas_reynolds modele_fonction_bas_reynolds_base ]
    [ cmu float ]
    [ prandtl_k float ]
    [ prandtl_eps float ]
    [ eps_min float ]
    [ eps_max float ]
    [ k_min float ]
    [ quiet ]
    [ 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 ]
}
```

where

- **transport\_k\_epsilon** *transport\_k\_epsilon* (5.39): Keyword to define the (k-eps) transportation equation.
- **modele\_fonc\_bas\_reynolds** *modele\_fonction\_bas\_reynolds\_base* (5.25.21): This keyword is used to set the bas Reynolds model used.
- **cmu** *float*: Keyword to modify the Cmu constant of k-eps model :  $Nut = Cmu * k^2 / \epsilon$  Default value is 0.09
- **prandtl\_k** *float*: Keyword to change the Prk value (default 1.0).
- **prandtl\_eps** *float*: Keyword to change the Pre value (default 1.3).
- **eps\_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **eps\_max** *float* for inheritance: Upper limitation of epsilon (default value 1.e+10).
- **k\_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **quiet** for inheritance: To disable printing of information about k and epsilon.
- **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* (32) 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* (5.25.1) 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  $1.e8$ ).

#### 5.25.21 modele\_fonction\_bas\_reynolds\_base

Description: `not_set`

See also: `objet_lecture` (35) `Lam_Bremhorst` (5.25.22) `Lauder_Sharma` (5.25.25) `Jones_Lauder` (5.25.26)

Usage:

#### 5.25.22 Lam\_Bremhorst

Description: Model described in 'C.K.G.Lam and K.Bremhorst, A modified form of the k- epsilon model for predicting wall turbulence, ASME J. Fluids Engng., Vol.103, p456, (1981)'. Only in VEF.

See also: `modele_fonction_bas_reynolds_base` (5.25.21) `standard_KEps` (5.25.23) `EASM_Baglietto` (5.25.24)

Usage:

```
Lam_Bremhorst {
    [ fichier_distance_paro str]
    [ reynolds_stress_isotrope int]
}
```

where

- **fichier\_distance\_paro** *str*: refer to `distance_paro` keyword
- **reynolds\_stress\_isotrope** *int*: keyword for isotropic Reynolds stress

#### 5.25.23 standard\_KEps

Description: Model described in 'E. Baglietto, CFD and DNS methodologies development for fuel bundle simulations, Nuclear Engineering and Design, 1503–1510 (236), 2006.'

See also: `Lam_Bremhorst` (5.25.22)



Usage:

**standard\_KEps** {

[ **fichier\_distance\_paro**i *str*]  
[ **reynolds\_stress\_isotrope** *int*]

}

where

- **fichier\_distance\_paro**i *str* for inheritance: refer to distance\_paro keyword
- **reynolds\_stress\_isotrope** *int* for inheritance: keyword for isotropic Reynolds stress

#### 5.25.24 EASM\_Baglietto

Description: Model described in ' E. Baglietto and H. Ninokata , A turbulence model study for simulating flow inside tight lattice rod bundles, Nuclear Engineering and Design, 773–784 (235), 2005. '

See also: Lam\_Bremhorst ([5.25.22](#))

Usage:

**EASM\_Baglietto** {

[ **fichier\_distance\_paro**i *str*]  
[ **reynolds\_stress\_isotrope** *int*]

}

where

- **fichier\_distance\_paro**i *str* for inheritance: refer to distance\_paro keyword
- **reynolds\_stress\_isotrope** *int* for inheritance: keyword for isotropic Reynolds stress

#### 5.25.25 Launder\_Sharma

Description: Model described in ' Launder, B. E. and Sharma, B. I. (1974), Application of the Energy-Dissipation Model of Turbulence to the Calculation of Flow Near a Spinning Disc, Letters in Heat and Mass Transfer, Vol. 1, No. 2, pp. 131-138.'

See also: modele\_fonction\_bas\_reynolds\_base ([5.25.21](#))

Usage:

#### 5.25.26 Jones\_Launder

Description: Model described in ' Jones, W. P. and Launder, B. E. (1972), The prediction of laminarization with a two-equation model of turbulence, Int. J. of Heat and Mass transfer, Vol. 15, pp. 301-314.'

See also: modele\_fonction\_bas\_reynolds\_base ([5.25.21](#))

Usage:

## 5.26 deuxmots

Description: Two words.

See also: [objet\\_lecture \(35\)](#)

Usage:

**mot\_1 mot\_2**

where

- **mot\_1** *str*: First word.
- **mot\_2** *str*: Second word.

## 5.27 floatfloat

Description: Two reals.

See also: [objet\\_lecture \(35\)](#)

Usage:

**a b**

where

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

## 5.28 traitement\_particulier

Description: Auxiliary class to post-process particular values.

See also: [objet\\_lecture \(35\)](#)

Usage:

**aco trait\_part acof**

where

- **aco** *str* into [' ']: Opening curly bracket.
- **trait\_part** *traitement\_particulier\_base* ([5.28.1](#)): Type of *traitement\_particulier*.
- **acof** *str* into [' ']: Closing curly bracket.

### 5.28.1 traitement\_particulier\_base

Description: Basic class to post-process particular values.

See also: [objet\\_lecture \(35\)](#) [temperature \(5.28.2\)](#) [canal \(5.28.3\)](#) [ec \(5.28.4\)](#) [thi \(5.28.5\)](#) [chmoy\\_faceperio \(5.28.7\)](#) [profils\\_thermo \(5.28.8\)](#) [brech \(5.28.9\)](#) [ceg \(5.28.10\)](#)

Usage:

### 5.28.2 temperature

Description: not\_set

See also: [traitement\\_particulier\\_base \(5.28.1\)](#)

Usage:

```
temperature {  
  
    bord str  
    direction int  
  
}
```

where

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

### 5.28.3 canal

Description: Keyword for statistics on a periodic plane channel.

See also: `traitement_particulier_base` ([5.28.1](#))

Usage:

```
canal {  
  
    [ dt_impr_moy_spat float]  
    [ dt_impr_moy_temp float]  
    [ debut_stat float]  
    [ fin_stat float]  
    [ pulsation_w float]  
    [ nb_points_par_phase int]  
    [ reprise str]  
  
}
```

where

- **dt\_impr\_moy\_spat** *float*: Period to print the spatial average (default value is 1e6).
- **dt\_impr\_moy\_temp** *float*: Period to print the temporal average (default value is 1e6).
- **debut\_stat** *float*: Time to start the temporal averaging (default value is 1e6).
- **fin\_stat** *float*: Time to end the temporal averaging (default value is 1e6).
- **pulsation\_w** *float*: Pulsation for phase averaging (in case of pulsating forcing term) (no default value).
- **nb\_points\_par\_phase** *int*: Number of samples to represent phase average all along a period (no default value).
- **reprise** *str*: `val_moy_temp_XXXXXX.sauv` : Keyword to resume a calculation with previous averaged quantities.

Note that for thermal and turbulent problems, averages on temperature and turbulent viscosity are automatically calculated. To resume a calculation with phase averaging, `val_moy_temp_XXXXXX.sauv_phase` file is required on the directory where the job is submitted (this last file will be then automatically loaded by TRUST).

### 5.28.4 ec

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

See also: `traitement_particulier_base` ([5.28.1](#))

Usage:

```
ec {  
    [ Ec ]  
    [ Ec_dans_repere_fixe ]  
    [ periode float ]  
}
```

where

- **Ec**
- **Ec\_dans\_repere\_fixe**
- **periode** *float*: *periode* is the keyword to set the period of printing into the file `datafile_Ec.son` or `datafile_Ec_dans_repere_fixe.son`.

### 5.28.5 thi

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

See also: `traitement_particulier_base` ([5.28.1](#)) `thi_thermo` ([5.28.6](#))

Usage:

```
thi {  
    init_Ec int  
    [ val_Ec float ]  
    [ facon_init int into [0, 1] ]  
    [ calc_spectre int into [0, 1] ]  
    [ periode_calc_spectre float ]  
    [ 3D int into [0, 1] ]  
    [ 1D int into [0, 1] ]  
    [ conservation_Ec ]  
    [ longueur_boite float ]  
}
```

where

- **init\_Ec** *int*: Keyword to renormalize initial velocity so that kinetic energy equals to the value given by keyword `val_Ec`.
- **val\_Ec** *float*: Keyword to impose a value for kinetic energy by velocity 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

### 5.28.6 thi\_thermo

Description: Treatment for the temperature field.

It offers the possibility to :

- evaluate the probability density function on temperature field,
- give in a file the temperature field for a future spectral analysis,
- monitor the evolution of the max and min temperature on the whole domain.

See also: thi ([5.28.5](#))

Usage:

```
thi_thermo {
    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* for inheritance: Keyword to renormalize initial velocity so that kinetic energy equals to the value given by keyword **val\_Ec**.
- **val\_Ec** *float* for inheritance: Keyword to impose a value for kinetic energy by velocity renormalized if **init\_Ec** value is 1.
- **facon\_init** *int into [0, 1]* for inheritance: Keyword to specify how kinetic energy is computed (0 or 1).
- **calc\_spectre** *int into [0, 1]* for inheritance: 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* for inheritance: Period for calculating spectrum of kinetic energy
- **3D** *int into [0, 1]* for inheritance: Calculate or not the 3D spectrum
- **1D** *int into [0, 1]* for inheritance: Calculate or not the 1D spectrum
- **conservation\_Ec** for inheritance: If set to 1, velocity field will be changed as to have a constant kinetic energy (default 0)
- **longueur\_boite** *float* for inheritance: Length of the calculation domain

### 5.28.7 chmoy\_faceperio

Description: non documente

See also: `traitement_particulier_base` ([5.28.1](#))

Usage:

**chmoy\_faceperio bloc**

where

- **bloc** *bloc\_lecture* ([3.44](#))

### 5.28.8 **profils\_thermo**

Description: non documente

See also: `traitement_particulier_base` ([5.28.1](#))

Usage:

**profils\_thermo bloc**

where

- **bloc** *bloc\_lecture* ([3.44](#))

### 5.28.9 **brech**

Description: non documente

See also: `traitement_particulier_base` ([5.28.1](#))

Usage:

**brech bloc**

where

- **bloc** *bloc\_lecture* ([3.44](#))

### 5.28.10 **ceg**

Description: Keyword for a CEG ( Gas Entrainment Criteria) calculation. An objective is deepening gas entrainment on the free surface. Numerical analysis can be performed to predict the hydraulic and geometric conditions that can handle gas entrainment from the free surface.

See also: `traitement_particulier_base` ([5.28.1](#))

Usage:

**ceg {**

**frontiere** *str*

**t\_deb** *float*

[ **t\_fin** *float*]

[ **dt\_post** *float*]

**haspi** *float*

[ **debug** *int*]

[ **areva** *ceg\_areva*]

[ **cea\_jaea** *ceg\_cea\_jaea*]

**}**

where

- **frontiere** *str*: To specify the boundaries conditions representing the free surfaces
- **t\_deb** *float*: value of the CEG's initial calculation time
- **t\_fin** *float*: not\_set time during which the CEG's calculation was stopped
- **dt\_post** *float*: periode refers to the printing period, this value is expressed in seconds
- **haspi** *float*: The suction height required to calculate AREVA's criterion
- **debug** *int*
- **areva** *ceg\_areva* (5.28.11): AREVA's criterion
- **cea\_jaea** *ceg\_cea\_jaea* (5.28.12): CEA\_JAEA's criterion

### 5.28.11 ceg\_areva

Description: not\_set

See also: objet\_lecture (35)

Usage:

```
{
    [ c float]
}
```

where

- **c** *float*

### 5.28.12 ceg\_cea\_jaea

Description: not\_set

See also: objet\_lecture (35)

Usage:

```
{
    [ normalise int]
    [ nb_mailles_mini int]
    [ min_critere_q_sur_max_critere_q float]
}
```

where

- **normalise** *int*: renormalize (1) or not (0) values alpha and gamma
- **nb\_mailles\_mini** *int*: Sets the minimum number of cells for the detection of a vortex.
- **min\_critere\_q\_sur\_max\_critere\_q** *float*: Is an optional keyword used to correct the minimum values of Q's criterion taken into account in the detection of a vortex

## 5.29 navier\_stokes\_phase\_field

Description: Navier Stokes equation for the Phase Field problem.

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

See also: navier\_stokes\_standard (5.31)

Usage:

```
navier_stokes_phase_field obj Lire obj {
```

```

approximation_de_boussinesq str into ['oui', 'non']
viscosite_dynamique_constante str into ['oui', 'non']
gravite n x1 x2 ... xn
[ 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

```

- **approximation\_de\_boussinesq** *str* into ['oui', 'non']: To use or not the Boussinesq approximation.
- **viscosite\_dynamique\_constante** *str* into ['oui', 'non']: To use or not a viscosity which will depends on concentration C (in fact, C is the unknown of Cahn-Hilliard equation).
- **gravite** *n x1 x2 ... xn*: Keyword to define gravity in the case Boussinesq approximation is not used.
- **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 equations) and avec\_sources\_et\_operateurs (lapP=f is solved as with the previous option avec\_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection\_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur\_pression** *solveur\_sys\_base* (10.12) for inheritance: Linear pressure system resolution method.
- **solveur\_bar** *solveur\_sys\_base* (10.12) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source\_Qdm\_lambdaup ). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt\_projection** *deuxmots* (5.26) for inheritance: nb value : This keyword checks every nb time-steps the equality of velocity divergence to zero. value is the criteria convergency for the solver used.
- **seuil\_divU** *floatfloat* (5.27) 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 (  $\text{lmax}(\text{DivU}) * dt < \text{value}$  )  
 $\text{Seuil}(t_{n+1}) = \text{Seuil}(t_n) * \text{factor}$



```
Else
Seuil(tn+1)= Seuil(tn)*factor
Endif
```

The first parameter (value) is the mass evolution the user is ready to accept per timestep, and the second one (factor) is the factor of evolution for 'seuil' (for example 1.1, so 10

- **traitement\_particulier** *traitement\_particulier* (5.28) for inheritance: Keyword to post-process particular values.
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

### 5.30 navier\_stokes\_qc

Description: Navier-Stokes equations under low Mach number.

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

See also: navier\_stokes\_standard (5.31)

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  $\text{lapP}=0$  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 equations) 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 equations). 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 equations.
- **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* (10.12) for inheritance: Linear pressure system resolution method.
- **solveur\_bar** *solveur\_sys\_base* (10.12) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and *Source\_Qdm\_lambdaup*). A file (*solveur.bar*) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt\_projection** *deuxmots* (5.26) for inheritance: *nb* value : This keyword checks every *nb* time-steps the equality of velocity divergence to zero. *value* is the criteria convergency for the solver used.
- **seuil\_divU** *floatfloat* (5.27) 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 (  $\text{lmax}(\text{DivU}) \cdot dt < \text{value}$  )  
    $\text{Seuil}(t_{n+1}) = \text{Seuil}(t_n) \cdot \text{factor}$   
 Else  
    $\text{Seuil}(t_{n+1}) = \text{Seuil}(t_n) \cdot \text{factor}$   
 Endif  
 The first parameter (*value*) is the mass evolution the user is ready to accept per timestep, and the second one (*factor*) is the factor of evolution for 'seuil' (for example 1.1, so 10)
- **traitement\_particulier** *traitement\_particulier* (5.28) for inheritance: Keyword to post-process particular values.
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be

separated by a comma)

- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

### 5.31 navier\_stokes\_standard

Description: Navier-Stokes equations.

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

See also: eqn\_base (5.22) navier\_stokes\_turbulent (5.32) navier\_stokes\_qc (5.30) navier\_stokes\_phase\_field (5.29)

Usage:

```
navier_stokes_standard obj Lire obj {
    [ methode_calcul_pression_initiale str into ['avec_les_cl', 'avec_sources', 'avec_sources_et-
    _opérateurs', '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_limités 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  $\text{lapP}=0$  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 equations) 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 equations). 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 equations.
- **projection\_initiale** *int*: 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* (10.12): Linear pressure system resolution method.
- **solveur\_bar** *solveur\_sys\_base* (10.12): This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and *Source\_Qdm\_lambdaup*). A file (*solveur.bar*) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt\_projection** *deuxmots* (5.26): *nb value* : This keyword checks every *nb* time-steps the equality of velocity divergence to zero. *value* is the criteria convergency for the solver used.
- **seuil\_divU** *floatfloat* (5.27): *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 (  $\text{lmax}(\text{DivU}) \cdot dt < \text{value}$  )  
    $\text{Seuil}(t_{n+1}) = \text{Seuil}(t_n) \cdot \text{factor}$   
 Else  
    $\text{Seuil}(t_{n+1}) = \text{Seuil}(t_n) \cdot \text{factor}$   
 Endif  
 The first parameter (*value*) is the mass evolution the user is ready to accept per timestep, and the second one (*factor*) is the factor of evolution for 'seuil' (for example 1.1, so 10)
- **traitement\_particulier** *traitement\_particulier* (5.28): Keyword to post-process particular values.
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limite** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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*  
 $x_1 \ y_1 \ [z_1] \ \text{val}_1$   
 ...  
 $x_n \ y_n \ [z_n] \ \text{val}_n$   
 The created files are named : *pbname\_fieldname\_[boundaryname]\_time.dat*
- **ecrire\_fichier\_xyz\_valeur\_bin** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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*  
 $x_1 \ y_1 \ [z_1] \ \text{val}_1$   
 ...  
 $x_n \ y_n \ [z_n] \ \text{val}_n$   
 The created files are named : *pbname\_fieldname\_[boundaryname]\_time.dat*
- **parametre\_equation** *parametre\_equation\_base* (5.6) for inheritance: Keyword used to specify ad-

ditional parameters for the equation

- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.32 navier\_stokes\_turbulent

Description: Navier-Stokes equations as well as the associated turbulence model equations.

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

See also: navier\_stokes\_standard (5.31) navier\_stokes\_turbulent\_qc (5.33) navier\_stokes\_ft\_disc (5.23)

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* (5.25): 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 lapP=0 is solved with Neuman boundary conditions on pressure if any), avec\_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec\_sources\_et\_operateurs (lapP=f is solved as with the previous option avec\_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicit when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection\_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur\_pression** *solveur\_sys\_base* (10.12) for inheritance: Linear pressure system resolution method.

- **solveur\_bar** *solveur\_sys\_base* (10.12) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source\_Qdm\_lambdaup ). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt\_projection** *deuxmots* (5.26) for inheritance: nb value : This keyword checks every nb time-steps the equality of velocity divergence to zero. value is the criteria convergency for the solver used.
- **seuil\_divU** *floatfloat* (5.27) 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* (5.28) for inheritance: Keyword to post-process particular values.
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) 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 equations are not solved between time t0 and t1.  
 Navier\_Sokes\_Standard  
 { equation\_non\_resolue (t>t0)\*(t<t1) }

### 5.33 navier\_stokes\_turbulent\_qc

Description: Navier-Stokes equations under low Mach number as well as the associated turbulence model equations.

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

See also: `navier_stokes_turbulent` (5.32)

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* (5.25) 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 lapP=0 is solved with Neuman boundary conditions on pressure if any), avec\_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec\_sources\_et\_operateurs (lapP=f is solved as with the previous option avec\_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection\_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur\_pression** *solveur\_sys\_base* (10.12) for inheritance: Linear pressure system resolution method.
- **solveur\_bar** *solveur\_sys\_base* (10.12) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source\_Qdm\_lambdaup ). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt\_projection** *deuxmots* (5.26) for inheritance: nb value : This keyword checks every nb time-steps the equality of velocity divergence to zero. value is the criteria convergency for the solver used.
- **seuil\_divU** *floatfloat* (5.27) 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 ( lmax(DivU)*dtl<value )
Seuil(tn+1)= Seuil(tn)*factor
Else
Seuil(tn+1)= Seuil(tn)*factor
Endif
```

The first parameter (value) is the mass evolution the user is ready to accept per timestep, and the second one (factor) is the factor of evolution for 'seuil' (for example 1.1, so 10

- **traitement\_particulier** *traitement\_particulier* (5.28) for inheritance: Keyword to post-process particular values.
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) 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 equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

### 5.34 transport\_interfaces\_ft\_disc

Description: Interface tracking equation for Front-Tracking problem in the discontinuous version.

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

See also: eqn\_base (5.22)

Usage:

```
transport_interfaces_ft_disc obj Lire obj {
    [ initial_conditions|conditions_initiales bloc_lecture]
```



```

[ methode_transport methode_transport_deriv]
[ iterations_correction_volume int]
[ n_iterations_distance int]
[ maillage str]
[ remaillage bloc_lecture_remaillage]
[ collisions str]
[ methode_interpolation_v str into ['valeur_a_elem', 'vdf_lineaire']]
[ volume_impose_phase_1 float]
[ parcours_interface parcours_interface]
[ interpolation_repere_local ]
[ interpolation_champ_face interpolation_champ_face_deriv]
[ n_iterations_interpolation_ibc int]
[ type_vitesse_imposee str into ['uniforme', 'analytique']]
[ nombre_facettes_retenues_par_cellule int]
[ seuil_convergence_uzawa float]
[ nb_iteration_max_uzawa int]
[ injecteur_interfaces str]
[ vitesse_imposee_regularisee int]
[ indic_faces_modifiee bloc_lecture]
[ distance_projete_faces str into ['simplifiee', 'initiale', 'modifiee']]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ 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

- **initial\_conditions|conditions\_initiales** *bloc\_lecture* (3.44): The keyword *conditions\_initiales* is used to define the shape of the initial interfaces through the zero level-set of a function, or through a mesh *fichier\_geom*. Indicator function is set to 0, that is *fluide0*, where the function is negative; indicator function is set to 1, that is *fluide1*, where the function is positive; the interfaces are the level-set 0 of that function:

```

conditions_initiales { fonction
  ( -((x-0.002)2 + (y-0.002)2 + z2 - (0.00125)2))*((x-0.005)2 + (y-0.007)2 + z2(0.00150)2))*
  (0.020 - z))
}

```

In the above example, there are three interfaces: two bubbles in a liquid with a free surface. One bubble has a radius of 0.00125, i.e. 1.25 mm, and its center is {0.002, 0.002, 0.000}. The other bubble has a radius of 0.00150, i.e. 1.5 mm, and its center is {0.005, 0.007, 0.000}. The free surface is above the two bubble, at a level  $z=0.02$ .

Additional feature in this block concerns the keywords *ajout\_phase0* and *ajout\_phase1*. They can be used to simplify the composition of different interfaces. When using these keywords, the initial function defines the indicator function; *ajout\_phase0* and *ajout\_phase1* are used to modify this initial field. Each time *ajout\_phase0* is used, the field is untouched where the function is positive whereas the indicator field is set to 0 where the function is negative. The keyword *ajout\_phase1* has the symmetrical use, keeping the field value where the function is negative and setting the indicator field to 1 where the function is positive. The previous example can also be written:

```

conditions_initiales {
fonction z-0.020 , NL fonction ajout_phase1  $(x - 0.002)^2 + (y - 0.002)^2 + z^2 - (0.00125)^2$  ,
fonction ajout_phase1  $(x - 0.005)^2 + (y - 0.007)^2 + z^2 - (0.00150)^2$ 
}

```

- **methode\_transport** *methode\_transport\_deriv* (5.35): Method of transport of interface.
- **iterations\_correction\_volume** *int*: Keyword to specify the number of iterations requested for the correction process that can be used to keep the volume of the phases constant during the transport process.
- **n\_iterations\_distance** *int*: Keyword to specify the number of iterations requested for the smoothing process of computing the field corresponding to the signed distance to the interfaces and located at the center of the Eulerian elements. This smoothing is necessary when there are more Lagrangian nodes than Eulerian two-phase cells.
- **maillage** *str*: This optional block is used to specify that we want a Gnuplot drawing of the initial mesh. There is only one keyword, *niveau\_plot*, that is used only to define if a Gnuplot drawing is active (value 1) or not active (value -1). By default, skipping the block will produce non Gnuplot drawing. This option is to be used only in a debug process.
- **remaillage** *bloc\_lecture\_remaillage* (5.36): This block is used to specify the operations that are used to keep the solid interfaces in a proper condition. The remaillage block only contains parameter's values.
- **collisions** *str*: This block is used to specify the operations that are used when a collision occurs between two parts of interfaces. When this occurs, it is necessary to build a new mesh that has locally a clear definition of what is inside and what is outside of the mesh. The collisions can either be active or inactive. If the collisions are active (highly recommended), the keyword *juric\_pour\_tout* indicates that the Juric level-set reconstruction method will be used to re-create the new mesh after each coalescence or breakup. The next line (*type\_remaillage*) is used to state whose field will be used for the level-set computation. Main option is Juric, a remeshing that is compatible with parallel computing. When using Juric level-set remeshing, the source field (*source\_isevalueur*) that is used to compute the level-sets is then defined. It can be either the indicator function (*indicatrice*), a choice which is the default one and the most robust, or a geometrical distance computed from the mesh at the beginning of the time step (*fonction\_distance*), a choice that may be more accurate in specific situations.

Type\_remaillage Thomas is an enhancement of the Juric global remeshing algorithm designed to compensate for mass loss during remeshing. The mesh is always reconstructed with the indicator function (not with the distance function). After having reconstructed the mesh with the Juric algorithm, the difference between the old indicator function (before remeshing) and the new indicator function is computed. The differences occurring at a distance below or equal to *N* elements from the interface are summed up and used to move the interface in the normal direction. The displacement of the interface is such that the volume of each phase after displacement is equal to the volume of the phase before remeshing. *N* (default value 1) must be smaller than *n\_iterations\_distance* (suggested value: 2).

An alternate choice for the remeshing type (*type\_remaillage*) is *collision\_seq*, which is more complex and tries to sew the two meshes that have collided, once the collision zone has been removed. This algorithm does not work in parallel computation.

- **methode\_interpolation\_v** *str* into [*'valeur\_a\_elem'*, *'vdf\_lineaire'*]: In this block, two keywords are possible for method to select the way the interpolation is performed. With the choice *valeur\_a\_elem* the speed of displacement of the nodes of the interfaces is the velocity at the center of the Eulerian element in which each node is located at the beginning of the time step. This choice is the default interpolation method. The choice *VDF\_lineaire* is only available with a VDF discretization (VDF). In this case, the speed of displacement of the nodes of the interfaces is linearly interpolated on the 4 (in 2D) or the 6 (in 3D) Eulerian velocities closest the location of each node at the beginning of the time step. In peculiar situation, this choice may provide a better interpolated value. Of course, this choice is not available with a VEF discretization (VEFPreP1B).
- **volume\_impose\_phase\_1** *float*: this keyword is used to specify the volume of one phase to keep

the volume of the phases constant during the remeshing process. It is an alternate solution to trouble in mass conservation. This option is mainly realistic when only one inclusion of phase 1 is present in the domain. In most other situations, the `iterations_correction_volume` keyword seems easier to justify. The volume to be keep is in m3 and should agree with initial condition.

- **parcours\_interface** *parcours\_interface* (5.37): `Parcours_interface` allows you to configure the algorithm that computes the surface mesh to volume mesh intersection. This algorithm has some serious trouble when the surface mesh points coincide with some faces of the volume mesh. Effects are visible on the indicator function, in VDF when a plane interface coincides with a volume mesh surface. To overcome these problems, the keyword `correction_parcours_thomas` keyword can be used: it allows the algorithm to slightly move some mesh points. This algorithm is experimental and is NOT activated by default.
- **interpolation\_repere\_local** : Triggers a new transport algorithm for the interface: the velocity vector of lagrangian nodes is computed in the moving frame of reference of the center of each connex component, in such a way that relative displacements of nodes within a connex component of the lagrangian mesh are minimized, hence reducing the necessity of barycentering, smooting and local remeshing. Very efficient for bubbly flows.
- **interpolation\_champ\_face** *interpolation\_champ\_face\_deriv* (5.38): It is possible to compute the imposed velocity for the solid-fluid interface by direct affectation (`interpolation_scheme` would be set to `base`) or by multi-linear interpolation (`interpolation_scheme` would be set to `lineaire`). The default value is `base`.
- **n\_iterations\_interpolation\_ibc** *int*: Useful only with `interpolation_champ_face` positioned to `lineaire`. Set the value concerning the width of the region of the linear interpolation. For the Penalized Direct Forcing model, a value equals to 1 is enough.
- **type\_vitesse\_imposee** *str into ['uniforme', 'analytique']*: Useful only with `interpolation_champ_face` positioned to `lineaire`. Value of the keyword is `uniforme` (for an uniform solid-fluide interface's velocity, i.e. zero for instance) or `analytique` (for an analytic expression of the solid-fluide interface's velocity depending on the spatial coordinates). The default value is `uniforme`.
- **nombre\_facettes\_retenues\_par\_cellule** *int*: Keyword to specify the default number (3) of facets per cell used to describe the geometry of the solid-solid interface. This number should be increased if the geometry of the solid-solid interface is complex in each cell (eulerian mesh too coarse for example).
- **seuil\_convergence\_uzawa** *float*: Optional option to change the default value (10-8) of the threshold convergence for the Uzawa algorithm if used in the Penalized Direct Forcing model. Sometime, the value should be decreased to insure a better convergence to force equality between sequential and parallel results.
- **nb\_iteration\_max\_uzawa** *int*: Optional option to change the default value (10-8) of the threshold convergence for the Uzawa algorithm if used in the Penalized Direct Forcing model. Sometime, the value should be decreased to insure a better convergence to force equality between sequential and parallel results.
- **injecteur\_interfaces** *str*
- **vitesse\_imposee\_regularisee** *int*
- **indic\_faces\_modifiee** *bloc\_lecture* (3.44)
- **distance\_projete\_faces** *str into ['simplifiee', 'initiale', 'modifiee']*
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **boundary\_conditions|conditions\_limite** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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 (5.5) for inheritance: This key-
word 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 (5.6) for inheritance: Keyword used to specify ad-
ditional parameters for the equation
• equation_non_resolue str for inheritance: The equation will not be solved while condition(t) is
verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not
solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

```

### 5.35 methode\_transport\_deriv

Description: Basic class for method of transport of interface.

See also: objet\_lecture (35) loi\_horaire (5.35.1) vitesse\_imposee (5.35.2) vitesse\_interpolee (5.35.3)

Usage:

**methode\_transport\_deriv**

#### 5.35.1 loi\_horaire

Description: not\_set

See also: methode\_transport\_deriv (5.35)

Usage:

**loi\_horaire nom\_loi**

where

- **nom\_loi** *str*

#### 5.35.2 vitesse\_imposee

Description: Class to specify that the speed of displacement of the nodes of the interfaces is imposed with an analytical formula.

See also: methode\_transport\_deriv (5.35)

Usage:

**vitesse\_imposee val**

where

- **val** *word1 word2 (word3)*: Analytical formula.

### 5.35.3 vitesse\_interpolee

Description: Class to specify that the interpolation will use the velocity field of the Navier-Stokes equation named `val` to compute the speed of displacement of the nodes of the interfaces.

See also: `methode_transport_deriv` (5.35)

Usage:

**vitesse\_interpolee** `val`  
where

- **val** *str*: Navier-Stokes equation.

### 5.36 bloc\_lecture\_remaillage

Description: Parameters for remeshing.

See also: `objet_lecture` (35)

Usage:

```
{  
    [ pas float]  
    [ pas_lissage float]  
    [ nb_iter_remaillage int]  
    [ nb_iter_barycentrage int]  
    [ relax_barycentrage float]  
    [ critere_arete float]  
    [ critere_remaillage float]  
    [ impr float]  
    [ facteur_longueur_ideale float]  
    [ nb_iter_correction_volume int]  
    [ seuil_dvolume_residuel float]  
    [ lissage_courbure_coeff float]  
    [ lissage_courbure_iterations int]  
    [ lissage_courbure_iterations_systematique int]  
    [ lissage_courbure_iterations_si_remaillage int]  
    [ critere_longueur_fixe float]  
}
```

where

- **pas** *float*: This keyword has default value -1.; when it is set to a negative value there is no remeshing. It is the time step in second (physical time) between two operations of remeshing.
- **pas\_lissage** *float*: This keyword has default value -1.; when it is set to a negative value there is no smoothing of mesh. It is the time step in second (physical time) between two operations of smoothing of the mesh.
- **nb\_iter\_remaillage** *int*: This keyword has default value 0; when it is set to the zero value there is no remeshing. It is the number of iterations performed during a remeshing process.
- **nb\_iter\_barycentrage** *int*: This keyword has default value 0; when it is set to the zero value there is no operation of barycentrage. The barycentrage operation consists in moving each node of the mesh tangentially to the mesh surface and in a direction that let it closer the center of gravity of its neighbors. If `relax_barycentrage` is set to 1, the node is move to the center of gravity. For values lower than unity, the motion is limited to the corresponding fraction. The parameter `nb_iter_barycentrage` is the number of iteration of these node displacements.

- **relax\_barycentrage** *float*: This keyword has default value 0; when it is set to the zero value there is no motion of the nodes. When  $0 < \text{relax\_barycentrage} \leq 1$ , this parameter provides the relaxation ratio to be used in the barycentrage operation described for the keyword `nb_iter_barycentrage`.
- **critere\_arete** *float*: This keyword is used to compute two sub-criteria : the minimum and the maximum edge length ratios used in the process of obtaining edges of length close to `critere_longueur_fixe`. Their respective values are set to  $(1 - \text{critere\_arete})^{**2}$  and  $(1 + \text{critere\_arete})^{**2}$ . The default values of the minimum and the maximum are set respectively to 0.5 and 1.5. When an edge is longer than  $\text{critere\_longueur\_fixe} * (1 + \text{critere\_arete})^{**2}$ , the edge is cut into two pieces; when its length is smaller than  $\text{critere\_longueur\_fixe} * (1 - \text{critere\_arete})^{**2}$ , this edge has to be suppressed.
- **critere\_remaillage** *float*: This keyword was previously used to compute two sub-criteria : the minimum and the maximum length used in the process of remeshing. Their respective values are set to  $(1 - \text{critere\_remaillage})^{**2}$  and  $(1 + \text{critere\_remaillage})^{**2}$ . The default values of the minimum and the maximum are set respectively to 0.2 and 1.7. There are currently not used in data files.
- **impr** *float*: This keyword is followed by a value that specify the printing time period given. The default value is -1, which means no printing.
- **facteur\_longueur\_ideale** *float*: This keyword is used to set a ratio between edge length and the cube root of volume cell for the remeshing process. The default value is 1.0.
- **nb\_iter\_correction\_volume** *int*: This keyword give the maximum number of iterations to be performed trying to satisfy the criterion `seuil_dvolume_residuel`. The default value is 0, which means no iteration.
- **seuil\_dvolume\_residuel** *float*: This keyword give the error volume (in m3) that is accepted to stop the iterations performed to keep the volume constant during the remeshing process. The default value is 0.0.
- **lissage\_courbure\_coeff** *float*: This keyword is used to specify the diffusion coefficient used in the diffusion process of the curvature in the curvature smoothing process with a time step. The default value is 0.05. That value usually provides a stable process. Too small values do not stabilize enough the interface, especially with several Lagrangian nodes per Eulerian cell. Too high values induce an additional macroscopic smoothing of the interface that should physically come from the surface tension and not from this numerical smoothing.
- **lissage\_courbure\_iterations** *int*: This keyword is used to specify the number of iterations to perform the curvature smoothing process. The default value is 1.
- **lissage\_courbure\_iterations\_systematique** *int*: These keywords allow a finer control than the previous `lissage_courbure_iterations` keyword. N1 iterations are applied systematically at each timestep. For proper DNS computation, N1 should be set to 0.
- **lissage\_courbure\_iterations\_si\_remaillage** *int*: N2 iterations are applied only if the local or the global remeshing effectively changes the lagrangian mesh connectivity.
- **critere\_longueur\_fixe** *float*: This keyword is used to specify the ideal edge length for a remeshing process. The default value is -1., which means that the remeshing does not try to have all edge lengths to tend towards a given value.

### 5.37 parours\_interface

Description: allows you to configure the algorithm that computes the surface mesh to volume mesh intersection. This algorithm has some serious trouble when the surface mesh points coincide with some faces of the volume mesh. Effects are visible on the indicator function, in VDF when a plane interface coincides with a volume mesh surface.

To overcome these problems, the keyword `correction_parours_thomas` keyword can be used: it allows the algorithm to slightly move some mesh points. This algorithm, which is experimental and is NOT activated by default, triggers a correction that avoids some errors in the computation of the indicator function for surface meshes that exactly cross some eulerian mesh edges (strongly suggested !).

See also: [objet\\_lecture \(35\)](#)

Usage:

```
{
    [ correction_parcours_thomas ]
}
where
```

- **correction\_parcours\_thomas**

### 5.38 interpolation\_champ\_face\_deriv

Description: not\_set

See also: objet\_lecture (35) base (5.38.1) lineaire (5.38.2)

Usage:

#### 5.38.1 base

Description: not\_set

See also: interpolation\_champ\_face\_deriv (5.38)

Usage:

**base**

#### 5.38.2 lineaire

Description: not\_set

See also: interpolation\_champ\_face\_deriv (5.38)

Usage:

```
lineaire {
    [ vitesse_fluide_explicite ]
}
where
```

- **vitesse\_fluide\_explicite**

### 5.39 transport\_k\_epsilon

Description: The (k-eps) transport equation. To resume from a previous mixing length calculation, an external MED-format file containing reconstructed K and Epsilon quantities can be read (see fichier\_ecriture\_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 Discretize should have already been used to read the object.

See also: eqn\_base (5.22)

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* (5.8) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **initial\_conditions|conditions\_initiales** *condinits* (5.3) for inheritance: Initial conditions.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) 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 equations are not solved between time *t0* and *t1*.  
*Navier\_Sokes\_Standard*  
{ *equation\_non\_resolue* (*t*>*t0*)\*(*t*<*t1*) }

## 5.40 transport\_marqueur\_ft

Description: *not\_set*

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

See also: *eqn\_base* (5.22)

Usage:

**transport\_marqueur\_ft** *obj Lire obj* {



```

[ initial_conditions|conditions_initiales bloc_lecture]
[ injection injection_marqueur]
[ transformation_bulles bloc_lecture]
[ phase_marquee int]
[ methode_transport str into ['vitesse_interpolee', 'vitesse_particules']]
[ methode_couplage str into ['suivi', 'one_way_coupling', 'two_way_coupling']]
[ nb_iterations int]
[ contribution_one_way int into [0, 1]]
[ implicite int into [0, 1]]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ 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

- **initial\_conditions|conditions\_initiales** *bloc\_lecture* (3.44): ne semble pas standard
- **injection** *injection\_marqueur* (5.41): The keyword injection can be used to inject periodically during the calculation some other particles. The syntax for ensemble\_points and proprietes\_particles is the same than the initial conditions for the particles. The keyword t\_debut\_injection give the injection initial time (by default, given by t\_debut\_integration) and dt\_injection gives the injection time period (by default given by dt\_min).
- **transformation\_bulles** *bloc\_lecture* (3.44): This keyword will activate the transformation of an inclusion (small bubbles) into a particle. localisation gives the sub-zones (N number of sub-zones and their names) where the transformation may happen. The diameter size for the inclusion transformation is given by either diameter\_min option, in this case the inclusion will be suppressed for a diameter less than diameter\_size, either by the beta\_transfo option, in this case the inclusion will be suppressed for a diameter less than diameter\_size\*cell\_volume (cell\_volume is the volume of the cell containing the inclusion). interface specifies the name of the inclusion interface and t\_debut\_transfo is the beginning time for the inclusion transformation operation (by default, it is t\_debut\_integr value) and dt\_transfo is the period transformation (by default, it is dt\_min value). In a two phase flow calculation, the particles will be suppressed when entering into the non marked phase
- **phase\_marquee** *int*: Phase number giving the marked phase, where the particles are located (when they leave this phase, they are suppressed). By default, for a the two phase fluide, the particles are supposed to be into the phase 0 (liquid).
- **methode\_transport** *str* into ['vitesse\_interpolee', 'vitesse\_particules']: Kind of transport method for the particles. With vitesse\_interpolee, the velocity of the particles is the velocity a fluid interpolation velocity (option by default). With vitesse\_particules, the velocity of the particles is governed by the resolution of a momentum equation for the particles.
- **methode\_couplage** *str* into ['suivi', 'one\_way\_coupling', 'two\_way\_coupling']: Way of coupling between the fluid and the particles. By default, (keyword suivi), there is no interaction between both. With one\_way\_coupling keyword, the fluid act on the particles. With two\_way\_coupling keyword, besides, particles act on the fluid.
- **nb\_iterations** *int*: Number of sub-timesteps to solve the momentum equation for the particles (1 per default).
- **contribution\_one\_way** *int* into [0, 1]: Activate (1, default) or not (0) the fluid forces on the particles when one\_way\_coupling or two\_way\_coupling coupling method is used.
- **implicite** *int* into [0, 1]: Impliciting (1) or not (0) the time scheme when weight added source term is used in the momentum equation
- **convection** *bloc\_convection* (5.8) for inheritance: Keyword to alter the convection scheme.

- **diffusion** *bloc\_diffusion* (5.2) for inheritance: Keyword to specify the diffusion operator.
- **boundary\_conditions|conditions\_limites** *condlims* (4.10.1) for inheritance: Boundary conditions.
- **sources** *sources* (5.4) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire\_fichier\_xyz\_valeur** *ecrire\_fichier\_xyz\_valeur\_param* (5.5) 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* (5.5) 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* (5.6) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.41 injection\_marqueur

Description: not\_set

See also: objet\_lecture (35)

Usage:

```
{
    ensemble_points bloc_lecture
    proprietes_particules bloc_lecture
    [ t_debut_injection float]
    [ dt_injection float]
```

```
}
```

where

- **ensemble\_points** *bloc\_lecture* (3.44)
- **proprietes\_particules** *bloc\_lecture* (3.44)
- **t\_debut\_injection** *float*
- **dt\_injection** *float*

## 6 algo\_base

Description: Basic class for multi-grid algorithms.

See also: [objet\\_u \(36\)](#) [algo\\_couple\\_1 \(6.1\)](#)

Usage:

## 6.1 algo\_couple\_1

Description: not\_set

See also: [algo\\_base \(6\)](#)

Usage:

```
algo_couple_1 obj Lire obj {  
    [ dt_uniforme ]
```

```
}
```

where

- **dt\_uniforme**

## 7 /\*

### 7.1 /\*

Description: bloc of Comment in a data file.

See also: [objet\\_u \(36\)](#)

Usage:

```
/* comm
```

where

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

## 8 champ\_generique\_base

Description: not\_set

See also: [objet\\_u \(36\)](#) [champ\\_post\\_de\\_champs\\_post \(8.1\)](#) [predefini \(8.15\)](#) [champ\\_post\\_refchamp \(8.17\)](#)

Usage:

### 8.1 champ\_post\_de\_champs\_post

Description: not\_set

See also: [champ\\_generique\\_base \(8\)](#) [champ\\_post\\_operateur\\_eqn \(8.5\)](#) [champ\\_post\\_transformation \(8.19\)](#) [champ\\_post\\_operateur\\_base \(8.4\)](#) [champ\\_post\\_statistiques\\_base \(8.6\)](#) [champ\\_post\\_extraction \(8.10\)](#) [champ\\_post\\_morceau\\_equation \(8.13\)](#) [champ\\_post\\_tparoi\\_vef \(8.18\)](#) [champ\\_post\\_interpolation \(8.12\)](#) [champ\\_post\\_reduction\\_0d \(8.16\)](#)

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* (8): the source field.
- **nom\_source** *str*: To name a source field with the `nom_source` keyword
- **source\_reference** *str*
- **sources\_reference** *list\_nom\_virgule* (8.2)
- **sources** *listchamp\_generique* (8.3): sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 8.2 list\_nom\_virgule

Description: List of name.

See also: `listobj` (34.3)

Usage:  
{ object1 , object2 .... }  
list of *nom\_anonyme* (25.1) separated with ,

## 8.3 listchamp\_generique

Description: XXX

See also: `listobj` (34.3)

Usage:  
{ object1 , object2 .... }  
list of *champ\_generique\_base* (8) separated with ,

## 8.4 champ\_post\_operateur\_base

Description: `not_set`

See also: `champ_post_de_champs_post` (8.1) `champ_post_operateur_gradient` (8.11) `champ_post_operateur-divergence` (8.8)

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* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

## 8.5 champ\_post\_operateur\_eqn

Synonymous: **operateur\_eqn**

Description: `not_set`

See also: `champ_post_de_champs_post` (8.1)

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* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

## 8.6 champ\_post\_statistiques\_base

Description: `not_set`

See also: `champ_post_de_champs_post` (8.1) `correlation` (8.7) `moyenne` (8.14) `ecart_type` (8.9)

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* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

## 8.7 correlation

Synonymous: **champ\_post\_statistiques\_correlation**

Description: to calculate the correlation between the two fields.

See also: `champ_post_statistiques_base` (8.6)

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* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

## 8.8 champ\_post\_operateur\_divergence

Synonymous: **divergence**

Description: To calculate divergency of a given field.

See also: `champ_post_operateur_base` (8.4)

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* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 8.9 ecart\_type

Synonymous: **champ\_post\_statistiques\_ecart\_type**

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

See also: `champ_post_statistiques_base` (8.6)

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* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 8.10 champ\_post\_extraction

Synonymous: **extraction**

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

See also: `champ_post_de_champs_post` (8.1)

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* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 8.11 champ\_post\_operateur\_gradient

Synonymous: **gradient**

Description: To calculate gradient of a given field.

See also: champ\_post\_operateur\_base (8.4)

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* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }



## 8.12 champ\_post\_interpolation

Synonymous: **interpolation**

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

See also: `champ_post_de_champs_post` (8.1)

Usage:

```
champ_post_interpolation obj Lire obj {  
    localisation str  
    [ methode str]  
    [ domaine str]  
    [ optimisation_sous_maillage str into ['default', 'yes', 'no']]  
    [ source champ_generique_base]  
    [ nom_source str]  
    [ source_reference str]  
    [ sources_reference list_nom_virgule]  
    [ sources listchamp_generique]  
}
```

where

- **localisation** *str*: `type_loc` indicate where is done the interpolation (elem for element or som for node).
- **methode** *str*: The optional keyword `methode` is limited to `calculer_champ_post` for the moment.
- **domaine** *str*: the domain name where the interpolation is done (by default, the calculation domain)
- **optimisation\_sous\_maillage** *str* into ['default', 'yes', 'no']
- **source** *champ\_generique\_base* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

## 8.13 champ\_post\_morceau\_equation

Synonymous: **morceau\_equation**

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

See also: `champ_post_de_champs_post` (8.1)

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* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: sources { *Champ\_Post....* { ... } *Champ\_Post..* { ... } }

## 8.14 moyenne

Synonymous: **champ\_post\_statistiques\_moyenne**

Description: to calculate the average of the field over time

See also: **champ\_post\_statistiques\_base** (8.6)

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* (16.1): This option allows to read a converged time averaged field in a .xyz file in order to calculate, when resuming the calculation, the statistics fields (rms, correlation) which depend on this average. In that case, the time averaged field is not updated during the resume of calculation. In this case, the time averaged field must be fully converged to avoid errors when calculating high order statistics.
- **t\_deb** *float* for inheritance: Start of integration time
- **t\_fin** *float* for inheritance: End of integration time
- **source** *champ\_generique\_base* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: sources { *Champ\_Post....* { ... } *Champ\_Post..* { ... } }

## 8.15 predefini

Description: This 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 (8)

Usage:

**predefini** obj Lire obj {

**pb\_champ** *deuxmots*

}

where

- **pb\_champ** *deuxmots* (5.26): { Pb\_champ nom\_pb nom\_champ } : nom\_pb is the problem name and nom\_champ is the selected field name.

## 8.16 champ\_post\_reduction\_0d

Synonymous: **reduction\_0d**

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

See also: champ\_post\_de\_champs\_post (8.1)

Usage:

**champ\_post\_reduction\_0d** obj Lire obj {

**methode** *str* into ['min', 'max', 'moyenne', 'average', 'moyenne\_ponderee', 'weighted\_average', 'somme', 'sum', 'somme\_ponderee', 'weighted\_sum', 'somme\_ponderee\_porosite', 'weighted\_sum\_porosity', 'euclidian\_norm', 'normalized\_euclidian\_norm', 'L1\_norm', 'L2\_norm', 'valeur\_a\_gauche', 'left\_value']

    [ **source** *champ\_generique\_base* ]

    [ **nom\_source** *str* ]

    [ **source\_reference** *str* ]

    [ **sources\_reference** *list\_nom\_virgule* ]

    [ **sources** *listchamp\_generique* ]

}

where

- **methode** *str* into ['min', 'max', 'moyenne', 'average', 'moyenne\_ponderee', 'weighted\_average', 'somme', 'sum', 'somme\_ponderee', 'weighted\_sum', 'somme\_ponderee\_porosite', 'weighted\_sum\_porosity', 'euclidian\_norm', 'normalized\_euclidian\_norm', 'L1\_norm', 'L2\_norm', 'valeur\_a\_gauche', 'left\_value']: name of the reduction method:
  - min for the minimum value,
  - max for the maximum value,
  - average (or moyenne) for a mean,
  - weighted\_average (or moyenne\_ponderee) for a mean ponderated by integration volumes, e.g: cell volumes for temperature and pressure in VDF, volumes around faces for velocity and temperature in VEF,
  - sum (or somme) for the sum of all the values of the field,

- `weighted_sum` (or `somme_ponderee`) for a weighted sum (integral),
- `weighted_average_porosity` (or `moyenne_ponderee_porosite`) and `weighted_sum_porosity` (or `somme_ponderee_porosite`) for the mean and sum weighted by the volumes of the elements, only for ELEM localisation,
- `euclidian_norm` for the euclidian norm,
- `normalized_euclidian_norm` for the euclidian norm normalized,
- `L1_norm` for norm L1,
- `L2_norm` for norm L2
- **source** *champ\_generique\_base* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 8.17 champ\_post\_refchamp

Synonymous: **refchamp**

Description: Field of prolem

See also: `champ_generique_base` (8)

Usage:

**champ\_post\_refchamp** obj Lire obj {

**pb\_champ** *deuxmots*  
[ **nom\_source** *str*]

}

where

- **pb\_champ** *deuxmots* (5.26): { `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

## 8.18 champ\_post\_tparoi\_vef

Synonymous: **tparoi\_vef**

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

See also: `champ_post_de_champs_post` (8.1)

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* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 8.19 champ\_post\_transformation

Synonymous: **transformation**

Description: To create a field with a transformation.

See also: *champ\_post\_de\_champs\_post* (8.1)

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* (8) 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* (8.2) for inheritance
- **sources** *listchamp\_generique* (8.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 9 chimie

Description: Keyword to describe the chemical reactions

See also: `objet_u` ([36](#))

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* ([9.1](#)): list of reactions
- **modele\_micro\_melange** *int*: `modele_micro_melange` (0 by default)
- **constante\_modele\_micro\_melange** *float*: `constante` of `modele` (1 by default)
- **espece\_en\_competition\_micro\_melange** *str*: `espece` in competition in reactions

### 9.1 reactions

Description: list of reactions

See also: `listobj` ([34.3](#))

Usage:

{ object1 , object2 .... }

list of *reaction* ([9.1.1](#)) separated with ,

#### 9.1.1 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` ([35](#))

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 CH<sub>4</sub>+2\*O<sub>2</sub>)
- **produits** *str*: RHS of equation (ex CO<sub>2</sub>+2\*H<sub>2</sub>O)
- **constante\_taux\_reaction** *float*: constante of cinetic K
- **coefficients\_activites** *bloc\_lecture* (3.44): coefficients of activity (exemple { CH<sub>4</sub> 1 O<sub>2</sub> 2 })
- **enthalpie\_reaction** *float*: DH
- **energie\_activation** *float*: Ea
- **exposant\_beta** *float*: Beta
- **contre\_reaction** *float*: K<sub>inv</sub>
- **contre\_energie\_activation** *float*: c<sub>r</sub>-Ea

## 10 class\_generic

Description: not\_set

See also: objet\_u (36) dt\_start (10.5) solveur\_sys\_base (10.12)

Usage:

### 10.1 cholesky

Description: Cholesky direct method.

See also: solveur\_sys\_base (10.12)

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

### 10.2 dt\_calc

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

See also: dt\_start (10.5)

Usage:

**dt\_calc**

### 10.3 dt\_fixe

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

See also: dt\_start (10.5)

Usage:

**dt\_fixe** *value*

where

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

## 10.4 dt\_min

Description: The first iteration is based on dt\_min.

See also: dt\_start (10.5)

Usage:

**dt\_min**

## 10.5 dt\_start

Description: not\_set

See also: class\_generic (10) dt\_calc (10.2) dt\_min (10.4) dt\_fixe (10.3)

Usage:

**dt\_start**

## 10.6 gcp\_ns

Description: not\_set

See also: gcp (10.11)

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 ]
    [ nb_it_max int ]
```

}

where

- **solveur0** *solveur\_sys\_base* (10.12): Solver type.
- **solveur1** *solveur\_sys\_base* (10.12): Solver type.
- **precond** *precond\_base* (27) for inheritance: Keyword to define system preconditioning in order to accelerate resolution by the conjugated gradient. Many parallel preconditioning methods are not equivalent to their sequential counterpart, and you should therefore expect differences, especially when you select a high value of the final residue (seuil). The result depends on the number of



processors and on the mesh splitting. It is sometimes useful to run the solver with no preconditioning at all. In particular:

- when the solver does not converge during initial projection,
- when comparing sequential and parallel computations.

With no preconditioning, except in some particular cases (no open boundary), the sequential and the parallel computations should provide exactly the same results within fpu accuracy. If not, there might be a coding error or the system of equations is singular.

- **precond\_nul** for inheritance: Keyword to not use a preconditioning method.
- **seuil** *float* for inheritance: Value of the final residue. The gradient ceases iteration when the Euclidean residue standard  $\|Ax-B\|$  is less than this value.
- **impr** for inheritance: Keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **quiet** for inheritance: To not displaying any outputs of the solver.
- **save\_matrix|save\_matrice** for inheritance: to save the matrix in a file.
- **optimized** for inheritance: This keyword triggers a memory and network optimized algorithms useful for strong scaling (when computing less than 100 000 elements per processor). The matrix and the vectors are duplicated, common items removed and only virtual items really used in the matrix are exchanged.  
Warning: this is experimental and known to fail in some VEF computations (L2 projection step will not converge). Works well in VDF.
- **nb\_it\_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gcp.

## 10.7 gen

Description: not\_set

See also: solveur\_sys\_base ([10.12](#))

Usage:

**gen data**

where

- **data** *bloc\_lecture* ([3.44](#))

## 10.8 gmres

Description: Gmres method (for non symmetric matrix).

See also: solveur\_sys\_base ([10.12](#))

Usage:

**gmres** obj Lire obj {

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

}

where

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

## 10.9 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 ([10.12](#))

Usage:

```
optimal obj Lire obj {
    seuil float
    [ impr ]
    [ quiet ]
    [ save_matrix|save_matrice ]
    [ frequence_recalc int]
    [ nom_fichier_solveur str]
    [ fichier_solveur_non_recree ]
```

```
}
```

where

- **seuil** *float*: Convergence threshold
- **impr** : To print the convergency of the fastest solver
- **quiet** : To disable printing of information
- **save\_matrix|save\_matrice** : To save the linear system (A, x, B) into a file
- **frequence\_recalc** *int*: To set a time step period (by default, 100) for re-checking the fastest solver
- **nom\_fichier\_solveur** *str*: To specify the file containing the list of the tested solvers
- **fichier\_solveur\_non\_recree** : To avoid the creation of the file containing the list

## 10.10 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

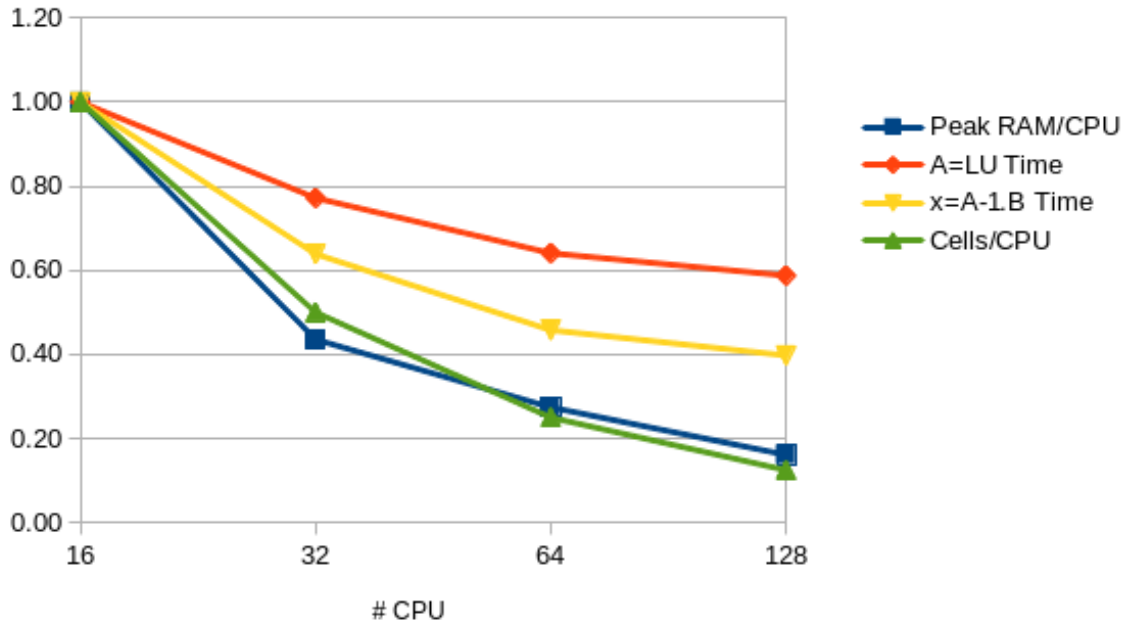
**IBICGSTAB** : Improved version of previous one for massive parallel computations (only a single global reduction operation instead of the usual 3 or 4).

**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 only be used during a parallel calculation, whereas the two first are available for sequential or parallel calculations. It seems that the CPU cost of A=LU factorization but also of the backward/forward elimination steps may sometimes be reduced by selecting a different ordering (Scotch seems often the best for b/f elimination) than the default one. Notice that this solver requires a huge 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 0<sup>th</sup> 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 disk (save RAM memory but add an extra CPU cost during  $Ax=B$  solve)

**CHOLESKY\_SUPERLU** : Parallelized Cholesky from SUPERLU\_DIST library (less CPU and RAM efficient than the previous one)

**CHOLESKY\_PASTIX** : Parallelized Cholesky from PASTIX library

**CHOLESKY\_UMFPACK** : Sequential Cholesky from UMFPACK library (seems fast).

**CLI** { string } : Command Line Interface. Should be used only by advanced users, to access the whole solver/preconditioners from the PETSC API. To find all the available options, run your calculation with the -ksp\_view -help options:

trust datafile [N] -ksp\_view -help

...

#### Preconditioner (PC) Options -----

-pc\_type Preconditioner:(one of) none jacobi pbjacobi bjacobi sor lu shell mg  
eisenstat ilu icc cholesky asm ksp composite redundant nn mat fieldsplit galerkin openmp spai hypre  
tfs (PCSetType)

HYPRE preconditioner options

-pc\_hypre\_type <pilut> (choose one of) pilut parasails boomeramg

HYPRE ParaSails Options

-pc\_hypre\_parasails\_nlevels <1>: Number of number of levels (None)

-pc\_hypre\_parasails\_thresh <0.1>: Threshold (None)

-pc\_hypre\_parasails\_filter <0.1>: filter (None)

-pc\_hypre\_parasails\_loadbal <0>: Load balance (None)

-pc\_hypre\_parasails\_logging: <FALSE> Print info to screen (None)

-pc\_hypre\_parasails\_reuse: <FALSE> Reuse nonzero pattern in preconditioner (None)  
 -pc\_hypre\_parasails\_sym <nonsymmetric> (choose one of) nonsymmetric SPD nonsymmetric, SPD

#### Krylov Method (KSP) Options -----

-ksp\_type Krylov method:(one of) cg cgne stcg gltr richardson chebychev gmres tcqmr  
 bcgs bcgsl cgs tfqmr cr lsqr preonly qcg bicg fgmres minres symmlq lgmres lcd (KSPSetType)  
 -ksp\_max\_it <10000>: Maximum number of iterations (KSPSetTolerances)  
 -ksp\_rtol <0>: Relative decrease in residual norm (KSPSetTolerances)  
 -ksp\_atol <1e-12>: Absolute value of residual norm (KSPSetTolerances)  
 -ksp\_divtol <10000>: Residual norm increase cause divergence (KSPSetTolerances)  
 -ksp\_converged\_use\_initial\_residual\_norm: Use initial residual residual norm for computing relative convergence  
 -ksp\_monitor\_singular\_value <stdout>: Monitor singular values (KSPMonitorSet)  
 -ksp\_monitor\_short <stdout>: Monitor preconditioned residual norm with fewer digits (KSPMonitorSet)  
 -ksp\_monitor\_draw: Monitor graphically preconditioned residual norm (KSPMonitorSet)  
 -ksp\_monitor\_draw\_true\_residual: Monitor graphically true residual norm (KSPMonitorSet)

Example to use the multigrid method as a solver, not only as a preconditioner:

**Solveur\_pression Petsc CLI** { -ksp\_type richardson -pc\_type hypre -pc\_hypre\_type boomeramg -ksp\_atol 1.e-7 }

*Precond* : Several preconditioners are available :

**NULL** { } : No preconditioner used

**BLOCK\_JACOBI\_ICC** { **level** k **ordering** *natural* | **rcm** } : Incomplete Cholesky factorization for symmetric matrix with the PETSc implementation. The integer k is the factorization level (default value, 1). In parallel, the factorization is done by block (one per processor by default). The ordering of the local matrix is **natural** by default, but **rcm** ordering, which reduces the 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 disk (where NBROWS is the number of rows of the matrix and NCPUS the number of CPUs used).
- III) Partition your VEF mesh with a **largeur\_joint** value of 1
- IV) Run your parallel calculation 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 equations):

-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` ([10.12](#))

Usage:

**petsc solveur option\_solveur**

where

- **solveur** *str*
- **option\_solveur** *bloc\_lecture* ([3.44](#))

## **10.11 gcp**

Description: Preconditioned conjugated gradient.

See also: `solveur_sys_base` ([10.12](#)) `gcp_ns` ([10.6](#))

Usage:

**gcp** obj Lire obj {

```

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

```

- **precond** *precond\_base* (27): Keyword to define system preconditioning in order to accelerate resolution by the conjugated gradient. Many parallel preconditioning methods are not equivalent to their sequential counterpart, and you should therefore expect differences, especially when you select a high value of the final residue (**seuil**). The result depends on the number of processors and on the mesh splitting. It is sometimes useful to run the solver with no preconditioning at all. In particular:
  - when the solver does not converge during initial projection,
  - when comparing sequential and parallel computations.
 With no preconditioning, except in some particular cases (no open boundary), the sequential and the parallel computations should provide exactly the same results within fpu accuracy. If not, there might be a coding error or the system of equations is singular.
- **precond\_nul** : Keyword to not use a preconditioning method.
- **seuil** *float*: Value of the final residue. The gradient ceases iteration when the Euclidean residue standard  $\|Ax-B\|$  is less than this value.
- **impr** : Keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **quiet** : To not displaying any outputs of the solver.
- **save\_matrix|save\_matrice** : to save the matrix in a file.
- **optimized** : This keyword triggers a memory and network optimized algorithms useful for strong scaling (when computing less than 100 000 elements per processor). The matrix and the vectors are duplicated, common items removed and only virtual items really used in the matrix are exchanged. Warning: this is experimental and known to fail in some VEF computations (L2 projection step will not converge). Works well in VDF.
- **nb\_it\_max** *int*: Keyword to set the maximum iterations number for the Gcp.

## 10.12 solveur\_sys\_base

Description: Basic class to solve the linear system.

See also: `class_generic` (10) `optimal` (10.9) `gen` (10.7) `petsc` (10.10) `gcp` (10.11) `cholesky` (10.1) `gmres` (10.8)

Usage:

## 11 #

### 11.1 #

Description: Comments in a data file.

See also: `objet_u` (36)

Usage:

# **comm**

where

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

## 12 **condlim\_base**

Description: Basic class of boundary conditions.

See also: [objet\\_u \(36\)](#) [paroi\\_fixe \(12.54\)](#) [symetrie \(12.71\)](#) [periodique \(12.67\)](#) [paroi\\_decalee\\_robin \(12.39\)](#) [paroi\\_adiabatique \(12.36\)](#) [dirichlet \(12.6\)](#) [neumann \(12.35\)](#) [paroi\\_contact \(12.37\)](#) [paroi\\_contact\\_fictif \(12.38\)](#) [paroi\\_echange\\_contact\\_vdf \(12.45\)](#) [paroi\\_echange\\_externes\\_impose \(12.49\)](#) [paroi\\_echange\\_global\\_impose \(12.53\)](#) [Paroi \(12.3\)](#) [frontiere\\_ouverte\\_k\\_eps\\_impose \(12.21\)](#) [paroi\\_flux\\_impose \(12.56\)](#) [frontiere\\_ouverte\\_fraction\\_massique\\_imposee \(12.16\)](#) [paroi\\_echange\\_contact\\_correlation\\_vdf \(12.41\)](#) [paroi\\_echange\\_contact\\_correlation\\_vef \(12.42\)](#) [Neumann\\_homogene \(12.1\)](#) [paroi\\_ft\\_disc \(12.60\)](#) [sortie\\_libre\\_rho\\_variable \(12.69\)](#) [flux\\_radiatif \(12.11\)](#) [contact\\_vdf\\_vef \(12.4\)](#) [contact\\_vef\\_vdf \(12.5\)](#) [echange\\_contact\\_vdf\\_ft\\_disc\\_solid \(12.9\)](#) [echange\\_contact\\_vdf\\_ft\\_disc \(12.8\)](#)

Usage:

**condlim\_base**

### 12.1 **Neumann\_homogene**

Description: Homogeneous neumann boundary condition

See also: [condlim\\_base \(12\)](#) [Neumann\\_pari\\_adiabatique \(12.2\)](#)

Usage:

**Neumann\_homogene**

### 12.2 **Neumann\_pari\_adiabatique**

Description: Adiabatic wall neumann boundary condition

See also: [Neumann\\_homogene \(12.1\)](#)

Usage:

**Neumann\_pari\_adiabatique**

### 12.3 **Paroi**

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

See also: [condlim\\_base \(12\)](#)

Usage:

**Paroi**

### 12.4 **contact\_vdf\_vef**

Description: Boundary condition in the case of two problems (VDF -> VEF).



See also: `condlim_base` ([12](#))

Usage:

**contact\_vdf\_vdf champ**

where

- **champ** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.5 contact\_vdf\_vdf

Description: Boundary condition in the case of two problems (VEF -> VDF).

See also: `condlim_base` ([12](#))

Usage:

**contact\_vdf\_vdf champ**

where

- **champ** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.6 dirichlet

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

See also: `condlim_base` ([12](#)) `paroi_defilante` ([12.40](#)) `paroi_knudsen_non_negligeable` ([12.62](#)) `paroi_rugueuse` ([12.63](#)) `frontiere_ouverte_vitesse_imposee` ([12.33](#)) `frontiere_ouverte_temperature_imposee` ([12.30](#)) `frontiere_ouverte_concentration_imposee` ([12.15](#)) `paroi_temperature_imposee` ([12.64](#)) `scalaire_impose_paro` ([12.68](#))

Usage:

**dirichlet**

## 12.7 echange\_contact\_rayo\_transp\_vdf

Description: Exchange boundary condition in VDF between the transparent fluid and the solid for a problem coupled with radiation. Without radiation, it is the equivalent of the `Paroi_Echange_contact_VDF` exchange condition.

See also: `paroi_echange_contact_vdf` ([12.45](#))

Usage:

**echange\_contact\_rayo\_transp\_vdf autrepb nameb temp h**

where

- **autrepb** *str*: Name of other problem.
- **nameb** *str*: Name of bord.
- **temp** *str*: Name of field.
- **h** *float*: Value assigned to a coefficient (expressed in W.K-1m-2) that characterises the contact between the two mediums. In order to model perfect contact, h must be taken to be infinite. This value must obviously be the same in both the two problems blocks.  
The surface thermal flux exchanged between the two mediums is represented by :  
$$fi = h (T1 - T2)$$
 where  $1/h = d1/\lambda_{a1} + 1/val\_h\_contact + d2/\lambda_{a2}$   
where  $d_i$  : distance between the node where  $T_i$  and the wall is found.

## 12.8 `echange_contact_vdf_ft_disc`

Description: `echange_conatct_vdf` en precisant la phase

See also: `condlim_base` ([12](#))

Usage:

**`echange_contact_vdf_ft_disc`** obj Lire obj {

```
    autre_probleme  str
    autre_bord      str
    autre_champ_temperature  str
    nom_mon_indicatrice  str
    phase           int
```

}

where

- **`autre_probleme`** *str*: name of other problem
- **`autre_bord`** *str*: name of other boundary
- **`autre_champ_temperature`** *str*: name of other field
- **`nom_mon_indicatrice`** *str*: name of indicatrice
- **`phase`** *int*: phase

## 12.9 `echange_contact_vdf_ft_disc_solid`

Description: `echange_conatct_vdf` en precisant la phase

See also: `condlim_base` ([12](#))

Usage:

**`echange_contact_vdf_ft_disc_solid`** obj Lire obj {

```
    autre_probleme  str
    autre_bord      str
    autre_champ_temperature_indic1  str
    autre_champ_temperature_indic0  str
    autre_champ_indicatrice  str
```

}

where

- **`autre_probleme`** *str*: name of other problem
- **`autre_bord`** *str*: name of other boundary
- **`autre_champ_temperature_indic1`** *str*: name of temperature indic 1
- **`autre_champ_temperature_indic0`** *str*: name of temperature indic 0
- **`autre_champ_indicatrice`** *str*: name of indicatrice

## 12.10 `entree_temperature_imposee_h`

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

See also: `frontiere_ouverte_temperature_imposee` ([12.30](#))

Usage:

**`entree_temperature_imposee_h`** ch

where

- **ch** *champ\_front\_base* (17.1): Boundary field type.

## 12.11 flux\_radiatif

Description: Boundary condition for radiation equation.

See also: *condlim\_base* (12) *flux\_radiatif\_vdf* (12.12) *flux\_radiatif\_vef* (12.13)

Usage:

**flux\_radiatif na a ne emissivite**

where

- **na** *str into ['A']*: Keyword for constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **a** *float*: Value of constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **ne** *str into ['emissivite']*: Keyword for wall emissivity.
- **emissivite** *champ\_front\_base* (17.1): Wall emissivity, value between 0 and 1.

## 12.12 flux\_radiatif\_vdf

Description: Boundary condition for radiation equation in VDF.

See also: *flux\_radiatif* (12.11)

Usage:

**flux\_radiatif\_vdf na a ne emissivite**

where

- **na** *str into ['A']*: Keyword for constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **a** *float*: Value of constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **ne** *str into ['emissivite']*: Keyword for wall emissivity.
- **emissivite** *champ\_front\_base* (17.1): Wall emissivity, value between 0 and 1.

## 12.13 flux\_radiatif\_vef

Description: Boundary condition for radiation equation in VEF.

See also: *flux\_radiatif* (12.11)

Usage:

**flux\_radiatif\_vef na a ne emissivite**

where

- **na** *str into ['A']*: Keyword for constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **a** *float*: Value of constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **ne** *str into ['emissivite']*: Keyword for wall emissivity.
- **emissivite** *champ\_front\_base* (17.1): Wall emissivity, value between 0 and 1.

## 12.14 `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` (12.35) `frontiere_ouverte_rayo_transp` (12.26) `frontiere_ouverte_rayo_semi_transp` (12.25)

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* (17.1): Boundary field type.

## 12.15 `frontiere_ouverte_concentration_imposee`

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

See also: `dirichlet` (12.6)

Usage:

**`frontiere_ouverte_concentration_imposee ch`**

where

- **`ch`** *champ\_front\_base* (17.1): Boundary field type.

## 12.16 `frontiere_ouverte_fraction_massique_imposee`

Description: `not_set`

See also: `condlim_base` (12)

Usage:

**`frontiere_ouverte_fraction_massique_imposee ch`**

where

- **`ch`** *champ\_front\_base* (17.1): Boundary field type.

## 12.17 `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` (12.35) `frontiere_ouverte_gradient_pression_impose_vefprep1b` (12.18)

Usage:

**`frontiere_ouverte_gradient_pression_impose ch`**

where

- **`ch`** *champ\_front\_base* (17.1): Boundary field type.

## 12.18 `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` ([12.17](#))

Usage:

**`frontiere_ouverte_gradient_pression_impose_vefprep1b`** **`ch`**

where

- **`ch`** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.19 `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` ([12.35](#))

Usage:

**`frontiere_ouverte_gradient_pression_libre_vef`**

## 12.20 `frontiere_ouverte_gradient_pression_libre_vefprep1b`

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

See also: `neumann` ([12.35](#))

Usage:

**`frontiere_ouverte_gradient_pression_libre_vefprep1b`**

## 12.21 `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 velocity condition.

See also: `condlim_base` ([12](#))

Usage:

**`frontiere_ouverte_k_eps_impose`** **`ch`**

where

- **`ch`** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.22 `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` ([12.35](#))

Usage:

**frontiere\_ouverte\_pression\_imposee ch**

where

- **ch** *champ\_front\_base* (17.1): Boundary field type.

### 12.23 frontiere\_ouverte\_pression\_imposee\_orlansky

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

See also: `neumann` (12.35)

Usage:

**frontiere\_ouverte\_pression\_imposee\_orlansky**

### 12.24 frontiere\_ouverte\_pression\_moyenne\_imposee

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

See also: `neumann` (12.35)

Usage:

**frontiere\_ouverte\_pression\_moyenne\_imposee pext**

where

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

### 12.25 frontiere\_ouverte\_rayo\_semi\_transp

Description: Keyword to set a boundary outlet temperature condition on the boundary called bord (edge) (diffusion flux zero) for a radiation problem with semi transparent gas.

See also: `frontiere_ouverte` (12.14)

Usage:

**frontiere\_ouverte\_rayo\_semi\_transp 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* (17.1): Boundary field type.

### 12.26 frontiere\_ouverte\_rayo\_transp

Description: Keyword to set a boundary outlet temperature condition on the boundary called bord (edge) (diffusion flux zero) for a radiation problem with transparent gas.

See also: `frontiere_ouverte` (12.14) `frontiere_ouverte_rayo_transp_vdf` (12.27) `frontiere_ouverte_rayo_transp_vef` (12.28)

Usage:

**frontiere\_ouverte\_rayo\_transp** **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* (17.1): Boundary field type.

## 12.27 frontiere\_ouverte\_rayo\_transp\_vdf

Description: doit disparaitre

See also: *frontiere\_ouverte\_rayo\_transp* (12.26)

Usage:

**frontiere\_ouverte\_rayo\_transp\_vdf** **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* (17.1): Boundary field type.

## 12.28 frontiere\_ouverte\_rayo\_transp\_vef

Description: doit disparaitre

See also: *frontiere\_ouverte\_rayo\_transp* (12.26)

Usage:

**frontiere\_ouverte\_rayo\_transp\_vef** **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* (17.1): Boundary field type.

## 12.29 frontiere\_ouverte\_rho\_u\_impose

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

See also: *frontiere\_ouverte\_vitesse\_imposee\_sortie* (12.34)

Usage:

**frontiere\_ouverte\_rho\_u\_impose** **ch**

where

- **ch** *champ\_front\_base* (17.1): Boundary field type.

### 12.30 **frontiere\_ouverte\_temperature\_imposee**

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

See also: [dirichlet \(12.6\)](#) [entree\\_temperature\\_imposee\\_h \(12.10\)](#) [frontiere\\_ouverte\\_temperature\\_imposee\\_rayo\\_transp \(12.32\)](#) [frontiere\\_ouverte\\_temperature\\_imposee\\_rayo\\_semi\\_transp \(12.31\)](#)

Usage:

**frontiere\_ouverte\_temperature\_imposee ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

### 12.31 **frontiere\_ouverte\_temperature\_imposee\_rayo\_semi\_transp**

Description: Imposed temperature condition for a radiation problem with semi transparent gas.

See also: [frontiere\\_ouverte\\_temperature\\_imposee \(12.30\)](#)

Usage:

**frontiere\_ouverte\_temperature\_imposee\_rayo\_semi\_transp ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

### 12.32 **frontiere\_ouverte\_temperature\_imposee\_rayo\_transp**

Description: Imposed temperature condition for a radiation problem with transparent gas.

See also: [frontiere\\_ouverte\\_temperature\\_imposee \(12.30\)](#)

Usage:

**frontiere\_ouverte\_temperature\_imposee\_rayo\_transp ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

### 12.33 **frontiere\_ouverte\_vitesse\_imposee**

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

See also: [dirichlet \(12.6\)](#) [frontiere\\_ouverte\\_vitesse\\_imposee\\_sortie \(12.34\)](#)

Usage:

**frontiere\_ouverte\_vitesse\_imposee ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.



### 12.34 **frontiere\_ouverte\_vitesse\_imposee\_sortie**

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

See also: `frontiere_ouverte_vitesse_imposee` ([12.33](#)) `frontiere_ouverte_rho_u_impose` ([12.29](#))

Usage:

**frontiere\_ouverte\_vitesse\_imposee\_sortie** **ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

### 12.35 **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` ([12](#)) `frontiere_ouverte_gradient_pression_libre_vef` ([12.19](#)) `frontiere_ouverte_gradient_pression_libre_vefprep1b` ([12.20](#)) `frontiere_ouverte_gradient_pression_impose` ([12.17](#)) `frontiere_ouverte_pression_imposee` ([12.22](#)) `frontiere_ouverte_pression_imposee_orlansky` ([12.23](#)) `frontiere_ouverte_pression_moyenne_imposee` ([12.24](#)) `frontiere_ouverte` ([12.14](#)) `sortie_libre_temperature_imposee_h` ([12.70](#))

Usage:

**neumann**

### 12.36 **paroi\_adiabatique**

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

See also: `condlim_base` ([12](#))

Usage:

**paroi\_adiabatique**

### 12.37 **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 2-2-2

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

2-2-2-2-2 2-2

OK

2-2 2-2-2

2-4-2 2-2

2-2 2-2

NOT OK

See also: `condlim_base` ([12](#))

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

### 12.38 **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` ([12](#))

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

### 12.39 **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` ([12](#))

Usage:

**paroi\_decalee\_robin** obj Lire obj {

**delta** *float*

}

where

- **delta** *float*

## 12.40 paroi\_defilante

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

See also: [dirichlet \(12.6\)](#)

Usage:

**paroi\_defilante** **ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.41 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 \(12\)](#)

Usage:

**paroi\_echange\_contact\_correlation\_vdf** obj Lire obj {

```
    dir int
    tin 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.
- **tin** *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 resuming calculation with this correlation.

## 12.42 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 ([12](#))

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 resuming calculation with this correlation.

### 12.43 paroi\_echange\_contact\_odvm\_vdf

Description: not\_set

See also: paroi\_echange\_contact\_vdf ([12.45](#))

Usage:

**paroi\_echange\_contact\_odvm\_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  $1/h = d_1/\lambda_{a1} + 1/val\_h\_contact + d_2/\lambda_{a2}$   
where  $d_i$  : distance between the node where  $T_i$  and the wall is found.

### 12.44 paroi\_echange\_contact\_rayo\_semi\_transp\_vdf

Description: Exchange boundary condition in VDF between the semi transparent fluid and the solid for a problem coupled with radiation.

See also: paroi\_echange\_contact\_vdf ([12.45](#))

Usage:

**paroi\_echange\_contact\_rayo\_semi\_transp\_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  $1/h = d_1/\lambda_{a1} + 1/val\_h\_contact + d_2/\lambda_{a2}$   
where  $d_i$  : distance between the node where  $T_i$  and the wall is found.

### 12.45 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 ([12](#)) paroi\_echange\_contact\_vdf\_ft ([12.46](#)) paroi\_echange\_contact\_odvm\_vdf ([12.43](#))

`exchange_contact_rayo_transp_vdf` ([12.7](#)) `paroi_echange_contact_rayo_semi_transp_vdf` ([12.44](#))

Usage:

**paroi\_echange\_contact\_vdf** **autrepb** **nameb** **temp** **h**

where

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

## 12.46 paroi\_echange\_contact\_vdf\_ft

Description: This boundary condition is used between a conduction problem and a thermohydraulic problem with two phases flow (Front-Tracking method) to modelize heat exchange.

See also: `paroi_echange_contact_vdf` ([12.45](#))

Usage:

**paroi\_echange\_contact\_vdf\_ft** **autrepb** **nameb** **temp** **h**

where

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

## 12.47 paroi\_echange\_contact\_vdf\_zoom\_fin

Description: External type exchange condition with a heat exchange coefficient and an imposed external temperature in the case of zoom (fine).

See also: `paroi_echange_externe_impose` ([12.49](#))

Usage:

**paroi\_echange\_contact\_vdf\_zoom\_fin** **h\_imp** **himpc** **text** **ch**

where

- **h\_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ\_front\_base* ([17.1](#)): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.48 paroi\_echange\_contact\_vdf\_zoom\_grossier

Description: External type exchange condition with a heat exchange coefficient and an imposed external temperature in the case of zoom (coarse).

See also: `paroi_echange_externe_impose` ([12.49](#))

Usage:

**paroi\_echange\_contact\_vdf\_zoom\_grossier h\_imp himpc text ch**  
where

- **h\_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himp** *champ\_front\_base* ([17.1](#)): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.49 paroi\_echange\_externe\_impose

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

See also: `condlim_base` ([12](#)) `paroi_echange_externe_impose_h` ([12.50](#)) `paroi_echange_externe_impose_rayo_transp` ([12.52](#)) `paroi_echange_externe_impose_rayo_semi_transp` ([12.51](#)) `paroi_echange_contact_vdf_zoom_grossier` ([12.48](#)) `paroi_echange_contact_vdf_zoom_fin` ([12.47](#))

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).
- **himp** *champ\_front\_base* ([17.1](#)): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.50 paroi\_echange\_externe\_impose\_h

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

See also: `paroi_echange_externe_impose` ([12.49](#))

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).
- **himp** *champ\_front\_base* ([17.1](#)): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

### 12.51 `paroi_echange_externe_impose_rayo_semi_transp`

Description: External type exchange condition for a coupled problem with radiation in semi transparent gas.

See also: `paroi_echange_externe_impose` ([12.49](#))

Usage:

**`paroi_echange_externe_impose_rayo_semi_transp h_imp himpc text ch`**  
where

- **`h_imp`** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **`himpc`** *champ\_front\_base* ([17.1](#)): Boundary field type.
- **`text`** *str*: External temperature value (expressed in oC or K).
- **`ch`** *champ\_front\_base* ([17.1](#)): Boundary field type.

### 12.52 `paroi_echange_externe_impose_rayo_transp`

Description: External type exchange condition for a coupled problem with radiation in transparent gas.

See also: `paroi_echange_externe_impose` ([12.49](#))

Usage:

**`paroi_echange_externe_impose_rayo_transp h_imp himpc text ch`**  
where

- **`h_imp`** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **`himpc`** *champ\_front\_base* ([17.1](#)): Boundary field type.
- **`text`** *str*: External temperature value (expressed in oC or K).
- **`ch`** *champ\_front\_base* ([17.1](#)): Boundary field type.

### 12.53 `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` ([12](#))

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* ([17.1](#)): Boundary field type.
- **`text`** *str*: External temperature value. The external temperature value is expressed in oC or K.
- **`ch`** *champ\_front\_base* ([17.1](#)): Boundary field type.

### 12.54 `paroi_fixe`

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



See also: `condlim_base` ([12](#)) `paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets` ([12.55](#))

Usage:

**paroi\_fixe**

## 12.55 `paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets`

Description: Boundary condition to obtain iso Geneppi2, without interest

See also: `paroi_fixe` ([12.54](#))

Usage:

**paroi\_fixe\_iso\_Genepi2\_sans\_contribution\_aux\_vitesses\_sommets**

## 12.56 `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` ([12](#)) `paroi_flux_impose_rayo_transp` ([12.59](#)) `paroi_flux_impose_rayo_semi_transp_vdf` ([12.57](#)) `paroi_flux_impose_rayo_semi_transp_vef` ([12.58](#))

Usage:

**paroi\_flux\_impose ch**

where

- **ch** `champ_front_base` ([17.1](#)): Boundary field type.

## 12.57 `paroi_flux_impose_rayo_semi_transp_vdf`

Description: Normal flux condition at the wall called bord (edge) for a radiation problem in semi transparent gas (in VDF).

See also: `paroi_flux_impose` ([12.56](#))

Usage:

**paroi\_flux\_impose\_rayo\_semi\_transp\_vdf ch**

where

- **ch** `champ_front_base` ([17.1](#)): Boundary field type.

## 12.58 `paroi_flux_impose_rayo_semi_transp_vef`

Description: Normal flux condition at the wall called bord (edge) for a radiation problem in semi transparent gas (in VEF).

See also: `paroi_flux_impose` ([12.56](#))

Usage:

**paroi\_flux\_impose\_rayo\_semi\_transp\_vef ch**

where

- **ch** `champ_front_base` ([17.1](#)): Boundary field type.

## 12.59 **paroi\_flux\_impose\_rayo\_transp**

Description: Normal flux condition at the wall called bord (edge) for a radiation problem in transparent gas.

See also: `paroi_flux_impose` ([12.56](#))

Usage:

**paroi\_flux\_impose\_rayo\_transp** **ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.60 **paroi\_ft\_disc**

Description: Boundary condition for Front-Tracking problem in the discontinuous version.

See also: `condlim_base` ([12](#))

Usage:

**paroi\_ft\_disc** **type**

where

- **type** *paroi\_ft\_disc\_deriv* ([12.61](#)): Symetrie condition.

## 12.61 **paroi\_ft\_disc\_deriv**

Description: `not_set`

See also: `objet_lecture` ([35](#)) `symetrie` ([12.61.1](#)) `constant` ([12.61.2](#))

Usage:

**paroi\_ft\_disc\_deriv**

### 12.61.1 **symetrie**

Description: Symetrie condition in the case of two-phase flows

See also: `paroi_ft_disc_deriv` ([12.61](#))

Usage:

**symetrie**

### 12.61.2 **constant**

Description: condition contact angle `fidex`. The angle is measured between the wall and the interface in the phase 0.

See also: `paroi_ft_disc_deriv` ([12.61](#))

Usage:

**constant** **ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.62 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 \(12.6\)](#)

Usage:

**paroi\_knudsen\_non\_negligeable** **name\_champ\_1** **champ\_1** **name\_champ\_2** **champ\_2**

where

- **name\_champ\_1** *str into ['vitesse\_paro', 'k']*: Field name.
- **champ\_1** *champ\_front\_base (17.1)*: Boundary field type.
- **name\_champ\_2** *str into ['vitesse\_paro', 'k']*: Field name.
- **champ\_2** *champ\_front\_base (17.1)*: Boundary field type.

## 12.63 paroi\_rugueuse

Description: Rough wall boundary

See also: [dirichlet \(12.6\)](#)

Usage:

**paroi\_rugueuse** **obj** Lire **obj** {

**erugu** *float*

}

where

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

## 12.64 paroi\_temperature\_imposee

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

See also: [dirichlet \(12.6\)](#) [temperature\\_imposee\\_paro \(12.72\)](#) [paroi\\_temperature\\_imposee\\_rayo\\_transp \(12.66\)](#) [paroi\\_temperature\\_imposee\\_rayo\\_semi\\_transp \(12.65\)](#)

Usage:

**paroi\_temperature\_imposee** **ch**

where

- **ch** *champ\_front\_base (17.1)*: Boundary field type.

### 12.65 **paroi\_temperature\_imposee\_rayo\_semi\_transp**

Description: Imposed temperature condition at the wall called bord (edge) for a radiation problem in semi transparent gas.

See also: `paroi_temperature_imposee` ([12.64](#))

Usage:

**paroi\_temperature\_imposee\_rayo\_semi\_transp** **ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

### 12.66 **paroi\_temperature\_imposee\_rayo\_transp**

Description: Imposed temperature condition at the wall called bord (edge) for a radiation problem in transparent gas.

See also: `paroi_temperature_imposee` ([12.64](#))

Usage:

**paroi\_temperature\_imposee\_rayo\_transp** **ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

### 12.67 **periodique**

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

See also: `condlim_base` ([12](#))

Usage:

**periodique**

### 12.68 **scalaire\_impose\_paro**

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

See also: `dirichlet` ([12.6](#))

Usage:

**scalaire\_impose\_paro** **ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.69 sortie\_libre\_rho\_variable

Description: Class to define an outlet boundary condition at which the pressure is defined through the given field, whereas the density of the two-phase flow may varies (value of  $P/\rho$  given in  $\text{Pa}/\text{kg}\cdot\text{m}^{-3}$ ).

See also: [condlim\\_base \(12\)](#)

Usage:

**sortie\_libre\_rho\_variable** **ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.70 sortie\_libre\_temperature\_imposee\_h

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

See also: [neumann \(12.35\)](#)

Usage:

**sortie\_libre\_temperature\_imposee\_h** **ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 12.71 symetrie

Description: 1). For Navier-Stokes equations, this keyword is used to designate a symmetry condition concerning the velocity at the boundary called bord (edge) (normal velocity at the edge equal to zero and tangential velocity gradient at the edge equal to zero); 2). For scalar transport equation, this keyword is used to set a symmetry condition on scalar on the boundary named bord (edge).

See also: [condlim\\_base \(12\)](#)

Usage:

**symetrie**

## 12.72 temperature\_imposee\_paro

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

See also: [paroi\\_temperature\\_imposee \(12.64\)](#)

Usage:

**temperature\_imposee\_paro** **ch**

where

- **ch** *champ\_front\_base* ([17.1](#)): Boundary field type.

## 13 discretisation\_base

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

See also: objet\_u (36) vdf (13.2) vef (13.3) ef (13.1)

Usage:

### 13.1 ef

Description: Element Finite discretization.

See also: discretisation\_base (13)

Usage:

### 13.2 vdf

Description: Finite difference volume discretization.

See also: discretisation\_base (13)

Usage:

### 13.3 vef

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

Warning: it becomes an obsolete discretization.

See also: discretisation\_base (13) vefprep1b (13.4)

Usage:

### 13.4 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 Read. By default, the VEFPreP1B keyword is equivalent to the former VEFPreP1B formulation (v1.5.4 and sooner). P0P1 (if used with the strong formulation for imposed pressure boundary) is equivalent to VEFPreP1B but the convergence is slower. VEFPreP1B dis is equivalent to VEFPreP1B dis Read dis { P0 P1 Changement\_de\_base\_P1Bulle 1 Cl\_pression\_sommet\_faible 0 }

See also: vef (13.3)

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.

## 14 domaine

Description: Keyword to create a domain.

See also: `objet_u` (36) `domaine_ale` (14.1)

Usage:

### 14.1 domaine\_ale

Description: Domain with nodes at the interior of the domain are displaced in an arbitrarily prescribed way thanks to ALE description.

See also: `domaine` (14)

Usage:

## 15 espece

Description: `not_set`

See also: `objet_u` (36)

Usage:

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

where

- **cp** *champ\_base* (16.1): Specific heat value (J.kg-1.K-1).
- **mu** *champ\_base* (16.1): Dynamic viscosity value (kg.m-1.s-1).
- **masse\_molaire** *float*: Gas molar mass.

## 16 champ\_base

### 16.1 champ\_base

Description: Basic class of fields.

See also: [objet\\_u \(36\)](#) [champ\\_don\\_base \(16.3\)](#) [champ\\_ostwald \(16.16\)](#) [champ\\_input\\_base \(16.14\)](#) [champ\\_fonc\\_med \(16.7\)](#) [field\\_uniform\\_keps\\_from\\_ud \(16.24\)](#) [Champ\\_Fonc\\_MEDfile \(16.2\)](#)

Usage:

### 16.2 Champ\_Fonc\_MEDfile

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

See also: [champ\\_base \(16.1\)](#)

Usage:

### 16.3 champ\_don\_base

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

See also: [champ\\_base \(16.1\)](#) [uniform\\_field \(16.27\)](#) [champ\\_uniforme\\_morceaux \(16.20\)](#) [champ\\_fonc\\_xyz \(16.23\)](#) [champ\\_fonc\\_txyz \(16.22\)](#) [champ\\_don\\_lu \(16.4\)](#) [init\\_par\\_partie \(16.25\)](#) [champ\\_tabule\\_temps \(16.19\)](#) [champ\\_fonc\\_t \(16.10\)](#) [champ\\_fonc\\_tabule \(16.11\)](#) [champ\\_init\\_canal\\_sinal \(16.12\)](#) [champ\\_som\\_lu\\_vdf \(16.17\)](#) [champ\\_som\\_lu\\_vef \(16.18\)](#) [tayl\\_green \(16.26\)](#) [champ\\_fonc\\_reprise \(16.8\)](#)

Usage:

### 16.4 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 \(16.3\)](#)

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

### 16.5 champ\_fonc\_fonction

Description: Field that is a function of another field.

See also: [champ\\_fonc\\_tabule \(16.11\)](#) [champ\\_fonc\\_fonction\\_txyz \(16.6\)](#)



Usage:

**champ\_fonc\_function dim inco bloc**

where

- **dim** *int*: Number of field components.
- **inco** *str*: Name of the field (for example: temperature).
- **bloc** *bloc\_lecture* (3.44): 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)).

## 16.6 champ\_fonc\_function\_txyz

Description: this refers to a field that is a function of another field and time and/or space coordinates

See also: `champ_fonc_function` (16.5)

Usage:

**champ\_fonc\_function\_txyz dim inco bloc**

where

- **dim** *int*: Number of field components.
- **inco** *str*: Name of the field (for example: temperature).
- **bloc** *bloc\_lecture* (3.44): 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)).

## 16.7 champ\_fonc\_med

Description: Field to read a data field in a MED-format file .med at a specified time. It is very useful, for example, to resume a calculation with a new or refined geometry. The field post-processed on the new geometry at med format is used as initial condition for the resume.

See also: `champ_base` (16.1)

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.

## 16.8 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 (16.3)

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* (16.9): 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.

## 16.9 fonction\_champ\_reprise

Description: not\_set

See also: objet\_lecture (35)

Usage:

**mot fonction**

where

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

## 16.10 champ\_fonc\_t

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

See also: champ\_don\_base (16.3)

Usage:

**champ\_fonc\_t** **val**

where

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

## 16.11 champ\_fonc\_tabule

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

See also: champ\_don\_base (16.3) champ\_fonc\_fonction (16.5)

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* (3.44): 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)).

## 16.12 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 (16.3)

Usage:

**champ\_init\_canal\_sinal** **dim bloc**

where

- **dim** *int*: Number of field components.
- **bloc** *bloc\_lec\_champ\_init\_canal\_sinal* (16.13): Parameters for the class champ\_init\_canal\_sinal.

## 16.13 bloc\_lec\_champ\_init\_canal\_sinal

Description: Parameters for the class champ\_init\_canal\_sinal.

in 2D:

$U = ucent * y(2h - y) / h / h$

$V = ampli\_bruit * rand + ampli\_sin * \sin(\omega * x)$

rand: unpredictable value between -1 and 1.

in 3D:

$U = ucent * y(2h - y) / h / h$

$V = ampli\_bruit * rand1 + ampli\_sin * \sin(\omega * x)$

$W = ampli\_bruit * rand2$

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

See also: objet\_lecture (35)

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 Z
 Default value for dir\_flow is 0
- **dir\_wall** *int into [0, 1, 2]*: Wall direction for the initialization of the flow in a channel.
  - if dir\_wall=0, the normal to the wall is in X direction
  - if dir\_wall=1, the normal to the wall is in Y direction
  - if dir\_wall=2, the normal to the wall is in Z direction
 Default value for dir\_flow is 1
- **min\_dir\_flow** *float*: Value of the minimum coordinate in the flow direction for the initialization of the flow in a channel. Default value for dir\_flow is 0.
- **min\_dir\_wall** *float*: Value of the minimum coordinate in the wall direction for the initialization of the flow in a channel. Default value for dir\_flow is 0.

## 16.14 champ\_input\_base

Description: not\_set

See also: champ\_base ([16.1](#)) champ\_input\_p0 ([16.15](#))

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*

## 16.15 champ\_input\_p0

Description: not\_set

See also: champ\_input\_base ([16.14](#))

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

## 16.16 champ\_ostwald

Description: This keyword is used to define the viscosity variation law:

$\text{Mu}(T) = K(T) \cdot (D:D/2)^{((n-1)/2)}$

See also: champ\_base ([16.1](#))

Usage:

**champ\_ostwald**

## 16.17 champ\_som\_lu\_vdf

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

See also: champ\_don\_base ([16.3](#))

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 ...

## 16.18 champ\_som\_lu\_vef

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

See also: champ\_don\_base (16.3)

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 ...

## 16.19 champ\_tabule\_temps

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

See also: champ\_don\_base (16.3)

Usage:

**champ\_tabule\_temps dim bloc**

where

- **dim** *int*: Number of field components.
- **bloc** *bloc\_lecture* (3.44): Values as a table. The value of the field at any time is calculated by linear interpolation from this table.

## 16.20 champ\_uniforme\_morceaux

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

See also: champ\_don\_base (16.3) champ\_uniforme\_morceaux\_tabule\_temps (16.21) valeur\_totale\_sur\_volume (16.28)

Usage:

**champ\_uniforme\_morceaux nom\_dom nb\_comp data**

where

- **nom\_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb\_comp** *int*: Number of field components.
- **data** *bloc\_lecture* (3.44): { 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.

## 16.21 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 ([16.20](#))

Usage:

**champ\_uniforme\_morceaux\_tabule\_temps** **nom\_dom** **nb\_comp** **data**  
where

- **nom\_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb\_comp** *int*: Number of field components.
- **data** *bloc\_lecture* ([3.44](#)): { 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.

## 16.22 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 ([16.3](#))

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).

## 16.23 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 ([16.3](#))

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).

## 16.24 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 ([16.1](#))

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

## 16.25 init\_par\_partie

Description: ne marche que pour n\_comp=1

See also: champ\_don\_base ([16.3](#))

Usage:

**init\_par\_partie** n\_comp val1 val2 val3

where

- **n\_comp** *int* into [1]
- **val1** *float*
- **val2** *float*
- **val3** *float*

## 16.26 tayl\_green

Description: Class Tayl\_green.

See also: champ\_don\_base ([16.3](#))

Usage:

**tayl\_green** dim

where

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

## 16.27 uniform\_field

Synonymous: **champ\_uniforme**

Description: Field that is constant in space and stationary.

See also: champ\_don\_base ([16.3](#))

Usage:

**uniform\_field** val

where

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



## 16.28 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 (16.20)

Usage:

**valeur\_totale\_sur\_volume nom\_dom nb\_comp data**

where

- **nom\_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb\_comp** *int*: Number of field components.
- **data** *bloc\_lecture* (3.44): { 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.

## 17 champ\_front\_base

### 17.1 champ\_front\_base

Description: Basic class for fields at domain boundaries.

See also: objet\_u (36) champ\_front\_uniforme (17.27) champ\_front\_fonc\_xyz (17.19) champ\_front\_fonc\_txyz (17.18) champ\_front\_fonc\_pois\_ipsn (17.15) champ\_front\_fonc\_pois\_tube (17.16) champ\_front\_tabule (17.25) champ\_front\_fonction (17.20) champ\_front\_bruit (17.9) champ\_front\_tangentiel\_vef (17.26) champ\_front\_lu (17.21) boundary\_field\_inward (17.3) champ\_front\_pression\_from\_u (17.23) champ\_front\_contact\_vef (17.13) champ\_front\_calc (17.10) champ\_front\_recyclage (17.24) ch\_front\_input (17.5) boundary\_field\_uniform\_keps\_from\_ud (17.4) champ\_front\_normal\_vef (17.22) champ\_front\_MED (17.7) champ\_front\_fonc\_t (17.17) champ\_front\_debit (17.14) Champ\_front\_debit\_QC\_VDF (17.2) champ\_front\_ale (17.8) champ\_front\_vortex (17.28) champ\_front\_zoom (17.29)

Usage:

### 17.2 Champ\_front\_debit\_QC\_VDF

Description: This field is used to define a flow rate field for quasi-compressible fluids in VDF discretization. The flow rate is kept constant during a transient.

See also: champ\_front\_base (17.1)

Usage:

**Champ\_front\_debit\_QC\_VDF dimension liste [ moyen ] pb\_name**

where

- **dimension** *int*: Problem dimension
- **liste** *bloc\_lecture* (3.44): List of the mass flow rate values [kg/s/m2] with the following syntaxe: { val1 ... valdim }
- **moyen** *str*: Option to use rho mean value
- **pb\_name** *str*: Problem name

### 17.3 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` ([17.1](#))

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.

### 17.4 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` ([17.1](#))

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

### 17.5 ch\_front\_input

Description: `not_set`

See also: `champ_front_base` ([17.1](#)) `ch_front_input_uniforme` ([17.6](#))

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**  $n \times 1 \times 2 \dots \times n$
- **probleme** *str*
- **sous\_zone** *str*

## 17.6 ch\_front\_input\_uniforme

Description: for coupling, you can use `ch_front_input_uniforme` which is a `champ_front_uniforme`, which use an external value. It must be used with `Problem.setInputField`.

See also: `ch_front_input` ([17.5](#))

Usage:

```
ch_front_input_uniforme obj Lire obj {
    nb_comp int
    nom str
    [ initial_value  $n \times 1 \times 2 \dots \times n$  ]
    probleme str
    [ sous_zone str ]
}
```

where

- **nb\_comp** *int* for inheritance
- **nom** *str* for inheritance
- **initial\_value**  $n \times 1 \times 2 \dots \times n$  for inheritance
- **probleme** *str* for inheritance
- **sous\_zone** *str* for inheritance

## 17.7 champ\_front\_MED

Description: Field allowing the loading of a boundary condition from a MED file using `Champ_fonc_med`

See also: `champ_front_base` ([17.1](#))

Usage:

```
champ_front_MED champ_fonc_med
where
```

- **champ\_fonc\_med** *champ\_base* ([16.1](#)): a `champ_fonc_med` loading the values of the unknown on a domain boundary

## 17.8 champ\_front\_ale

Description: Class to define a boundary condition on a moving boundary of a mesh.

See also: `champ_front_base` ([17.1](#))

Usage:

```
champ_front_ale val
where
```

- **val**  $n \text{ word1 word2 } \dots \text{ wordn}$ : Example:  
2 20\*0.3\*SIN(6.28\*y)\*COS(20\*t) 0.

## 17.9 champ\_front\_bruite

Description: Field which is variable in time and space in a random manner.

See also: `champ_front_base` ([17.1](#))

Usage:

**champ\_front\_bruite nb\_comp bloc**

where

- **nb\_comp** *int*: Number of field components.
- **bloc** *bloc\_lecture* ([3.44](#)): { [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 velocity:  $u=U0(t)$   $v=U1(t)$   $Uj(t)=Mj+2\cdot Aj\cdot \text{bruit\_blanc}$  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).

## 17.10 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` ([17.1](#))

Usage:

**champ\_front\_calc problem\_name bord field\_name**

where

- **problem\_name** *str*: Name of the other problem to which pb1 is coupled.
- **bord** *str*: Name of the side which is the boundary between the 2 domains in the domain object description associated with the `problem_name` object.
- **field\_name** *str*: Name of the field containing the value that the user wishes to use at the boundary. The `field_name` object must be recognized by the `problem_name` object.

## 17.11 champ\_front\_contact\_rayo\_semi\_transp\_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 with radiation in semi transparent fluid.

See also: `champ_front_contact_vef` ([17.13](#))

Usage:

**champ\_front\_contact\_rayo\_semi\_transp\_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.

## 17.12 champ\_front\_contact\_rayo\_transp\_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 with radiation in transparent fluid.

See also: `champ_front_contact_vef` ([17.13](#))

Usage:

**champ\_front\_contact\_rayo\_transp\_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.

## 17.13 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` ([17.1](#)) `champ_front_contact_rayo_transp_vef` ([17.12](#)) `champ_front_contact_rayo_semi_transp_vef` ([17.11](#))

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.

## 17.14 champ\_front\_debit

Description: This field is used to define a flow rate field instead of a velocity field for a Dirichlet boundary condition on Navier-Stokes equations.

See also: `champ_front_base` ([17.1](#))

Usage:

**champ\_front\_debit ch**  
where

- **ch** *champ\_front\_base* ([17.1](#)): uniform field in space to define the flow rate. It could be, for example, `champ_front_uniforme`, `ch_front_input_uniform` or `champ_front_fonc_t`.

### 17.15 champ\_front\_fonc\_pois\_ipsn

Description: Boundary field champ\_front\_fonc\_pois\_ipsn.

See also: champ\_front\_base ([17.1](#))

Usage:

**champ\_front\_fonc\_pois\_ipsn** **r\_tube** **umoy** **r\_loc**

where

- **r\_tube** *float*
- **umoy** *n x1 x2 ... xn*
- **r\_loc** *x1 x2 (x3)*

### 17.16 champ\_front\_fonc\_pois\_tube

Description: Boundary field champ\_front\_fonc\_pois\_tube.

See also: champ\_front\_base ([17.1](#))

Usage:

**champ\_front\_fonc\_pois\_tube** **r\_tube** **umoy** **r\_loc** **r\_loc\_mult**

where

- **r\_tube** *float*
- **umoy** *n x1 x2 ... xn*
- **r\_loc** *x1 x2 (x3)*
- **r\_loc\_mult** *n1 n2 (n3)*

### 17.17 champ\_front\_fonc\_t

Description: Boundary field that depends only on time.

See also: champ\_front\_base ([17.1](#))

Usage:

**champ\_front\_fonc\_t** **val**

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

### 17.18 champ\_front\_fonc\_txyz

Description: Boundary field which is not constant in space and in time.

See also: champ\_front\_base ([17.1](#))

Usage:

**champ\_front\_fonc\_txyz** **val**

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

### 17.19 champ\_front\_fonc\_xyz

Description: Boundary field which is not constant in space.

See also: champ\_front\_base ([17.1](#))

Usage:

**champ\_front\_fonc\_xyz val**

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

### 17.20 champ\_front\_fonction

Description: boundary field that is function of another field

See also: champ\_front\_base ([17.1](#))

Usage:

**champ\_front\_fonction dim inco expression**

where

- **dim** *int*: Number of field components.
- **inco** *str*: Name of the field (for example: temperature).
- **expression** *str*: keyword to use a analytical expression like 10.\*EXP(-0.1\*val) where val be the keyword for the field.

### 17.21 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 ([17.1](#))

Usage:

**champ\_front\_lu domaine dim file**

where

- **domaine** *str*: Name of domain
- **dim** *int*: number of components
- **file** *str*: path for the read file

### 17.22 champ\_front\_normal\_vef

Description: Field to define the normal vector field standard at the boundary in VEF discretization.

See also: champ\_front\_base ([17.1](#))

Usage:

**champ\_front\_normal\_vef mot vit\_tan**

where

- **mot** *str* into [*'valeur\_normale'*]: Name of vector field.
- **vit\_tan** *float*: normal vector value (positive value for a vector oriented outside to inside).

### 17.23 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` ([17.1](#))

Usage:

**champ\_front\_pression\_from\_u** **expression**

where

- **expression** *str*: value depending of a velocity (like  $2 * u_{moy}^2$ ).

### 17.24 champ\_front\_recyclage

Description: This keyword is used on a boundary to get a field from another boundary. New keyword since the 1.6.1 version which replaces and generalizes several obsolete ones:

`Champ_front_calc_intern`  
`Champ_front_calc_recycl_fluct_pbperio`  
`Champ_front_calc_recycl_champ`  
`Champ_front_calc_intern_2pbs`  
`Champ_front_calc_recycl_fluct`

It is to use, in a general way, on a boundary of a local\_pb problem, a field calculated from a linear combination of an imposed field  $g(x,y,z,t)$  with an instantaneous  $f(x,y,z,t)$  and a spatial mean field  $\langle f \rangle(t)$  or a temporal mean field  $\langle f \rangle(x,y,z)$  extracted from a plane of a problem named pb (pb may be local\_pb itself): For each component i, the field F applied on the boundary will be:

$$F_i(x,y,z,t) = \alpha_i * g_i(x,y,z,t) + \chi_i * [f_i(x,y,z,t) - \beta_i * \langle f_i \rangle]$$

Usage:

**Champ\_front\_recyclage** {

**pb\_champ\_evaluateur** *problem\_name field nb\_comp*  
[ **distance\_plan** *x1 x2 (x3)* ]  
[ **moyenne\_imposee** *methode\_moy [fichier file [second\_file]]* ]  
[ **moyenne\_recyclee** *methode\_recyc [fichier file [second\_file]]* ]  
[ **direction\_anisotrope** *int* ]  
[ **ampli\_moyenne\_imposee** *n x1 x2 ... xn* ]  
[ **ampli\_moyenne\_recyclee** *n x1 x2 ... xn* ]  
[ **ampli\_fluctuation** *n x1 x2 ... xn* ]

}

where:

- **pb\_champ\_evaluateur** *problem\_name field nb\_comp*: To give the name of the problem, the name of the field of the problem and its number of components nb\_comp.
- **distance\_plan** *x1 x2 (x3)*: Vector which gives the distance between the boundary and the plane from where the field F will be extracted. By default, the vector is zero, that should imply the two domains have coincident boundaries.
- **ampli\_moyenne\_imposee** *2|3 alpha(0) alpha(1) [alpha(2)]*:  $\alpha_i$  coefficients (by default =1)
- **ampli\_moyenne\_recyclee** *2|3 beta(0) beta(1) [beta(2)]*:  $\beta_i$  coefficients (by default =1)



- **ampli\_fluctuation** 2|3 *gamma(0) gamma(1) [gamma(2)]*: *gamma\_i* coefficients (by default =1)
- **direction\_anisotrope** *int into [1,2,3]*: If an integer is given for direction (X:1, Y:2, Z:3, by default, direction is negative), the imposed field *g* will be 0 for the 2 other directions.
- **moyenne\_imposee** *methode\_moy*: Value of the imposed *g* field. The *methode\_moy* option can be:

**profil** [2|3] *valx(x,y,z,t) valy(x,y,z,t) [valz(x,y,z,t)]*: To specify analytic profile for the imposed *g* field.

**interpolation fichier** *file*: To create an imposed field built by interpolation of values read from a file. The imposed field is applied on the direction given by the keyword *direction\_anisotrope* (the field is zero for the other directions). The format of the file is:

```
pos(1) val(1)
pos(2) val(2)
...
pos(N) val(N)
```

If direction given by *direction\_anisotrope* is 1 (or 2 or 3), then *pos* will be X (or Y or Z) coordinate and *val* will be X value (or Y value, or Z value) of the imposed field.

**connexion\_approchee fichier** *file*: To read the imposed field from a file where positions and values are given (it is not necessary that the coordinates of points match the coordinates of the boundary faces, indeed, the nearest point of each face of the boundary will be used). The format of the file is:

```
N
x(1) y(1) [z(1)] valx(1) valy(1) [valz(1)]
x(2) y(2) [z(2)] valx(2) valy(2) [valz(2)]
...
x(N) y(N) [z(N)] valx(N) valy(N) [valz(N)]
```

**connection\_exacte fichier** *file second\_file*: To read the imposed field from two files. The first file contains the points coordinates (which should be the same as the coordinates of the boundary faces) and the *second\_file* contains the mean values. The format of the first file is:

```
N
1 x(1) y(1) [z(1)]
2 x(2) y(2) [z(2)]
...
N x(N) y(N) [z(N)]
```

while the format of the *second\_file* is:

```
N
1 valx(1) valy(1) [valz(1)]
2 valx(2) valy(2) [valz(2)]
...
N valx(N) valy(N) [valz(N)]
```

**logarithmique diametre** *float u\_tau float visco\_cin float direction int*: To specify the imposed field (in this case, velocity) by an analytical logarithmic law of the wall:

$$g(x,y,z) = u\_tau * ( \log(0.5*diametre*u\_tau/visco\_cin)/Kappa + 5.1 )$$

with  $g(x,y,z)=u(x,y,z)$  if **direction** is set to 1 ( $g=v(x,y,z)$  if **direction** is set to 2, and  $g=w(x,y,z)$  if it is set to 3)

- **moyenne\_recylee** *methode\_recyc*: Method used to perform a spatial or a temporal averaging of *f* field to specify <f>. <f> can be the surface mean of *f* on the plane (surface option, see below) or it can be read from several files (for example generated by the *chmoy\_faceperio* option of the *Traitement\_particulier* keyword to obtain a temporal mean field). The option *methode\_recyc* can be:

**surfacique:** Surface mean for  $\langle f \rangle$  from  $f$  values on the plane

Or one of the following *methode\_moy* options applied to read a temporal mean field  $\langle f \rangle(x,y,z)$ :

**interpolation**

**connexion\_approchee**

**connexion\_exacte**

See also: `champ_front_base` ([17.1](#))

Usage:

**champ\_front\_recyclage bloc**

where

- **bloc** *str*

## 17.25 champ\_front\_tabule

Description: Constant field on the boundary, tabulated as a function of time.

See also: `champ_front_base` ([17.1](#))

Usage:

**champ\_front\_tabule nb\_comp bloc**

where

- **nb\_comp** *int*: Number of field components.
  - **bloc** *bloc\_lecture* ([3.44](#)): {nt1 t2 t3 ....tn u1 [v1 w1 ...] u2 [v2 w2 ...] u3 [v3 w3 ...] ... un [vn wn ...]}
- Values are entered into a table based on  $n$  couples ( $t_i$ ,  $u_i$ ) if `nb_comp` value is 1. The value of a field at a given time is calculated by linear interpolation from this table.

## 17.26 champ\_front\_tangentiel\_vef

Description: Field to define the tangential velocity vector field standard at the boundary in VEF discretization.

See also: `champ_front_base` ([17.1](#))

Usage:

**champ\_front\_tangentiel\_vef mot vit\_tan**

where

- **mot** *str* into [*'vitesse\_tangentielle'*]: Name of vector field.
- **vit\_tan** *float*: Vector field standard [m/s].

### 17.27 champ\_front\_uniforme

Description: Boundary field which is constant in space and stationary.

See also: `champ_front_base` ([17.1](#))

Usage:

**champ\_front\_uniforme** **val**

where

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

### 17.28 champ\_front\_vortex

Description: `not_set`

See also: `champ_front_base` ([17.1](#))

Usage:

**champ\_front\_vortex** **dom geom nu utau**

where

- **dom** *str*: Name of domain.
- **geom** *str*
- **nu** *float*
- **utau** *float*

### 17.29 champ\_front\_zoom

Description: Basic class for fields at boundaries of two problems (global problem and local problem).

See also: `champ_front_base` ([17.1](#))

Usage:

**champ\_front\_zoom** **pbMg pb\_1 pb\_2 bord inco**

where

- **pbMg** *str*: Name of multi-grid problem.
- **pb\_1** *str*: Name of first problem.
- **pb\_2** *str*: Name of second problem.
- **bord** *str*: Name of bord.
- **inco** *str*: Name of field.

## 18 loi\_etat\_base

Description: Basic class for state laws.

See also: `objet_u` ([36](#)) `gaz_parfait` ([18.3](#)) `gaz_reel_rhot` ([18.1](#)) `melange_gaz_parfait` ([18.2](#))

Usage:

## 18.1 gaz\_reel\_rhot

Description: Real gas.

See also: `loi_etat_base` ([18](#))

Usage:

**gaz\_reel\_rhot** bloc

where

- **bloc** *bloc\_lecture* ([3.44](#)): Description.

## 18.2 melange\_gaz\_parfait

Description: Mixing of perfect gas.

See also: `loi_etat_base` ([18](#))

Usage:

**melange\_gaz\_parfait** obj Lire obj {

```
    sc float  
    [ cp float]  
    prandtl float  
    [ correction_fraction ]  
    [ ignore_check_fraction ]  
    [ dtol_fraction float]
```

}

where

- **sc** *float*: Schmidt number of the gas  $Sc = \nu/D$  (D: diffusion coefficient of the mixing).
- **cp** *float*: Specific heat at constant pressure of the gas  $C_p$ .
- **prandtl** *float*: Prandtl number of the gas  $Pr = \mu * C_p / \lambda$
- **correction\_fraction** : To force mass fractions between 0. and 1.
- **ignore\_check\_fraction** : Not to check if mass fractions between 0. and 1.
- **dtol\_fraction** *float*: Delta tolerance on mass fractions for check testing (default value 1.e-6).

## 18.3 gaz\_parfait

Description: Perfect gas.

See also: `loi_etat_base` ([18](#))

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*:  $C_p/C_v$
- **Prandtl** *float*: Prandtl number of the gas  $Pr = \mu * C_p / \lambda$
- **rho\_constant\_pour\_debug** *champ\_base* ([16.1](#))

## 19 loi\_fermeture\_base

Description: Class for appends fermeture to problem

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

See also: [objet\\_u \(36\)](#) [loi\\_fermeture\\_test \(19.1\)](#)

Usage:

### 19.1 loi\_fermeture\_test

Description: Loi for test only

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

See also: [loi\\_fermeture\\_base \(19\)](#)

Usage:

```
loi_fermeture_test obj Lire obj {
    [ coef float ]
}
```

where

- **coef** *float*: coefficient

## 20 loi\_horaire

Description: to define the movement with a time-dependant law for the solid interface.

See also: [objet\\_u \(36\)](#)

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*

## 21 milieu\_base

Description: Basic class for medium (physics properties of medium).

See also: objet\_u (36) solide (21.6) constituant (21.1) fluide\_incompressible (21.2) solide\_milieu\_variable (21.7)

Usage:

```
milieu_base obj Lire obj {  
    [ rho champ_base]  
    [ cp champ_base]  
    [ lambda champ_base]  
}  
where
```

- **rho** *champ\_base* (16.1): Density (kg.m-3).
- **cp** *champ\_base* (16.1): Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* (16.1): Conductivity (W.m-1.K-1).

### 21.1 constituant

Description: Constituent.

See also: milieu\_base (21)

Usage:

```
constituant obj Lire obj {  
    [ coefficient_diffusion champ_base]  
    [ rho champ_base]  
    [ cp champ_base]  
    [ lambda champ_base]  
}  
where
```

- **coefficient\_diffusion** *champ\_base* (16.1): Constituent diffusion coefficient value (m<sup>2</sup>.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* (16.1) for inheritance: Density (kg.m-3).
- **cp** *champ\_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).

### 21.2 fluide\_incompressible

Description: This is a uncompressible fluid.

See also: milieu\_base (21) fluide\_quasi\_compressible (21.4) fluide\_ostwald (21.3)

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 (16.1): Thermal expansion (K-1).
- **mu** champ\_base (16.1): Dynamic viscosity (kg.m-1.s-1).
- **beta\_co** champ\_base (16.1): Volume expansion coefficient values in concentration.
- **indice** champ\_base (16.1): Refractivity of fluid.
- **kappa** champ\_base (16.1): Absorptivity of fluid (m-1).
- **rho** champ\_base (16.1) for inheritance: Density (kg.m-3).
- **cp** champ\_base (16.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** champ\_base (16.1) for inheritance: Conductivity (W.m-1.K-1).

### 21.3 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 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 (21.2)

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 (16.1): Fluid consistency.
- **n** champ\_base (16.1): Fluid structure index.
- **beta\_th** champ\_base (16.1) for inheritance: Thermal expansion (K-1).
- **mu** champ\_base (16.1) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **beta\_co** champ\_base (16.1) for inheritance: Volume expansion coefficient values in concentration.
- **indice** champ\_base (16.1) for inheritance: Refractivity of fluid.
- **kappa** champ\_base (16.1) for inheritance: Absorptivity of fluid (m-1).

- **rho** *champ\_base* (16.1) for inheritance: Density (kg.m-3).
- **cp** *champ\_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).

## 21.4 fluide\_quasi\_compressible

Description: Compressible flow at low mach number.

See also: fluide\_incompressible (21.2)

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* (21.5): Sutherland law for viscosity and for conductivity.
- **pression** *float*: Initial pressure.
- **loi\_etat** *loi\_etat\_base* (18): 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\*g (It is the default). 2) 'moins\_rho\_moyen': the gravity term is evaluated with (rho-rhomoy) \*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* (16.1) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **indice** *champ\_base* (16.1) for inheritance: Refractivity of fluid.
- **kappa** *champ\_base* (16.1) for inheritance: Absorptivity of fluid (m-1).
- **rho** *champ\_base* (16.1) for inheritance: Density (kg.m-3).
- **cp** *champ\_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).



## 21.5 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  $\lambda(T)=\mu_0*C_p/Prandtl*((T_0+Slambda)/(T+Slambda))*(T/T_0)**1.5$

See also: [objet\\_lecture \(35\)](#)

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*

## 21.6 solide

Description: Solid.

See also: [milieu\\_base \(21\)](#)

Usage:

**solide** obj Lire obj {

[ **rho** *champ\_base*  
[ **cp** *champ\_base*  
[ **lambda** *champ\_base*

}

where

- **rho** *champ\_base* ([16.1](#)) for inheritance: Density (kg.m-3).
- **cp** *champ\_base* ([16.1](#)) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* ([16.1](#)) for inheritance: Conductivity (W.m-1.K-1).

## 21.7 solide\_milieu\_variable

Description: Solid with cp and/or rho non-uniform.

See also: [milieu\\_base \(21\)](#)

Usage:

**solide\_milieu\_variable** obj Lire obj {

[ **rho** *champ\_base*  
[ **cp** *champ\_base*  
[ **lambda** *champ\_base*

}

where

- **rho** *champ\_base* (16.1) for inheritance: Density (kg.m-3).
- **cp** *champ\_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).

## 22 milieu\_v2\_base

Description: Basic class for medium (physics properties of medium) composed of constituents (fluids and solids).

See also: [objet\\_u](#) (36) [fluide\\_diphasique](#) (22.1)

Usage:

### 22.1 fluide\_diphasique

Description: Two-phase fluid.

See also: [milieu\\_v2\\_base](#) (22)

Usage:

**fluide\_diphasique bloc**

where

- **bloc** *bloc\_lecture* (3.44): Two-phase fluid description.

## 23 modele\_rayonnement\_base

Description: Basic class for wall thermal radiation model.

See also: [objet\\_u](#) (36) [modele\\_rayonnement\\_milieu\\_transparent](#) (23.1)

Usage:

### 23.1 modele\_rayonnement\_milieu\_transparent

Description: Wall thermal radiation model for a transparent gas and resolving a radiation-conduction-thermohydraulics coupled problem in VDF or VEF.

Modele\_Rayonnement\_Milieu\_Transparent mod

Read mod {

nom\_pb\_rayonnant

problem\_name

fichier\_fij

file\_name

fichier\_face\_rayo

file\_name

[fichier\_matrice | fichier\_matrice\_binaire file\_name]

}

nom\_pb\_rayonnant problem\_name : problem\_name is the name of the radiating fluid problem

fichier\_fij file\_name : file\_name is the name of the file which contains the shape factor matrix between all the faces.

fichier\_face\_rayo file\_name : file\_name is the name of the file which contains the radiating faces characteristics (area, emission value ...)

fichier\_matrice|fichier\_matrice\_binaire file\_name : file\_name is the name of the ASCII (or binary) file which contains the inverted shape factor matrix. It is an optional keyword, if not defined, the inverted shape factor matrix will be calculated and written in a file.

The two first files can be generated by a preprocessor, they allow the radiating face characteristics to be entered (set of faces considered to be uniform with respect to radiation for emission value, flux, etc.) and the form factors for these various faces. These files have the following format:

File on radiating faces:

N M -> N nombre de faces rayonnantes (=bords) et

(N is the number of radiating faces (=edges) and

-> M nombre de faces rayonnantes a emissivitee non nulle

M equals the number of non-zero emission radiating faces

Nom(i) S(i) E(i) -> Nom du bord i, surface du bord i, valeur de

(Name of the edge i, surface area of the edge i)

-> l'emissivite (comprise entre 0 et 1) (emission value (between 0 and 1))

Exemple:

13 4

Gauche 50.0 0.0

Droit1 50.0 0.5

Bas 10.0 0.0

Haut 10.0 0.0

Arriere 5.0 0.0

Avant 5.0 0.0

Droit2 30.0 0.5

Bas1 40.0 0.0

Haut1 20.0 0.0

Avant1 20.0 0.0

Arriere1 20.0 0.0

Entree 20.0 0.5

Sortie 20.0 0.5

File on form factors:

N -> Nombre de faces rayonnantes (Number of radiating faces)

Fij -> Matrice des facteurs de formes avec i,j entre 1 et N (Matrix of form factors where i, j between 1 and N)

Example:

13

1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

0.00 0.00 0.00 0.00 0.00 0.00 0.24 0.20 0.10 0.10 0.10 0.10 0.16

0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

0.00 0.40 0.00 0.00 0.00 0.00 0.00 0.20 0.10 0.10 0.10 0.10 0.00

0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.00 0.15 0.10 0.10 0.15 0.10

0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.30 0.00 0.10 0.10 0.00 0.10

0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.20 0.10 0.00 0.10 0.10 0.10

0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.20 0.10 0.10 0.00 0.10 0.10

0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.30 0.00 0.10 0.10 0.00 0.10

0.00 0.40 0.00 0.00 0.00 0.00 0.00 0.20 0.10 0.10 0.10 0.10 0.00

Caution:

a) The radiation model's precision is decided by the user when he/she names the domain edges. In fact, a radiating face is recognised by the preprocessor as the set of domain edges faces bearing the same name. Thus, if the user subdivides the edge into two edges which are named differently, he/she thus creates two radiating faces instead of one.

- b) The form factors are entered by the user, the preprocessor carries out no calculations other than checking preservation relationships on form factors.
- c) The fluid is considered to be a transparent gas.

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

See also: `modele_rayonnement_base` (23)

Usage:

**modele\_rayonnement\_milieu\_transparent bloc**

where

- **bloc** *bloc\_lecture* (3.44): See description.

## 24 modele\_turbulence\_scal\_base

Description: Basic class for turbulence model for energy equation.

See also: `objet_u` (36) `prandtl` (24.1) `schmidt` (24.2) `sous_maille_dyn` (24.3)

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* (33): 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».

### 24.1 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` (24)

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 :  $\text{Alphat} = \text{Nut} / \text{Prdt}$  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  $\alpha$  is the thermal diffusivity.
- **turbulence\_paro** *turbulence\_paro\_scalaire\_base* (33) 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 :  $\text{Nu} = ((\lambda + \lambda_t) / \lambda) * d_{\text{wall}} / d_{\text{eq}}$  where  $d_{\text{wall}}$  is the distance from the first mesh to the wall and  $d_{\text{eq}}$  is given by the wall law. This option also gives the value of  $d_{\text{eq}}$  and  $h = (\lambda + \lambda_t) / d_{\text{eq}}$  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».

## 24.2 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` (24)

Usage:

```
schmidt obj Lire obj {
```

```
  [ scturb float]
```

```
  turbulence_paro turbulence_paro_scalaire_base
```

```
  [ dt_impr_nusselt float]
```

```
}
```

where

- **scturb** *float*: Keyword to modify the constant (Sct) of Schmlidt model :  $\text{Dt} = \text{Nut} / \text{Sct}$  Default value is 0.7.
- **turbulence\_paro** *turbulence\_paro\_scalaire\_base* (33) 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 :  $\text{Nu} = ((\lambda + \lambda_t) / \lambda) * d_{\text{wall}} / d_{\text{eq}}$  where  $d_{\text{wall}}$  is the distance from the first mesh to the wall and  $d_{\text{eq}}$  is given by the wall law. This option also gives the value of  $d_{\text{eq}}$  and  $h = (\lambda + \lambda_t) / d_{\text{eq}}$  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».

## 24.3 sous\_maille\_dyn

Description: Dynamic sub-grid turbulence modele.

Warning : Available in VDF only. Not coded in VEF yet.

See also: modele\_turbulence\_scal\_base (24)

Usage:

**sous\_maille\_dyn** obj Lire obj {

[ **stabilise** *str* into ['6\_points', 'moy\_euler', 'plans\_paralleles']]

[ **nb\_points** *int*]

**turbulence\_paro**i *turbulence\_paro\_scalaire\_base*

[ **dt\_impr\_nusselt** *float*]

}

where

- **stabilise** *str* into ['6\_points', 'moy\_euler', 'plans\_paralleles']
- **nb\_points** *int*
- **turbulence\_paro**i *turbulence\_paro\_scalaire\_base* (33) 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».

## 25 nom

Description: Class to name the TRUST objects.

See also: objet\_u (36) nom\_anonyme (25.1)

Usage:

**nom** [ **mot** ]

where

- **mot** *str*: Chain of characters.

### 25.1 nom\_anonyme

Description: not\_set

See also: nom (25)

Usage:

[ **mot** ]

where

- **mot** *str*: Chain of characters.

## 26 partitionneur\_deriv

Description: not\_set

See also: objet\_u (36) metis (26.2) sous\_zones (26.4) tranche (26.5) partition (26.3) fichier\_decoupage (26.1)

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).

### 26.1 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 (26)

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).

### 26.2 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` (26)

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 partitioning 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 partitioning 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).

## 26.3 partition

Synonymous: **decouper**

Description: This algorithm re-use the partition of the domain named DOMAINE\_NAME. It is useful to partition for example a post processing domain. The partition should match with the calculation domain.

See also: `partitionneur_deriv` (26)

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).

## 26.4 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` (26)

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).

## 26.5 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` (26)

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).

## 27 precondition\_base

Description: Basic class for preconditioning.

See also: `objet_u` (36) `ssor` (27.2) `ssor_bloc` (27.3) `precondsolv` (27.1)

Usage:

## 27.1 precondsolv

Description: not\_set

See also: `precond_base` (27)

Usage:

**precondsolv** **solveur**

where

- **solveur** *solveur\_sys\_base* (10.12): Solver type.

## 27.2 ssor

Description: Symmetric successive over-relaxation algorithm.

See also: `precond_base` (27)

Usage:

**ssor** obj Lire obj {

**omega** *float*

}

where

- **omega** *float*: Over-relaxation facteur (between 1 and 2, optimal value around 1.5-1.6).

## 27.3 ssor\_bloc

Description: not\_set

See also: `precond_base` (27)

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* (27)
- **alpha\_1** *float*
- **precond1** *precond\_base* (27)
- **alpha\_a** *float*
- **preconda** *precond\_base* (27)

## 28 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 \(36\)](#) [scheme\\_euler\\_explicit \(28.4\)](#) [schema\\_predictor\\_corrector \(28.19\)](#) [Sch\\_CN\\_iteratif \(28.3\)](#) [runge\\_kutta\\_ordre\\_3 \(28.7\)](#) [runge\\_kutta\\_ordre\\_4\\_d3p \(28.8\)](#) [leap\\_frog \(28.5\)](#) [runge\\_kutta\\_rationnel\\_ordre\\_2 \(28.9\)](#) [schema\\_implicite\\_base \(28.17\)](#) [schema\\_adams\\_bashforth\\_order\\_2 \(28.10\)](#) [schema\\_adams\\_bashforth\\_order\\_3 \(28.11\)](#) [schema\\_phase\\_field \(28.18\)](#)

Usage:

```
schema_temps_base obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max float]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ facsec float]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int]  
    [ diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int]  
    [ no_error_if_not_converged_diffusion_implicite int]  
    [ no_conv_subiteration_diffusion_implicite int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicite int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
}
```

where

- **tinit** *float*: Value of initial calculation time (0 by default).
- **tmax** *float*: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float*: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float*: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float*: Maximum calculation time step (1e30s by default).
- **dt\_sauv** *float*: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float*: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float*: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.

Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.

- **seuil\_statio** *float*: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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*
- **diffusion\_implicit** *int*: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt = facsec * dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt = facsec * dt_{max}$ .
- **seuil\_diffusion\_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*: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int*
- **no\_conv\_subiteration\_diffusion\_implicit** *int*
- **dt\_start** *dt\_start* (10.5):  $dt_{start} dt_{min}$  : the first iteration is based on  $dt_{min}$ .  
 $dt_{start} dt_{calc}$  : the time step at first iteration is calculated in agreement with CFL condition.  
 $dt_{start} dt_{fixe}$  value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
 By default, the first iteration is based on  $dt_{calc}$ .
- **nb\_pas\_dt\_max** *int*: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int*: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** : To disable the writing of the .progress file.
- **disable\_dt\_ev** : To disable the writing of the .dt\_ev file.

## 28.1 implicit\_euler\_steady\_scheme

Synonymous: **schema\_euler\_implicit\_stationnaire**

Description: This is the Implicit Euler scheme using a dual time step procedure (using local and global  $dt$ ) for steady problems. Remark: the only possible solver choice for this scheme is the implicit\_steady solver.

See also: **schema\_implicit\_base** (28.17)

Usage:

**implicit\_euler\_steady\_scheme** obj Lire obj {

[ **max\_iter\_implicit** *int*]  
 [ **steady\_security\_facteur** *float*]

```

[ steady_global_dt float]
solveur solveur_implicite_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
where

```

- **max\_iter\_implicite** *int*: Maximum number of iterations allowed for the solver (by default 200)
- **steady\_security\_facteur** *float*: Parameter used in the local time step calculation procedure in order to increase or decrease the local dt value (by default 0.5). We expect a strictly positive value
- **steady\_global\_dt** *float*: This is the global time step used in the dual time step algorithm (by default 100). We expect a strictly positive value
- **solveur** *solveur\_implicite\_base* (29) 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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every *dt\_sauv*, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not

entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.

- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance:  $dt_{start} dt_{min}$  : the first iteration is based on  $dt_{min}$ .  
 $dt_{start} dt_{calc}$  : the time step at first iteration is calculated in agreement with CFL condition.  
 $dt_{start} dt_{fixe}$  value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on  $dt_{calc}$ .
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.2 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 by default), *niter\_max* (6 by default), *niter\_avg* (3 by default), *facsec\_max* (20 by default), *seuil* (0.05 by default)

See also: Sch\_CN\_iteratif ([28.3](#))

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]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
```

}

where

- **omega** float: relaxation factor (0.1 by default)
- **niter\_min** int for inheritance: minimal number of p-iterations to satisfy convergence criteria (2 by default)
- **niter\_max** int for inheritance: number of maximum p-iterations allowed to satisfy convergence criteria (6 by default)
- **niter\_avg** int for inheritance: threshold of p-iterations (3 by default). If the number of p-iterations is greater than *niter\_avg*, *facsec* is reduced, if lesser than *niter\_avg*, *facsec* is increased (but limited by the *facsec\_max* value).
- **facsec\_max** float for inheritance: maximum ratio allowed between dynamical time step returned by iterative process and stability time returned by CFL condition (2 by default).
- **seuil** float for inheritance: criteria for ending iterative process ( $\text{Max}(\|u(p) - u(p-1)\|/\text{Max} \|u(p)\|) < \text{seuil}$ ) (0.001 by default)
- **tinit** float for inheritance: Value of initial calculation time (0 by default).

- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt\_sauv**, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: **dt\_start dt\_min** : the first iteration is based on dt\_min.  
**dt\_start dt\_calc** : the time step at first iteration is calculated in agreement with CFL condition.  
**dt\_start dt\_fixe** value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.



### 28.3 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` (28) `Sch_CN_EX_iteratif` (28.2)

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]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ facsec float]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int]  
    [ diffusion_implicit int]  
    [ seuil_diffusion_implicit float]  
    [ impr_diffusion_implicit int]  
    [ no_error_if_not_converged_diffusion_implicit int]  
    [ no_conv_subiteration_diffusion_implicit int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicit int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
}
```

where

- **niter\_min** *int*: minimal number of p-iterations to satisfy convergence criteria (2 by default)
- **niter\_max** *int*: number of maximum p-iterations allowed to satisfy convergence criteria (6 by default)

- **niter\_avg** *int*: threshold of p-iterations (3 by default). If the number of p-iterations is greater than niter\_avg, facsec is reduced, if lesser than niter\_avg, facsec is increased (but limited by the facsec\_max value).
- **facsec\_max** *float*: maximum ratio allowed between dynamical time step returned by iterative process and stability time returned by CFL condition (2 by default).
- **seuil** *float*: criteria for ending iterative process ( $\text{Max}(\|u(p) - u(p-1)\| / \text{Max} \|u(p)\|) < \text{seuil}$ ) (0.001 by default)
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt = \text{facsec} * dt_{\text{convection}}$ ). 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 avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt = \text{facsec} * dt_{\text{max}}$ .
- **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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance:  $dt_{\text{start}} dt_{\text{min}}$  : the first iteration is based on  $dt_{\text{min}}$ .  
 $dt_{\text{start}} dt_{\text{calc}}$  : the time step at first iteration is calculated in agreement with CFL condition.  
 $dt_{\text{start}} dt_{\text{fixe}}$  value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on  $dt_{\text{calc}}$ .
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **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.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values

printed into .out files (by default 3).

- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.4 scheme\_euler\_explicit

Synonymous: **schema\_euler\_explicite**

Description: This is the Euler explicit scheme.

See also: [schema\\_temps\\_base \(28\)](#)

Usage:

```
scheme_euler_explicit obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max float]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ facsec float]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int]  
    [ diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int]  
    [ no_error_if_not_converged_diffusion_implicite int]  
    [ no_conv_subiteration_diffusion_implicite int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicite int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
}
```

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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not

entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.

- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance:  $dt_{start} dt_{min}$  : the first iteration is based on  $dt_{min}$ .  
 $dt_{start} dt_{calc}$  : the time step at first iteration is calculated in agreement with CFL condition.  
 $dt_{start} dt_{fixe}$  value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on  $dt_{calc}$ .
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.5 leap\_frog

Description: This is the leap-frog scheme.

See also: [schema\\_temps\\_base](#) (28)

Usage:

```
leap_frog obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max float]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ facsec float]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int]  
    [ diffusion_implicit int]  
    [ seuil_diffusion_implicit float]  
    [ impr_diffusion_implicit int]  
    [ no_error_if_not_converged_diffusion_implicit int]  
    [ no_conv_subiteration_diffusion_implicit int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicit int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
}
```

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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt\_sauv**, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based

on the convection time step ( $dt=facsec*dt\_convection$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt\_max$ .

- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance:  $dt\_start$   $dt\_min$  : the first iteration is based on  $dt\_min$ .  
 $dt\_start$   $dt\_calc$  : the time step at first iteration is calculated in agreement with CFL condition.  
 $dt\_start$   $dt\_fixe$  value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on  $dt\_calc$ .
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.6 rk3\_ft

Description: Keyword for Runge Kutta time scheme for Front\_Tracking calculation.

See also: `runge_kutta_ordre_3` (28.7)

Usage:

```
rk3_ft obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int]
    [ no_error_if_not_converged_diffusion_implicit int]
```

```

[ no_conv_subiteration_diffusion_implicit int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]

```

}

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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: dt\_start dt\_min : the first iteration is based on dt\_min.  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.

- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.7 runge\_kutta\_ordre\_3

Description: This is the Runge-Kutta scheme of third order.

See also: `schema_temps_base` (28) `rk3_ft` (28.6)

Usage:

```
runge_kutta_ordre_3 obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int]
    [ no_error_if_not_converged_diffusion_implicit int]
    [ no_conv_subiteration_diffusion_implicit int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicit int]
    [ precision_impr int]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
    [ disable_progress ]
    [ disable_dt_ev ]
}
```

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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: dt\_start dt\_min : the first iteration is based on dt\_min.  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.8 runge\_kutta\_ordre\_4\_d3p

Description: not\_set

See also: schema\_temps\_base (28)

Usage:

```
runge_kutta_ordre_4_d3p obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max float]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ facsec float]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int]  
    [ diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int]  
    [ no_error_if_not_converged_diffusion_implicite int]  
    [ no_conv_subiteration_diffusion_implicite int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicite int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
}
```

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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported

values  $G_i$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt = facsec * dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large *facsec* value. Start with a *facsec* value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt = facsec * dt_{max}$ .
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value ( $1e-6$ ) of convergence criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: *dt\_start dt\_min* : the first iteration is based on *dt\_min*.  
*dt\_start dt\_calc* : the time step at first iteration is calculated in agreement with CFL condition.  
*dt\_start dt\_fixe* value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on *dt\_calc*.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps ( $1e9$  by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.9 runge\_kutta\_rationnel\_ordre\_2

Description: This is the Runge-Kutta rational scheme of second order. The method is described in the note: Wambeck - Rational Runge-Kutta methods for solving systems of ordinary differential equations, at the link: <https://link.springer.com/article/10.1007/BF02252381>. Although rational methods require more computational work than linear ones, they can have some other properties, such as a stable behaviour with explicitness, which make them preferable. The CFD application of this RRK2 scheme is described in the note: [https://link.springer.com/content/pdf/10.1007%2F3-540-13917-6\\_112.pdf](https://link.springer.com/content/pdf/10.1007%2F3-540-13917-6_112.pdf).

See also: [schema\\_temps\\_base](#) (28)

Usage:

**runge\_kutta\_rationnel\_ordre\_2** obj Lire obj {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
```

```

[ dt_max float]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil\_statio\_relatif\_deconseille** *int* for inheritance
- **diffusion\_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec\*dt\_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec\*dt\_max.

- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: *dt\_start dt\_min* : the first iteration is based on *dt\_min*.  
*dt\_start dt\_calc* : the time step at first iteration is calculated in agreement with CFL condition.  
*dt\_start dt\_fixe* value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on *dt\_calc*.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.10 schema\_adams\_bashforth\_order\_2

Description: not\_set

See also: schema\_temps\_base (28)

Usage:

```
schema_adams_bashforth_order_2 obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int]
    [ no_error_if_not_converged_diffusion_implicit int]
    [ no_conv_subiteration_diffusion_implicit int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicit int]
    [ precision_impr int]
    [ periode_sauvegarde_securite_en_heures int]
```

```

[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt\_sauv**, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt = facsec * dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt = facsec * dt_{max}$ .
- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: **dt\_start dt\_min** : the first iteration is based on **dt\_min**.  
**dt\_start dt\_calc** : the time step at first iteration is calculated in agreement with CFL condition.  
**dt\_start dt\_fixe** value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on **dt\_calc**.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).

- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.11 schema\_adams\_bashforth\_order\_3

Description: not\_set

See also: schema\_temps\_base (28)

Usage:

**schema\_adams\_bashforth\_order\_3** obj Lire obj {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
```

}

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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.



- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance:  $dt_{start} dt_{min}$  : the first iteration is based on  $dt_{min}$ .  
 $dt_{start} dt_{calc}$  : the time step at first iteration is calculated in agreement with CFL condition.  
 $dt_{start} dt_{fixe}$  value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on  $dt_{calc}$ .
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.12 schema\_adams\_moulton\_order\_2

Description: not\_set

See also: schema\_implicit\_base (28.17)

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]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int]
[ no_error_if_not_converged_diffusion_implicit int]
[ no_conv_subiteration_diffusion_implicit int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
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\_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur\_implicit\_base* (29) 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 Implicit (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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **seuil\_diffusion\_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicite** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicite** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: dt\_start dt\_min : the first iteration is based on dt\_min.  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).

- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

### 28.13 schema\_adams\_moulton\_order\_3

Description: not\_set

See also: schema\_implicite\_base ([28.17](#))

Usage:

```

schema_adams_moulton_order_3 obj Lire obj {
    [ facsec_max float]
    [ max_iter_implicite int]
    solveur solveur_implicite_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int]
    [ no_error_if_not_converged_diffusion_implicite int]
    [ no_conv_subiteration_diffusion_implicite int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicite int]
    [ precision_impr int]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
    [ disable_progress ]
    [ disable_dt_ev ]
}

```

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\_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur\_implicit\_base* (29) 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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every *dt\_sauv*, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .

- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: *dt\_start dt\_min* : the first iteration is based on *dt\_min*.  
*dt\_start dt\_calc* : the time step at first iteration is calculated in agreement with CFL condition.  
*dt\_start dt\_fixe* value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on *dt\_calc*.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.14 schema\_backward\_differentiation\_order\_2

Description: not\_set

See also: schema\_implicit\_base (28.17)

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]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int]
    [ no_error_if_not_converged_diffusion_implicit int]
    [ no_conv_subiteration_diffusion_implicit int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
```

```

[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
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\_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur\_implicit\_base* (29) 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 Implicit (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 Implicit 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 Implicit 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 Implicit 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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.

- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance:  $dt_{start} dt_{min}$  : the first iteration is based on  $dt_{min}$ .  
 $dt_{start} dt_{calc}$  : the time step at first iteration is calculated in agreement with CFL condition.  
 $dt_{start} dt_{fixe}$  value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on  $dt_{calc}$ .
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.15 schema\_backward\_differentiation\_order\_3

Description: not\_set

See also: schema\_implicit\_base (28.17)

Usage:

**schema\_backward\_differentiation\_order\_3** obj Lire obj {

[ **facsec\_max** *float*]  
[ **max\_iter\_implicit** *int*]



```

solveur solveur_implicite_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}

```

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* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur\_implicite\_base* (29) 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 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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt\_sauv**, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt = facsec * dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt = facsec * dt_{max}$ .
- **seuil\_diffusion\_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicite** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicite** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: **dt\_start dt\_min** : the first iteration is based on **dt\_min**.  
**dt\_start dt\_calc** : the time step at first iteration is calculated in agreement with CFL condition.  
**dt\_start dt\_fixe** value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on **dt\_calc**.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.16 **schema\_euler\_implicit**

Synonymous: **schema\_euler\_implicite**

Description: This is the Euler implicit scheme.

See also: **schema\_implicite\_base** ([28.17](#))

Usage:

```
schema_euler_implicit obj Lire obj {
    [ facsec_max float]
    [ max_iter_implicite int]
    solveur solveur_implicite_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int]
    [ no_error_if_not_converged_diffusion_implicite int]
    [ no_conv_subiteration_diffusion_implicite int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicite int]
    [ precision_impr int]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
    [ disable_progress ]
    [ disable_dt_ev ]
}
```

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\_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur\_implicit\_base* (29) 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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every *dt\_sauv*, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .

- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: *dt\_start dt\_min* : the first iteration is based on *dt\_min*.  
*dt\_start dt\_calc* : the time step at first iteration is calculated in agreement with CFL condition.  
*dt\_start dt\_fixe* value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on *dt\_calc*.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.17 schema\_implicit\_base

Description: Basic class for implicit time scheme.

See also: [schema\\_temps\\_base \(28\)](#) [scheme\\_euler\\_implicit \(28.16\)](#) [schema\\_adams\\_moulton\\_order\\_2 \(28.12\)](#) [schema\\_adams\\_moulton\\_order\\_3 \(28.13\)](#) [schema\\_backward\\_differentiation\\_order\\_2 \(28.14\)](#) [schema\\_backward\\_differentiation\\_order\\_3 \(28.15\)](#) [implicit\\_euler\\_steady\\_scheme \(28.1\)](#)

Usage:

```
schema_implicit_base obj Lire obj {
    [ max_iter_implicit int]
    solveur solveur_implicit_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int]
    [ no_error_if_not_converged_diffusion_implicit int]
    [ no_conv_subiteration_diffusion_implicit int]
    [ dt_start dt_start]
```

```

[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
where

```

- **max\_iter\_implicite** *int*: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur\_implicite\_base* (29): 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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every *dt\_sauv*, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the *facsec* to 0.5.  
Warning: Some schemes needs a *facsec* lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* for inheritance
- **diffusion\_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large *facsec* value. Start with a *facsec* value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .

- **seuil\_diffusion\_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicite** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicite** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: *dt\_start dt\_min* : the first iteration is based on *dt\_min*.  
*dt\_start dt\_calc* : the time step at first iteration is calculated in agreement with CFL condition.  
*dt\_start dt\_fixe* value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on *dt\_calc*.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.18 schema\_phase\_field

Description: Keyword for the only available Scheme for time discretization of the Phase Field problem.

See also: [schema\\_temps\\_base](#) (28)

Usage:

```
schema_phase_field obj Lire obj {
    [ schema_ch schema_temps_base]
    [ schema_ns schema_temps_base]
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int]
    [ no_error_if_not_converged_diffusion_implicite int]
    [ no_conv_subiteration_diffusion_implicite int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicite int]
```



```

[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
where

```

- **schema\_ch** *schema\_temps\_base* (28): Time scheme for the Cahn-Hilliard equation.
- **schema\_ns** *schema\_temps\_base* (28): Time scheme for the Navier-Stokes equation.
- **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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil\_statio\_relatif\_deconseille** *int* for inheritance
- **diffusion\_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec\*dt\_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec\*dt\_max.
- **seuil\_diffusion\_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* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: dt\_start dt\_min : the first iteration is based on dt\_min.  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).

- **niter\_max\_diffusion\_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.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 28.19 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` ([28](#))

Usage:

```

schema_predictor_corrector obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int]
    [ no_error_if_not_converged_diffusion_implicit int]
    [ no_conv_subiteration_diffusion_implicit int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicit int]
    [ precision_impr int]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
    [ disable_progress ]
    [ disable_dt_ev ]
}
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).
- **dt\_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt\_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil\_statio\_relatif\_deconseille** *int* for inheritance
- **diffusion\_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ( $dt = facsec * dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore  $dt = facsec * dt_{max}$ .
- **seuil\_diffusion\_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no\_error\_if\_not\_converged\_diffusion\_implicite** *int* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicite** *int* for inheritance
- **dt\_start** *dt\_start* (10.5) for inheritance: dt\_start dt\_min : the first iteration is based on dt\_min.  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter\_max\_diffusion\_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode\_sauvegarde\_securite\_en\_heures** *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.
- **disable\_progress** for inheritance: To disable the writing of the .progress file.
- **disable\_dt\_ev** for inheritance: To disable the writing of the .dt\_ev file.

## 29 solveur\_implicite\_base

Description: Class for solver in the situation where the time scheme is the implicit scheme. Solver allows equation diffusion and convection operators to be set as implicit terms.

See also: `objet_u` (36) `solveur_lineaire_std` (29.6) `simpler` (29.5)

Usage:

### 29.1 implicit\_steady

Description: this is the implicit solver using a dual time step. Remark: this solver can be used only with the `Implicit_Euler_Steady_Scheme` time scheme.

See also: `implicite` (29.2)

Usage:

```
implicit_steady obj Lire obj {  
    [ seuil_convergence_implicite float]  
    [ nb_corrections_max int]  
    [ seuil_convergence_solveur float]  
    [ seuil_generation_solveur float]  
    [ seuil_verification_solveur float]  
    [ seuil_test_preliminaire_solveur float]  
    [ solveur solveur_sys_base]  
    [ no_qdm ]  
    [ nb_it_max int]  
    [ controle_residu ]  
}
```

where

- **seuil\_convergence\_implicite** *float* for inheritance: Convergence criteria.
- **nb\_corrections\_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections 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* (10.12) 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.

## 29.2 implicite

Description: similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.

See also: [piso \(29.3\)](#) [implicit\\_steady \(29.1\)](#)

Usage:

```
implicite obj Lire obj {
    [ seuil_convergence_implicite float]
    [ nb_corrections_max int]
    [ seuil_convergence_solveur float]
    [ seuil_generation_solveur float]
    [ seuil_verification_solveur float]
    [ seuil_test_preliminaire_solveur float]
    [ solveur solveur_sys_base]
    [ no_qdm ]
    [ nb_it_max int]
    [ controle_residu ]
}
```

where

- **seuil\_convergence\_implicite** float for inheritance: Convergence criteria.
- **nb\_corrections\_max** int for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections 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 (10.12) 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.

### 29.3 piso

Description: Piso (Pressure Implicit with Split Operator) - method to solve N\_S.

See also: [simpler \(29.5\)](#) [implicite \(29.2\)](#) [simple \(29.4\)](#)

Usage:

```
piso obj Lire obj {  
    [ seuil_convergence_implicite float]  
    [ nb_corrections_max int]  
    [ seuil_convergence_solveur float]  
    [ seuil_generation_solveur float]  
    [ seuil_verification_solveur float]  
    [ seuil_test_preliminaire_solveur float]  
    [ solveur solveur_sys_base]  
    [ no_qdm ]  
    [ nb_it_max int]  
    [ controle_residu ]  
}
```

where

- **seuil\_convergence\_implicite** *float*: Convergence criteria.
- **nb\_corrections\_max** *int*: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections 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* ([10.12](#)) 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.

### 29.4 simple

Description: SIMPLE type algorithm

See also: [piso \(29.3\)](#)

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 SIM-  
PLE algorithm for relaxing the increment of pressure.
- **seuil\_convergence\_implicite** *float* for inheritance: Convergence criteria.
- **nb\_corrections\_max** *int* for inheritance: Maximum number of corrections performed by the PISO  
algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections  
then nb\_corrections\_max if the accuracy of the projection is sufficient. (By default nb\_corrections-  
\_max is set to 21).
- **seuil\_convergence\_solveur** *float* for inheritance: value of the convergence criteria for the resolution  
of the implicit system build by solving several times per time step the Navier\_Stokes equation and the  
scalar equations if any. This value **MUST** be used when a coupling between problems is considered  
(should be set to a value typically of 0.1 or 0.01).
- **seuil\_generation\_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as  
the convergence threshold (implicit linear system  $Ax=B$  will be solved if residual error  $\|Ax-B\|$  is  
lesser than vrel).
- **seuil\_verification\_solveur** *float* for inheritance: Option to check if residual error  $\|Ax-B\|$  is lesser  
than vrel after the implicit linear system  $Ax=B$  has been solved.
- **seuil\_test\_preliminaire\_solveur** *float* for inheritance: Option to decide if the implicit linear system  
 $Ax=B$  should be solved by checking if the residual error  $\|Ax-B\|$  is bigger than vrel.
- **solveur** *solveur\_sys\_base* (10.12) 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 conver-  
gence occurs if the residu suddenly increases.

## 29.5 simplifier

Description: Simpler method for incompressible systems.

See also: solveur\_implicite\_base (29) piso (29.3)

Usage:

```

simplifier 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\_implicit** *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* (10.12): 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.

## 29.6 solveur\_lineaire\_std

Description: not\_set

See also: solveur\_implicit\_base (29)

Usage:

```

solveur_lineaire_std obj Lire obj {
    [ solveur solveur_sys_base]
}
where

```

- **solveur** *solveur\_sys\_base* (10.12)

## 30 source\_base

Description: Basic class of source terms introduced in the equation.

See also: objet\_u (36) source\_generique (30.21) boussinesq\_temperature (30.4) boussinesq\_concentration (30.3) dirac (30.8) puissance\_thermique (30.17) source\_qdm\_lambdaup (30.24) source\_th\_tdivu (30.30)

source\_robin (30.27) source\_robin\_scalaire (30.28) canal\_perio (30.5) source\_constituant (30.19) source\_transport\_k\_eps (30.32) acceleration (30.2) coriolis (30.6) source\_qdm (30.23) perte\_charge\_singuliere (30.16) perte\_charge\_directionnelle (30.12) perte\_charge\_isotrope (30.13) perte\_charge\_anisotrope (30.10) perte\_charge\_circulaire (30.11) darcy (30.7) forchheimer (30.9) perte\_charge\_reguliere (30.14) trainee (30.31) flottabilite (30.20) masse\_ajoutee (30.22) source\_qdm\_phase\_field (30.25) source\_con\_phase\_field (30.18) source\_rayo\_semi\_transp (30.26)

Usage:

### 30.1 Source\_Transport\_K\_Eps\_anisotherme

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transport 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 (30.32)

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.

### 30.2 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 (30)

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* (16.1): 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* (16.1): 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* (16.1): Keyword for a rotation of the referential R' into the R referential [rad.s-1]. *field\_base* is a 3D time dependant field specified for example by a *Champ\_Fonc\_t* keyword. The *time\_field* field should have 3 components even in 2D (In 2D: 0 0 omega).
- **domegadt** *champ\_base* (16.1): Keyword to define the time derivative of the previous rotation [rad.s-2]. Should be zero if the rotation is constant. The *time\_field* field should have 3 components even in 2D (In 2D: 0 0 domegadt).
- **centre\_rotation** *champ\_base* (16.1): Keyword to specify the centre of rotation (expressed in R' coordinates) of R' into R (if the domain rotates with the R' referential, the centre of rotation is 0'=(0,0,0)). The *time\_field* should have 2 or 3 components according the dimension 2 or 3.
- **option** *str* into ['terme\_complet', 'coriolis\_seul', 'entrainement\_seul']: Keyword to specify the kind of calculation: *terme\_complet* (default option) will calculate both the Coriolis and centrifugal forces, *coriolis\_seul* will calculate the first one only, *entrainement\_seul* will calculate the second one only.

### 30.3 boussinesq\_concentration

Description: Class to describe a source term that couples the movement quantity equation and constituent transport equation with the Boussinesq hypothesis.

See also: *source\_base* (30)

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.

### 30.4 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* (30)

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.



### 30.5 canal\_perio

Description: Momentum source term to maintain flow rate. The expression of the source term is:

$$S(t) = (2*(Q(0) - Q(t)) - (Q(0) - Q(t-dt)))/(coeff*dt*area)$$

Where:

coeff=damping coefficient

area=area of the periodic boundary

Q(t)=flow rate at time t

dt=time step

Three files will be created during calculation on a datafile named DataFile.data. The first file contains the flow rate evolution. The second file is useful for resuming a calculation with the flow rate of the previous stopped calculation, and the last one contains the pressure gradient evolution:

-DataFile\_Channel\_Flow\_Rate\_ProblemName\_BoundaryName

-DataFile\_Channel\_Flow\_Rate\_repr\_ProblemName\_BoundaryName

-DataFile\_Pressure\_Gradient\_ProblemName\_BoundaryName

See also: [source\\_base \(30\)](#)

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.

### 30.6 coriolis

Description: Keyword for a Coriolis term in hydraulic equation. Warning: Only available in VDF.

See also: [source\\_base \(30\)](#)

Usage:

**coriolis** **omega**

where

- **omega** *str*: Value of omega.

### 30.7 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` (30)

Usage:

**darcy bloc**

where

- **bloc** *bloc\_lecture* (3.44): Description.

### 30.8 dirac

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: `source_base` (30)

Usage:

**dirac position ch**

where

- **position** *n x1 x2 ... xn*
- **ch** *champ\_base* (16.1): Thermal power field type. To impose a volume power on a domain sub-area, the `Champ_Uniforme_Morceaux` (`partly_uniform_field`) type must be used.  
Warning : The volume thermal power is expressed in W.m<sup>-3</sup>.

### 30.9 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` (30)

Usage:

**forchheimer bloc**

where

- **bloc** *bloc\_lecture* (3.44): Description.

### 30.10 perte\_charge\_anisotrope

Description: Anisotropic pressure loss.

See also: `source_base` (30)

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* (16.3): Hydraulic diameter value.
- **direction** *champ\_don\_base* (16.3): Field which indicates the direction of the pressure loss.
- **sous\_zone** *str*: Optional sub-area where pressure loss applies.

### 30.11 perte\_charge\_circulaire

Description: New pressure loss.

See also: [source\\_base \(30\)](#)

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(Re\_tot, Re\_long, t, x, y, z) for loss coefficient in the longitudinal direction
- **lambda\_ortho** *str*: function: Function f(Re\_tot, Re\_ortho, t, x, y, z) for loss coefficient in transverse direction
- **diam\_hydr** *champ\_don\_base* (16.3): Hydraulic diameter value.
- **diam\_hydr\_ortho** *champ\_don\_base* (16.3): Transverse hydraulic diameter value.
- **direction** *champ\_don\_base* (16.3): Field which indicates the direction of the pressure loss.
- **sous\_zone** *str*: Optional sub-area where pressure loss applies.

### 30.12 perte\_charge\_directionnelle

Description: Directional pressure loss.

See also: [source\\_base \(30\)](#)

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/Re).
- **diam\_hydr** *champ\_don\_base* (16.3): Hydraulic diameter value.
- **direction** *champ\_don\_base* (16.3): Field which indicates the direction of the pressure loss.
- **sous\_zone** *str*: Optional sub-area where pressure loss applies.

### 30.13 perte\_charge\_isotrope

Description: Isotropic pressure loss.

See also: [source\\_base](#) (30)

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* (16.3): Hydraulic diameter value.
- **sous\_zone** *str*: Optional sub-area where pressure loss applies.

### 30.14 perte\_charge\_reguliere

Description: Source term modelling the presence of a bundle of tubes in a flow.

See also: [source\\_base](#) (30)

Usage:

```
perte_charge_reguliere spec zone_name  
where
```

- **spec** *spec\_pdc\_base* (30.15): 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.

### 30.15 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](#) (35) longitudinale (30.15.1) transversale (30.15.2)

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.

### 30.15.1 longitudinale

Description: Class to define the pressure loss in the direction of the tube bundle.

See also: [spec\\_pdcr\\_base \(30.15\)](#)

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.

### 30.15.2 transversale

Description: Class to define the pressure loss in the direction perpendicular to the tube bundle.

See also: [spec\\_pdcr\\_base \(30.15\)](#)

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.

## 30.16 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 \(30\)](#)

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* (3.44): Two syntaxes are possible for the surface definition block:  
For VDF and VEF: { X|Y|Z = location subzone\_name }  
Only for VEF: { Surface surface\_name }.

### 30.17 puissance\_thermique

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: [source\\_base \(30\)](#)

Usage:

**puissance\_thermique ch**

where

- **ch** *champ\_base* (16.1): Thermal power field type. To impose a volume power on a domain sub-area, the Champ\_Uniforme\_Morceaux (partly\_uniform\_field) type must be used.  
Warning : The volume thermal power is expressed in W.m-3 in 3D. It is a power per volume unit (in a porous media, it is a power per fluid volume unit).

### 30.18 source\_con\_phase\_field

Description: Keyword to define the source term of the Cahn-Hilliard equation.

See also: [source\\_base \(30\)](#)

Usage:

**source\_con\_phase\_field** obj Lire obj {

```
    temps_d_affichage int
    alpha float
    beta float
    kappa float
    kappa_variable str into ['oui', 'non']
    moyenne_de_kappa str
    multiplicateur_de_kappa float
    couplage_NS_CH str
    implicitation_CH str into ['oui', 'non']
    gmres_non_lineaire str into ['oui', 'non']
    seuil_cv_iterations_ptfixe float
    seuil_residu_ptfixe float
    seuil_residu_gmresnl float
    dimension_espace_de_krylov int
    nb_iterations_gmresnl int
    residu_min_gmresnl float
    residu_max_gmresnl float
```

}  
where

- **temps\_d\_affichage** *int*: Time during the characteristics of the problem are shown before calculation.
- **alpha** *float*: Internal capillary coefficient  $\alpha$ .
- **beta** *float*: Parameter  $\beta$  of the model.
- **kappa** *float*: Mobility coefficient  $\kappa_0$ .
- **kappa\_variable** *str* into ['oui', 'non']: To define a mobility which depends on concentration  $C$ .
- **moyenne\_de\_kappa** *str*: To define how mobility  $\kappa$  is calculated on faces of the mesh according to cell-centered values (chaîne is arithmetique/harmonique/geometrique).
- **multiplieur\_de\_kappa** *float*: To define the parameter of the mobility expression when mobility depends on  $C$ .
- **couplage\_NS\_CH** *str*: Evaluating time chosen for the term source calculation into the Navier Stokes equation (chaîne is  $\tilde{n}+1/2/\tilde{n}$ ), in order to be conservative, the first choice seems better).
- **implication\_CH** *str* into ['oui', 'non']: To define if the Cahn-Hilliard will be solved using a implicit algorithm or not.
- **gmres\_non\_lineaire** *str* into ['oui', 'non']: To define the algorithm to solve Cahn-Hilliard equation (oui: Newton-Krylov method, non: fixed point method).
- **seuil\_cv\_iterations\_ptfixe** *float*: Convergence threshold (an option of the fixed point method).
- **seuil\_residu\_ptfixe** *float*: Threshold for the matrix inversion used in the method (an option of the fixed point method).
- **seuil\_residu\_gmresnl** *float*: Convergence threshold (an option of the Newton-Krylov method).
- **dimension\_espace\_de\_krylov** *int*: Vector numbers used in the method (an option of the Newton-Krylov method).
- **nb\_iterations\_gmresnl** *int*: Maximal iteration (an option of the Newton-Krylov method).
- **residu\_min\_gmresnl** *float*: Minimal convergence threshold (an option of the Newton-Krylov method).
- **residu\_max\_gmresnl** *float*: Maximal convergence threshold (an option of the Newton-Krylov method).

### 30.19 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 \(30\)](#)

Usage:

**source\_constituant** **ch**

where

- **ch** *champ\_base* (16.1): Field type.

### 30.20 flottabilite

Description: buoyancy effect

See also: [source\\_base \(30\)](#)

Usage:

**flottabilite**

### 30.21 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 \(30\)](#)

Usage:

**source\_generique champ**

where

- **champ** *champ\_generique\_base* (8): the source field

### 30.22 masse\_ajoutee

Description: weight added effect

See also: [source\\_base \(30\)](#)

Usage:

**masse\_ajoutee**

### 30.23 source\_qdm

Description: Momentum source term in the Navier-Stokes equations.

See also: [source\\_base \(30\)](#)

Usage:

**source\_qdm ch**

where

- **ch** *champ\_base* (16.1): Field type.

### 30.24 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 \(30\)](#)

Usage:

**source\_qdm\_lambdaup** obj Lire obj {

```
    lambda float  
    [ lambda_min float]  
    [ lambda_max float]  
    [ ubar_umprim_cible float]
```

}

where



- **lambda** *float*: value of lambda
- **lambda\_min** *float*: value of lambda\_min
- **lambda\_max** *float*: value of lambda\_max
- **ubar\_umprim\_cible** *float*: value of ubar\_umprim\_cible

### 30.25 source\_qdm\_phase\_field

Description: Keyword to define the capillary force into the Navier Stokes equation for the Phase Field problem.

See also: [source\\_base \(30\)](#)

Usage:

```
source_qdm_phase_field obj Lire obj {
    forme_du_terme_source int
}
```

where

- **forme\_du\_terme\_source** *int*: Kind of the source term (1, 2, 3 or 4).

### 30.26 source\_rayo\_semi\_transp

Description: Radiative term source in energy equation.

See also: [source\\_base \(30\)](#)

Usage:

```
source_rayo_semi_transp
```

### 30.27 source\_robin

Description: This source term should be used when a Paroi\_decalee\_Robin boundary condition is set in a hydraulic equation. The source term will be applied on the N specified boundaries. To post-process the values of tauw, u\_tau and Reynolds\_tau into the files tauw\_robin.dat, reynolds\_tau\_robin.dat and u\_tau\_robin.dat, you must add a block Traitement\_particulier { canal { } }

See also: [source\\_base \(30\)](#)

Usage:

```
source_robin bords
where
```

- **bords** *vect\_nom* [\(3.108\)](#)

### 30.28 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 Ith boundary. The last value dt\_impr is a printing period which is mandatory to specify in the data file but has no effect yet.

See also: [source\\_base \(30\)](#)

Usage:

**source\_robin\_scalaire bords**  
where

- **bords** *listdeuxmots\_sacc* ([30.29](#))

### 30.29 listdeuxmots\_sacc

Description: List of groups of two words (without curly brackets).

See also: [listobj \(34.3\)](#)

Usage:

n object1 object2 ....  
list of *deuxmots* ([5.26](#))

### 30.30 source\_th\_tdivu

Description: This term source is dedicated for any scalar (called T) transport. Coupled with upwind (amont) or muscl scheme, this term gives for final expression of convection :  $\text{div}(U.T)-T.\text{div}(U)=U.\text{grad}(T)$  This ensures, in incompressible flow when divergence free is badly resolved, to stay in a better way in the physical boundaries.

Warning: Only available in VEF discretization.

See also: [source\\_base \(30\)](#)

Usage:

**source\_th\_tdivu**

### 30.31 trainee

Description: drag effect

See also: [source\\_base \(30\)](#)

Usage:

**trainee**

### 30.32 source\_transport\_k\_eps

Description: Keyword to alter the source term constants in the standard k-eps model epsilon transport equation. By default, these constants are set to: C1\_eps=1.44 C2\_eps=1.92

See also: [source\\_base \(30\)](#) [Source\\_Transport\\_K\\_Eps\\_anisotherme \(30.1\)](#) [source\\_transport\\_k\\_eps\\_aniso-concen \(30.33\)](#) [source\\_transport\\_k\\_eps\\_aniso\\_therm\\_concen \(30.34\)](#)

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.

### 30.33 source\_transport\_k\_eps\_aniso\_concen

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transport 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 ([30.32](#))

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.

### 30.34 source\_transport\_k\_eps\_aniso\_therm\_concen

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transport 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 ([30.32](#))

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.

## 31 sous\_zone

Description: It is an object type describing a domain sub-set.

A Sous\_Zone (Sub-area) type object must be associated with a Domaine type object. The Read (Lire) 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 Associate (Associer) `nom_sous_zone nom_domaine` instruction; this instruction must always be preceded by the read instruction.

See also: `objet_u` (36)

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 Read keyword.
- **rectangle** *bloc\_origine\_cotes* (31.1): The sub-area will include all the domain elements whose centre of gravity is within the Rectangle (in dimension 2).
- **segment** *bloc\_origine\_cotes* (31.1)
- **boite** *bloc\_origine\_cotes* (31.1): The sub-area will include all the domain elements whose centre of gravity is within the Box (in dimension 3).
- **liste** *n n1 n2 ... nn*: The sub-area will include *n* domain items, numbers No. 1 No. *i* No. *n*.
- **fichier** *str*: The sub-area is read into the file filename.
- **intervalle** *deuxentiers* (5.25.11): The sub-area will include domain items whose number is between *n1* and *n2* (where *n1* ≤ *n2*).
- **polynomes** *bloc\_lecture* (3.44): A REPENDRE
- **couronne** *bloc\_couronne* (31.2): In 2D case, to create a couronne.
- **tube** *bloc\_tube* (31.3): 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 Read keyword.

### 31.1 bloc\_origine\_cotes

Description: Class to create a rectangle (or a box).

See also: `objet_lecture` (35)

Usage:

```
name origin name2 cotes
where
```

- **name** *str into ['Origine']*: Keyword to define the origin of the rectangle (or the box).

- **origin** *x1 x2 (x3)*: Coordinates of the origin of the rectangle (or the box).
- **name2** *str into ['Cotes']*: Keyword to define the length along the axes.
- **cotes** *x1 x2 (x3)*: Length along the axes.

### 31.2 bloc\_couronne

Description: Class to create a couronne (2D).

See also: [objet\\_lecture \(35\)](#)

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.

### 31.3 bloc\_tube

Description: Class to create a tube (3D).

See also: [objet\\_lecture \(35\)](#)

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 height of the tube.
- **h** *float*: Height of the tube.

## 32 turbulence\_paroι\_base

Description: Basic class for wall laws for Navier-Stokes equations.

See also: [objet\\_u \(36\)](#) [loi\\_standard\\_hydr\\_old \(32.5\)](#) [loi\\_standard\\_hydr \(32.4\)](#) [paroi\\_tble \(32.8\)](#) [negligeable \(32.7\)](#) [utau\\_imp \(32.12\)](#) [loi\\_puissance\\_hydr \(32.3\)](#)

Usage:

### 32.1 loi\_ciofalo\_hydr

Description: A Loi\_ciofalo\_hydr law for wall turbulence for NAVIER STOKES equations.

See also: loi\_standard\_hydr (32.4)

Usage:

**loi\_ciofalo\_hydr**

### 32.2 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 (32.4)

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)

### 32.3 loi\_puissance\_hydr

Description: A Loi\_puissance\_hydr law for wall turbulence for NAVIER STOKES equations.

See also: turbulence\_paro\_base (32)

Usage:

### 32.4 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 (32) loi\_expert\_hydr (32.2) loi\_ww\_hydr (32.6) loi\_ciofalo\_hydr (32.1)

Usage:

**loi\_standard\_hydr**

### 32.5 loi\_standard\_hydr\_old

Description: not\_set

See also: turbulence\_paro\_base (32)

Usage:

**loi\_standard\_hydr\_old**

### 32.6 loi\_ww\_hydr

Description: laws have been qualified on channel calculation

See also: loi\_standard\_hydr (32.4)

Usage:

### 32.7 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 ( $\tau_{\text{tan}}/\rho = \nu \, dU/dy$ ).

Warning: This keyword is not available for k-epsilon models. In that case you must choose a wall law.

See also: turbulence\_paro\_base (32)

Usage:

**negligeable**

### 32.8 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 (32)

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* (32.9): 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* (32.10)
- **restart**
- **stationnaire** *entierfloat* (32.11)
- **lambda** *str*
- **mu** *str*
- **sans\_source\_boussinesq**
- **alpha** *float*
- **kappa** *float*

## 32.9 twofloat

Description: two reals.

See also: [objet\\_lecture \(35\)](#)

Usage:

```

a b
where

```

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

## 32.10 liste\_sonde\_tble

Description: not\_set

See also: [listobj \(34.3\)](#)

Usage:

```

n object1 object2 ....
list of sonde_tble (32.10.1)

```

### 32.10.1 sonde\_tble

Description: not\_set

See also: [objet\\_lecture \(35\)](#)

Usage:

```

name point
where

```



- **name** *str*
- **point** *un\_point* ([3.12.3](#))

### 32.11 entierfloat

Description: An integer and a real.

See also: `objet_lecture` ([35](#))

Usage:

**the\_int the\_float**

where

- **the\_int** *int*: Integer.
- **the\_float** *float*: Real.

### 32.12 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` ([32](#))

Usage:

**utau\_imp** obj Lire obj {

    [ **u\_tau** *champ\_base*]

    [ **lambda\_c** *str*]

    [ **diam\_hydr** *champ\_base*]

}

where

- **u\_tau** *champ\_base* ([16.1](#)): 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* ([16.1](#)): The hydraulic diameter.

## 33 turbulence\_paro\_scalaire\_base

Description: Basic class for wall laws for energy equation.

See also: `objet_u` ([36](#)) `loi_standard_hydr_scalaire` ([33.6](#)) `loi_analytique_scalaire` ([33.2](#)) `paroi_tble_scal` ([33.8](#)) `loi_paro_nu_impose` ([33.5](#)) `negligeable_scalaire` ([33.7](#)) `loi_odvm` ([33.4](#)) `loi_WW_scalaire` ([33.1](#))

Usage:

### 33.1 loi\_WW\_scalaire

Description: not\_set

See also: turbulence\_paro\_scalaire\_base (33)

Usage:

**loi\_WW\_scalaire**

### 33.2 loi\_analytique\_scalaire

Description: not\_set

See also: turbulence\_paro\_scalaire\_base (33)

Usage:

**loi\_analytique\_scalaire**

### 33.3 loi\_expert\_scalaire

Description: Keyword similar to keyword Loi\_standard\_hydr\_scalaire but with additional option.

See also: loi\_standard\_hydr\_scalaire (33.6)

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.

### 33.4 loi\_odvm

Description: Thermal wall-function based on the simultaneous 1D resolution of a turbulent thermal boundary-layer and a variance transport equation, adapted to conjugate heat-transfer problems with fluid/solid thermal interaction (where a specific boundary condition should be used : Paroi\_Echange\_Contact\_OVDM\_VDF). This law is also available with isothermal walls.

See also: turbulence\_paro\_scalaire\_base (33)

Usage:

```
loi_odvm obj Lire obj {  
    n int  
    gamma float  
    [ stats floatfloat ]  
    [ check_files ]
```

}  
where

- **n** *int*: Number of points per face in the 1D uniform meshes. n should be chosen in order to have the first point situated near  $\Delta y = 1/3$ .
- **gamma** *float*: Smoothing parameter of the signal between  $10e-5$  (no smoothing) and  $10e-1$  (high averaging).
- **stats** *floatfloat* (5.27): value\_t0 value\_dt : Only for plane channel flow, it gives mean and root mean square profiles in the fine meshes, since value\_t0 and every value\_dt seconds. The values are printed into files named ODVM\_fields\*.dat.
- **check\_files** : It gives for one boundary face a historical view of local instantaneous and filtered values, as well as the calculated variance profiles from the resolution of the equation. The printed values are into the file Suivi\_nde.dat.

### 33.5 loi\_paro\_i\_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\_paro\_i\_scalaire\_base (33)

Usage:

```
loi_paro_i_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* (16.1): The hydraulic diameter.

### 33.6 loi\_standard\_hydr\_scalaire

Description: Keyword for the law of the wall.

See also: turbulence\_paro\_i\_scalaire\_base (33) loi\_expert\_scalaire (33.3)

Usage:

```
loi_standard_hydr_scalaire
```

### 33.7 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\_paro\_i\_scalaire\_base (33)

Usage:

```
negligeable_scalaire
```

### 33.8 paroi\_tble\_scal

Description: Keyword for the Thin Boundary Layer Equation thermal wall-model.

See also: turbulence\_paro\_scalaire\_base ([33](#))

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* ([33.9](#)): 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* ([32.10](#))
- **prandtl** *float*

### 33.9 fourfloat

Description: Four reals.

See also: objet\_lecture ([35](#))

Usage:

```
a b c d  
where
```

- **a** *float*: First real.
- **b** *float*: Second real.
- **c** *float*: Third real.
- **d** *float*: Fourth real.

## 34 listobj\_impl

Description: not\_set

See also: objet\_u ([36](#)) listobj ([34.3](#))

Usage:

### 34.1 list\_un\_pb

Description: pour les groupes

See also: listobj (34.3)

Usage:

{ object1 , object2 .... }  
list of *un\_pb* (34.2) separated with ,

### 34.2 un\_pb

Description: pour les groupes

See also: objet\_lecture (35)

Usage:

**mot**

where

- **mot** *str*: the string

### 34.3 listobj

Description: List of objects.

See also: listobj\_impl (34) champs\_a\_post (4.2.18) list\_stat\_post (4.2.21) listpoints (4.2.7) sondes (4.2.3) listchamp\_generique (8.3) list\_nom\_virgule (8.2) definition\_champs (4.2.1) post\_processings (4.3) liste\_post (4.5) liste\_post\_ok (4.4) condlims (4.10.1) sources (5.4) vect\_nom (3.108) list\_nom (3.93) list\_bord (3.54.4) list\_bloc\_mailler (3.54) list\_un\_pb (34.1) list\_list\_nom (4.8) ecrire\_fichier\_xyz\_valeur\_param (5.5) pp (5.18) listdeuxmots\_sacc (30.29) liste\_sonde\_tble (32.10) listeqn (4.12) list\_info\_med (4.38) listsous\_zone\_valeur (5.8.12) reactions (9.1)

Usage:

## 35 objet\_lecture

Description: Auxiliary class for reading.

See also: objet\_u (36) bloc\_lecture (3.44) deuxmots (5.26) format\_file (4.6) deuxentiers (5.25.11) floatfloat (5.27) entierfloat (32.11) champ\_a\_post (4.2.19) champs\_posts (4.2.17) stat\_post\_deriv (4.2.22) stats\_posts (4.2.20) stats\_serie\_posts (4.2.28) sonde\_base (4.2.5) un\_point (3.12.3) sonde (4.2.4) definition\_champ (4.2.2) postraitement\_base (4.4.2) un\_postraitement (4.3.1) type\_un\_post (4.5.2) type\_postraitement\_ft\_lata (4.5.3) un\_postraitement\_spec (4.5.1) nom\_postraitement (4.4.1) condinit (5.3.1) condinits (5.3) condlimlu (4.10.2) mailler\_base (3.54.1) bloc\_pave (3.54.3) defbord (3.54.7) bord\_base (3.54.5) parametre\_equation\_base (5.6) un\_pb (34.2) bords\_ecrire (5.5.2) ecrire\_fichier\_xyz\_valeur\_item (5.5.1) convection\_deriv (5.8.1) bloc\_convection (5.8) diffusion\_deriv (5.2.1) op\_implicite (5.2.9) bloc\_diffusion (5.2) traitement\_particulier\_base (5.28.1) traitement\_particulier (5.28) penalisation\_l2\_ftd\_lec (5.18.1) dt\_impr\_ustar\_mean\_only (5.25.1) modele\_turbulence\_hyd\_deriv (5.25) paroi\_ft\_disc\_deriv (12.61) bloc\_sutherland (21.5) form\_a\_nb\_points (5.25.4) modele\_fonction\_has\_reynolds\_base (5.25.21) fourfloat (33.9) twofloat (32.9) sonde\_tble (32.10.1) remove\_elem\_bloc (3.81) lecture\_bloc\_moment\_base (3.12) bloc\_origine\_cotes (31.1) bloc\_couronne (31.2) bloc\_tube (31.3) verifiercoin\_bloc (3.111) bloc\_lecture\_poro (3.65) bloc\_lec\_champ\_init\_canal\_sinal (16.13)

fonction\_champ\_reprise (16.9) bloc\_decouper (3.62) troisi (3.38) spec\_pdc\_base (30.15) format\_lata\_to-  
\_med (3.50) info\_med (4.38.1) methode\_transport\_deriv (5.35) bloc\_ef (5.8.9) sous\_zone\_valeur (5.8.13)  
bloc\_diffusion\_standard (5.2.7) reaction (9.1.1) bloc\_lecture\_remaillage (5.36) objet\_lecture\_maintien-  
\_temperature (5.20) interpolation\_champ\_face\_deriv (5.38) parcours\_interface (5.37) injection\_marqueur  
(5.41) penalisation\_forage (5.24) floatentier (5.25.12) eq\_rayo\_semi\_transp (4.10) ceg\_cea\_jaea (5.28.12)  
ceg\_areva (5.28.11)

Usage:

## 36 index

## Index

/\*, 184  
#, 204  
  
, 108, 111, 114, 160  
associer, 17  
champ\_post\_statistiques\_correlation, 72, 187  
champ\_post\_statistiques\_ecart\_type, 71, 188  
champ\_post\_statistiques\_moyenne, 71, 191  
champ\_uniforme, 237  
decouper, 42, 261  
discretiser, 24  
divergence, 188  
ecrire\_fichier, 62  
extraction, 189  
fin, 31  
gradient, 189  
interpolation, 190  
lire, 47  
lire\_fichier, 48  
lire\_fichier\_bin, 48  
lire\_med, 16  
morceau\_equation, 191  
operateur\_eqn, 186  
postraitement, 74  
postraitements, 73  
raffiner\_simplexes, 46  
rectify\_mesh, 49  
reduction\_0d, 192  
refchamp, 193  
resoudre, 53  
schema\_euler\_explicite, 272  
schema\_euler\_implicite, 295  
schema\_euler\_implicite\_stationnaire, 265  
tparoi\_vef, 194  
transformation, 194  
6\_points, 154, 259  
≤, 37  
=, 37  
A, 208  
a, 314  
amont, 118  
analytique, 174, 176  
ancien, 121, 122  
antisym, 116, 117  
arrete, 140–155  
avec\_energie\_cinetique, 128, 129  
avec\_les\_cl, 135, 137, 165, 167, 169–172  
avec\_sources, 135, 137, 165, 167, 169–172  
avec\_sources\_et\_operateurs, 135, 137, 165, 167, 169–172  
average, 192, 193  
  
b, 314  
binaire, 24, 69, 76, 231  
bords, 112  
C, 254  
C\_ext, 209, 211, 212  
centre, 118  
cf, 314  
chakravarthy, 118  
champ\_frontiere, 189  
chsom, 65  
composante, 194, 195  
conservation\_masse, 253  
constant, 253  
coriolis\_seul, 308, 309  
Cotes, 322  
d, 314  
debit\_total, 33  
default, 190  
defaut\_bar, 109, 116  
dir, 322  
distant, 38  
divrhouT\_moins\_Tdivrhou, 121, 122  
divuT\_moins\_Tdivu, 121, 122  
dt\_integr, 72  
dt\_post, 69, 70  
edo, 253  
elem, 40, 41, 69, 71, 72, 230  
emissivite, 208  
entrainement\_seul, 308, 309  
euclidian\_norm, 192, 193  
faces, 69, 71, 72  
family\_names\_from\_group\_names, 16, 17  
filtrer\_resu, 110, 116, 117  
Fluctu\_Temperature\_ext, 209, 211, 212  
flux\_bords, 191  
Flux\_Chaleur\_Turb\_ext, 209, 211, 212  
fonction, 231  
format\_post\_sup, 34  
formatte, 24, 69, 76, 231  
formule, 194, 195  
grad\_i, 135, 136  
grad\_Ubar, 110  
grav, 65  
hauteur, 322  
homogene, 38  
implicite, 110  
initiale, 174, 177  
integrale\_en\_z, 33  
k, 224  
K\_Eps\_ext, 209, 211, 212

kx , 315  
 ky , 315  
 kz , 315  
 L1\_norm , 192, 193  
 L2\_norm , 192, 193  
 last\_time , 230  
 lata , 34, 45, 64, 74  
 lata\_v1 , 34, 45, 64, 74  
 lata\_v2 , 34, 45, 64, 74  
 left\_value , 192, 193  
 lml , 34, 45, 64, 74  
 local , 38  
 max , 192, 193  
 med , 34, 45, 64, 74  
 med\_major , 64, 74  
 min , 192, 193  
 minmod , 118  
 modifiee , 174, 177  
 moins\_rho\_moyen , 253  
 moy\_euler , 154, 259  
 moyenne , 192, 193  
 moyenne\_ponderee , 192, 193  
 mu0 , 254  
 muscl , 118  
 nb\_pas\_dt\_post , 69, 70  
 no , 181, 190  
 nodes , 65  
 non , 41, 42, 165, 315, 316  
 normalized\_euclidian\_norm , 192, 193  
 norme , 194, 195  
 nu , 110  
 nu\_transp , 110  
 nut , 110  
 nut\_transp , 110  
 one\_way\_coupling , 182, 183  
 Origine , 321, 322  
 oui , 41, 42, 165, 315, 316  
 periode , 65  
 plans\_paralleles , 154, 259  
 post\_processing , 75  
 postraitement , 75  
 postraitement\_ft\_lata , 75  
 postraitement\_lata , 75  
 produit\_scalaire , 194, 195  
 que\_les\_faces\_des\_elts\_dirichlet , 323  
 re , 322  
 rho\_g , 135, 136  
 ri , 322  
 sans\_energie\_cinetique , 128, 129  
 sans\_rien , 135, 137, 165, 167, 169–172  
 scotti , 140–155  
 short\_family\_names , 17  
 simplifiee , 174, 177  
 Slambda , 254  
 solveur , 110  
 som , 40, 41, 65, 69, 71, 72, 230  
 somme , 192, 193  
 somme\_ponderee , 192, 193  
 somme\_ponderee\_porosite , 192, 193  
 stabilite , 191  
 standard , 253  
 suivi , 182, 183  
 sum , 192, 193  
 superbeee , 118  
 T0 , 254  
 T\_ext , 209, 211, 212  
 terme\_complet , 308, 309  
 toutes\_les\_faces\_accrochees , 323  
 trace , 189  
 transportant\_bar , 116, 117  
 transporte\_bar , 116, 117  
 two\_way\_coupling , 182, 183  
 uniforme , 174, 176  
 use\_existing\_domain , 230  
 V2\_ext , 209, 211, 212  
 valeur\_a\_elem , 174, 176  
 valeur\_a\_gauche , 192, 193  
 valeur\_normale , 245  
 vanalbada , 118  
 vanleer , 118  
 vdf\_lineaire , 174, 176  
 vecteur , 194, 195  
 vef , 16, 17  
 vitesse\_interpolee , 182, 183  
 vitesse\_parois , 224  
 vitesse\_particules , 182, 183  
 vitesse\_tangentielle , 247  
 volume , 140–155  
 volume\_sans\_lissage , 140–155  
 weighted\_average , 192, 193  
 weighted\_sum , 192, 193  
 weighted\_sum\_porosity , 192, 193  
 X , 37, 52, 322  
 x , 314  
 xyz , 76, 231  
 Y , 37, 52, 322  
 y , 314  
 yes , 181, 190  
 Z , 37, 52, 322  
 z , 314  
 , 108, 111, 114, 160  
**champs** , 64, 74  
**conditions\_initiales** , 107, 114, 121–124, 126–130, 132–134, 137, 166, 168, 169, 171, 173, 174, 181, 182  
**conditions\_limites** , 78, 107, 114, 121–124, 126–130, 132–134, 137, 166, 168, 169, 171, 173, 177, 181, 183



fichier , 45  
 nom\_zones , 43  
 partitionneur , 43  
 postraitement , 63, 77, 79–87, 89–97, 99–103, 105, 106  
 postraitements , 63, 77, 79–85, 87, 89–97, 99–103, 105, 106  
 Read\_file , 61  
 save\_matrice , 198, 199, 204  
 sondes , 64, 74  
 1D , 162, 163  
 3D , 162, 163  
 A\_plus , 323  
 acceleration , 308  
 alias , 123, 124, 126, 129  
 alpha , 117, 316, 325  
 alpha\_0 , 263  
 alpha\_1 , 263  
 alpha\_a , 263  
 alpha\_sous\_zone , 117  
 amont\_sous\_zone , 117  
 ampli\_bruit , 233  
 ampli\_sin , 233  
 approximation\_de\_boussinesq , 165  
 areva , 164  
 ascii , 16, 55  
 autre\_bord , 207  
 autre\_champ\_indicatrice , 207  
 autre\_champ\_temperature , 207  
 autre\_champ\_temperature\_indic0 , 207  
 autre\_champ\_temperature\_indic1 , 207  
 autre\_probleme , 207  
 avec\_certains\_bords , 28  
 avec\_certains\_bords\_pour\_extraire\_surface , 27  
 avec\_les\_bords , 28  
 beta , 316  
 beta\_co , 252  
 beta\_th , 252  
 binaire , 22, 45  
 boite , 321  
 bord , 20, 160, 310  
 bords\_a\_decouper , 22  
 boundaries , 139  
 boundary\_conditions , 78, 107, 114, 121–124, 126–130, 132–134, 137, 166, 168, 169, 171, 173, 177, 181, 183  
 boundary\_xmax , 40  
 boundary\_xmin , 39  
 boundary\_ymax , 40  
 boundary\_ymin , 40  
 boundary\_zmax , 40  
 boundary\_zmin , 40  
 btd , 120  
 c , 164  
 c0 , 309  
 c1\_eps , 308, 320  
 c2\_eps , 308, 320  
 c3\_eps , 308, 320  
 calc\_spectre , 162  
 calcul\_ldp\_en\_flux\_impose , 327  
 canal , 148  
 canalx , 145  
 cea\_jaea , 164  
 centre\_rotation , 309  
 champ\_med , 33  
 changement\_de\_base\_p1bulle , 228  
 check\_files , 328  
 cl\_pression\_sommet\_faible , 228  
 clipping\_courbure\_interface , 136  
 cmu , 157  
 coef , 250  
 coeff , 310  
 coefficient\_diffusion , 251  
 coefficients\_activites , 196  
 collisions , 175  
 compo , 191  
 condition\_elements , 27, 28  
 condition\_faces , 28  
 condition\_geometrique , 22  
 conduction , 80  
 conduction\_milieu\_variable , 81  
 conservation\_Ec , 162, 163  
 constante\_modele\_micro\_melange , 195  
 constante\_taux\_reaction , 196  
 contre\_energie\_activation , 196  
 contre\_reaction , 196  
 contribution\_one\_way , 183  
 controle\_residu , 199, 303–307  
 convection , 114, 121–124, 126–130, 132–134, 137, 166, 168, 169, 171, 173, 177, 181, 183  
 convection\_diffusion\_chaleur\_qc , 96, 97  
 convection\_diffusion\_chaleur\_turbulent\_qc , 101, 102  
 convection\_diffusion\_concentration , 83, 84, 92, 93  
 convection\_diffusion\_concentration\_turbulent , 85, 86, 94, 95  
 convection\_diffusion\_phase\_field , 89  
 convection\_diffusion\_temperature , 91–93, 98  
 convection\_diffusion\_temperature\_turbulent , 94, 95, 100, 103  
 correction\_fraction , 249  
 correction\_parcours\_thomas , 180  
 correction\_visco\_turb\_pour\_controle\_pas\_de\_temps , 138, 140, 142–144, 146–151, 153–157  
 correction\_visco\_turb\_pour\_controle\_pas\_de\_temps\_parametre , 138, 140, 142–144, 146–149, 151–157

corriger\_partition , 260  
 couplage\_NS\_CH , 316  
 couronne , 321  
 Cp , 249  
 cp , 216, 217, 228, 249, 251–255  
 crank , 112  
 critere\_absolu , 29  
 critere\_arete , 179  
 critere\_longueur\_fixe , 179  
 critere\_remaillage , 179  
 cs , 143  
 Cv , 250  
 cw , 142  
 d , 237, 239  
 debit , 216, 217  
 debit\_impose , 310  
 debug , 164  
 debut\_stat , 160  
 definition\_champs , 64, 74  
 delta , 215  
 derivee\_rotation , 250  
 dh , 216, 217  
 diag , 199  
 diam\_hydr , 312, 313, 326, 328  
 diam\_hydr\_ortho , 312  
 diffusion , 107, 114, 121–124, 126–130, 132–134, 137, 166, 168, 169, 171, 173, 177, 181, 183  
 diffusion\_implicite , 265, 267, 269, 271, 273, 274, 276, 278, 280, 281, 283, 285, 287, 289, 292, 294, 296, 298, 300, 302  
 dim\_espace\_krilov , 199  
 dimension\_espace\_de\_krylov , 316  
 dir , 216, 217  
 dir\_flow , 233  
 dir\_wall , 233  
 direction , 20, 29–31, 160, 312, 313  
 disable\_dt\_ev , 265, 267, 269, 272, 273, 275, 277, 278, 280, 282, 284, 285, 288, 290, 292, 295, 297, 299, 301, 302  
 disable\_progress , 265, 267, 269, 272, 273, 275, 277, 278, 280, 282, 284, 285, 288, 290, 292, 295, 297, 299, 301, 302  
 distance\_projete\_faces , 177  
 dmax , 145  
 domain , 39  
 domaine , 20, 22, 27–31, 45, 64, 74, 189, 190, 261  
 domaine\_final , 21, 29  
 domaine\_flottant\_fluide , 138  
 domaine\_grossier , 22  
 domaine\_init , 21, 29  
 domaines , 45  
 domegadt , 309  
 dt\_impr , 139, 216, 217, 264, 267, 269, 271, 273, 274, 276, 278, 279, 281, 283, 284, 287, 289, 291, 294, 296, 298, 300, 302  
 dt\_impr\_moy\_spat , 160  
 dt\_impr\_moy\_temp , 160  
 dt\_impr\_nusselt , 257–259  
 dt\_impr\_ustar , 139, 141–144, 146–148, 150–157  
 dt\_impr\_ustar\_mean\_only , 139, 141–144, 146–148, 150–157  
 dt\_injection , 184  
 dt\_max , 264, 266, 269, 271, 272, 274, 276, 277, 279, 281, 283, 284, 287, 289, 291, 294, 296, 298, 300, 301  
 dt\_min , 264, 266, 269, 271, 272, 274, 276, 277, 279, 281, 283, 284, 287, 289, 291, 294, 296, 298, 300, 301  
 dt\_post , 164  
 dt\_projection , 137, 166, 167, 169, 171, 173  
 dt\_sauv , 264, 266, 269, 271, 272, 274, 276, 278, 279, 281, 283, 284, 287, 289, 291, 294, 296, 298, 300, 302  
 dt\_start , 265, 267, 269, 271, 273, 275, 276, 278, 280, 282, 283, 285, 287, 290, 292, 294, 297, 299, 300, 302  
 dt\_uniforme , 184  
 dtol\_fraction , 249  
 Ec , 161  
 Ec\_dans\_repere\_fixe , 161  
 ecrire\_decoupage , 43  
 ecrire\_fichier\_xyz\_valeur , 107, 114, 121–123, 125–130, 132–134, 137, 166, 168, 170, 171, 173, 177, 181, 183  
 ecrire\_fichier\_xyz\_valeur\_bin , 107, 114, 121–123, 125–130, 132, 133, 135, 137, 166, 168, 170, 171, 173, 177, 181, 183  
 ecrire\_frontiere , 45  
 ecrire\_lata , 43  
 emissivite\_pour\_rayonnement\_entre\_deux\_plaques-quasi\_infinies , 218  
 energie\_activation , 196  
 ensemble\_points , 184  
 enthalpie\_reaction , 196  
 epaisseur , 27, 29  
 eps\_max , 156, 157  
 eps\_min , 156, 157  
 eq\_rayo\_semi\_transp , 77  
 equation\_frequence\_resolue , 113  
 equation\_interface , 124, 132  
 equation\_interfaces\_proprietes\_fluide , 136  
 equation\_interfaces\_vitesse\_imposee , 136  
 equation\_navier\_stokes , 132  
 equation\_non\_resolue , 107, 113, 114, 121–123, 125–130, 132, 134, 135, 138, 166, 168, 170, 172, 173, 177, 181, 183

equation\_temperature\_mpoint , 136  
 equation\_temperature\_mpoint\_vapeur , 136  
 equations\_interfaces\_vitesse\_imposee , 136  
 equations\_scalaires\_passifs , 79, 84, 86, 93, 95, 97, 98, 102, 103  
 Erugu , 323  
 erugu , 224  
 espece , 127, 128  
 espece\_en\_competition\_micro\_melange , 195  
 expert\_only , 61  
 exposant\_beta , 196  
 expression , 195  
 facon\_init , 162  
 facsec , 264, 267, 269, 271, 273, 274, 276, 278, 279, 281, 283, 285, 287, 289, 291, 294, 296, 298, 300, 302  
 facsec\_max , 268, 271, 286, 288, 291, 293, 295  
 facteur , 120, 325, 329  
 facteur\_longueur\_ideale , 179  
 facteurs , 36  
 fichier , 64, 74, 145, 260, 321  
 fichier\_distance\_paroie , 158  
 fichier\_ecriture\_K\_Eps , 145  
 fichier\_matrice , 55  
 fichier\_post , 20  
 fichier\_secmem , 55  
 fichier\_solution , 55  
 fichier\_solveur , 55  
 fichier\_solveur\_non\_recree , 200  
 fichier\_sortie , 33  
 fields , 64, 74  
 file , 45  
 file\_coord\_x , 39  
 file\_coord\_y , 39  
 file\_coord\_z , 39  
 fin\_stat , 160  
 fonction , 51, 144  
 fonction\_filtre , 41  
 fonction\_sous\_zone , 321  
 format , 45, 64, 74  
 format\_post , 40  
 formate , 43  
 forme\_du\_terme\_source , 318  
 formulation\_a\_nb\_points , 140, 142–146, 148–154  
 frequence\_recalc , 200  
 frontiere , 164  
 function\_coord\_x , 39  
 function\_coord\_y , 39  
 function\_coord\_z , 39  
 gamma , 250, 328  
 genere\_fichier\_solveur , 55  
 ghost\_thickness , 39  
 gmres\_non\_lineaire , 316  
 gravite , 165  
 groupes , 76, 82, 105  
 h , 233, 310  
 haspi , 164  
 hexa\_old , 29  
 ignore\_check\_fraction , 249  
 implication\_CH , 316  
 implicite , 183  
 impr , 55, 179, 197–199, 204  
 impr\_diffusion\_implicite , 265, 267, 269, 271, 273, 275, 276, 278, 280, 282, 283, 285, 287, 290, 292, 294, 297, 299, 300, 302  
 indic\_faces\_modifiee , 177  
 indice , 252, 253  
 info , 109  
 init\_Ec , 162  
 initial\_conditions , 107, 114, 121–124, 126–130, 132–134, 137, 166, 168, 169, 171, 173, 174, 181, 182  
 initial\_value , 233, 234, 240  
 injecteur\_interfaces , 177  
 injection , 182  
 interfaces , 64, 74  
 interpolation\_champ\_face , 176  
 interpolation\_repere\_local , 176  
 intervalle , 321  
 inverse\_condition\_element , 27  
 iterations\_correction\_volume , 175  
 joints\_non\_postraites , 45  
 k , 252  
 k\_min , 156, 157  
 kappa , 252, 253, 316, 323, 325  
 kappa\_variable , 316  
 kmetis , 261  
 lambda , 216, 217, 251–255, 312, 313, 317, 325  
 lambda\_c , 326  
 lambda\_max , 318  
 lambda\_min , 318  
 lambda\_ortho , 312  
 larg\_joint , 43  
 Lire\_fichier , 61  
 lissage\_courbure\_coeff , 179  
 lissage\_courbure\_iterations , 179  
 lissage\_courbure\_iterations\_si\_remaillage , 179  
 lissage\_courbure\_iterations\_systematique , 179  
 liste , 51, 321  
 liste\_cas , 25  
 liste\_de\_postraitements , 63, 77, 79–85, 87–96, 98–103, 105, 106  
 liste\_postraitements , 63, 77, 79–85, 87–96, 98–103, 105, 106  
 localisation , 40, 190, 195  
 loi\_etat , 253  
 longueur\_boite , 162, 163

longueur\_maille , 140, 142–146, 148–153, 155  
 longueurs , 35  
 maillage , 175  
 main , 44  
 maintien\_temperature , 132  
 masse\_molaire , 123, 124, 126, 129, 228  
 matrice\_pression\_invariante , 136  
 max\_iter\_implicit , 266, 286, 289, 291, 293, 296, 298  
 methode , 33, 189, 190, 193, 194  
 methode\_calcul\_face\_keps\_impose , 323  
 methode\_calcul\_pression\_initiale , 136, 165, 167, 169, 171, 172  
 methode\_couplage , 183  
 methode\_interpolation\_v , 176  
 methode\_transport , 175, 182  
 min\_critere\_q\_sur\_max\_critere\_q , 165  
 min\_dir\_flow , 233  
 min\_dir\_wall , 233  
 mode\_calcul\_convection , 121, 122  
 modele\_fonc\_bas\_reynolds , 157  
 modele\_micro\_melange , 195  
 modele\_turbulence , 122, 125, 128, 133, 136, 171, 172  
 modele\_visco , 325, 329  
 modif\_div\_face\_dirichlet , 228  
 moyenne\_convergee , 192  
 moyenne\_de\_kappa , 316  
 mpoint\_inactif\_sur\_qdm , 136  
 mpoint\_vapeur\_inactif\_sur\_qdm , 136  
 mu , 216, 217, 228, 252, 253, 325  
 mu\_1 , 129  
 mu\_2 , 129  
 multiplicateur\_de\_kappa , 316  
 n , 217, 252, 325, 328, 329  
 n\_iterations\_distance , 175  
 n\_iterations\_interpolation\_ibc , 176  
 name\_of\_initial\_zones , 16  
 name\_of\_new\_zones , 16  
 navier\_stokes\_phase\_field , 89  
 navier\_stokes\_qc , 96, 97  
 navier\_stokes\_standard , 82–84, 91–93, 98  
 navier\_stokes\_turbulent , 85–87, 94, 95, 100, 103  
 navier\_stokes\_turbulent\_qc , 101, 102  
 nb\_comp , 233, 234, 239, 240, 329  
 nb\_corrections\_max , 303–306  
 nb\_it\_max , 198, 199, 204, 303–307  
 nb\_iter\_barycentrage , 179  
 nb\_iter\_correction\_volume , 179  
 nb\_iter\_remaillage , 179  
 nb\_iteration\_max\_uzawa , 176  
 nb\_iterations , 183  
 nb\_iterations\_gmresnl , 316  
 nb\_mailles\_mini , 165  
 nb\_nodes , 39  
 nb\_parts , 260–262  
 nb\_parts\_geom , 22  
 nb\_parts\_naif , 22  
 nb\_parts\_tot , 43  
 nb\_pas\_dt\_max , 265, 267, 269, 271, 273, 275, 276, 278, 280, 282, 283, 285, 287, 290, 292, 294, 297, 299, 300, 302  
 nb\_points , 154, 259  
 nb\_points\_par\_phase , 161  
 nb\_procs , 25  
 nb\_test , 55  
 nb\_tranche , 33  
 nb\_tranches , 29–31  
 nb\_var , 144  
 new\_jacobian , 109  
 niter\_avg , 268, 270  
 niter\_max , 268, 270  
 niter\_max\_diffusion\_implicit , 113, 265, 267, 269, 271, 273, 275, 277, 278, 280, 282, 283, 285, 287, 290, 292, 294, 297, 299, 300, 302  
 niter\_min , 268, 270  
 no\_check\_disk\_space , 265, 267, 269, 272, 273, 275, 277, 278, 280, 282, 284, 285, 288, 290, 292, 294, 297, 299, 301, 302  
 no\_conv\_subiteration\_diffusion\_implicit , 265, 267, 269, 271, 273, 275, 276, 278, 280, 282, 283, 285, 287, 290, 292, 294, 297, 299, 300, 302  
 no\_error\_if\_not\_converged\_diffusion\_implicit , 265, 267, 269, 271, 273, 275, 276, 278, 280, 282, 283, 285, 287, 290, 292, 294, 297, 299, 300, 302  
 no\_qdm , 303–307  
 nom , 233, 234, 239, 240  
 nom\_bord , 29  
 nom\_cl\_derriere , 31  
 nom\_cl\_devant , 31  
 nom\_domaine , 40  
 nom\_fichier\_post , 40  
 nom\_fichier\_solveur , 200  
 nom\_fichier\_sortie , 22  
 nom\_frontiere , 189  
 nom\_inconnue , 123–125, 129  
 nom\_mon\_indicatrice , 207  
 nom\_pb , 40  
 nom\_source , 185–195  
 nombre\_de\_noeuds , 36  
 nombre\_facettes\_retenues\_par\_cellule , 176  
 noms\_champs , 40  
 normal\_value , 239  
 normalise , 165  
 nu , 109, 216, 217

nu\_transp , 109  
 numero , 191, 195  
 numero\_op , 186  
 numero\_source , 186  
 nusselt , 328  
 nut , 109  
 nut\_max , 139, 141–144, 146–148, 150–157  
 nut\_transp , 109  
 old , 117  
 omega , 233, 263, 268, 308  
 omega\_relaxation\_drho\_dt , 253  
 optimisation\_sous\_maillage , 190  
 optimized , 198, 204  
 option , 124, 191, 309  
 Origine , 35  
 origine , 27  
 p0 , 228  
 p1 , 228  
 p\_imposee\_aux\_faces , 42  
 pa , 228  
 par\_sous\_zone , 21  
 parametre\_equation , 107, 114, 121–123, 125–130, 132, 134, 135, 137, 166, 168, 170, 172, 173, 177, 181, 183  
 parcours\_interface , 176  
 Partition\_tool , 43  
 pas , 178  
 pas\_de\_solution\_initiale , 55  
 pas\_lissage , 179  
 pb\_champ , 192, 193  
 pb\_name , 44  
 penalisation\_forage , 136  
 penalisation\_l2\_ftd , 130, 132  
 perio\_x , 39  
 perio\_y , 39  
 perio\_z , 39  
 periode , 161  
 periode\_calc\_spectre , 162, 163  
 periode\_sauvegarde\_securite\_en\_heures , 265, 267, 269, 272, 273, 275, 277, 278, 280, 282, 283, 285, 287, 290, 292, 294, 297, 299, 301, 302  
 periodique , 43  
 phase , 124, 132, 207  
 phase\_marquee , 182  
 point1 , 27  
 point2 , 27  
 point3 , 27  
 polynomes , 321  
 position , 250  
 Post\_processing , 63, 77, 79–87, 89–97, 99–103, 105, 106  
 Post\_processings , 63, 77, 79–85, 87, 89–97, 99–103, 105, 106  
 potentiel\_chimique\_generalise , 129  
 prandtl\_turbulent\_fonction\_nu\_t\_alpha , 258  
 Prandtl , 250  
 prandtl , 249, 329  
 prandtl\_eps , 157  
 prandtl\_k , 157  
 prdt , 258  
 prdt\_sur\_kappa , 327  
 precision\_impr , 265, 267, 269, 271, 273, 275, 277, 278, 280, 282, 283, 285, 287, 290, 292, 294, 297, 299, 301, 302  
 precondition , 198, 204  
 precondition0 , 263  
 precondition1 , 263  
 precondition\_nul , 198, 204  
 preconda , 263  
 preconditionnement\_diag , 112  
 pression , 253  
 pression\_reference , 138  
 Probes , 64, 74  
 probleme , 27, 28, 233, 234, 240  
 produits , 196  
 projection\_initiale , 137, 166, 167, 169, 171, 173  
 projection\_normale\_bord , 29  
 proprietes\_particules , 184  
 pulsation\_w , 160  
 quiet , 156, 157, 197–199, 204  
 reactifs , 196  
 reactions , 195  
 rectangle , 321  
 relax\_barycentrage , 179  
 relax\_pression , 306  
 remaillage , 175  
 reorder , 43  
 reprise , 63, 77, 79–83, 85–95, 97–103, 105, 106, 161  
 reprise\_correlation , 217, 218  
 residu\_max\_gmresnl , 316  
 residu\_min\_gmresnl , 316  
 resolution\_explicite , 113  
 restart , 325  
 restriction , 321  
 resume\_last\_time , 63, 77, 79–81, 83–94, 96–103, 105, 106  
 reynolds\_stress\_isotrope , 158  
 rho , 216, 217, 251–254  
 rho\_1 , 129  
 rho\_2 , 129  
 rho\_constant\_pour\_debug , 250  
 rotation , 250  
 sans\_passer\_par\_le2d , 29  
 sans\_solveur\_masse , 186  
 sans\_source\_boussinesq , 325

sauvegarde , 63, 77, 79–85, 87–96, 98–103, 105, 106  
sauvegarde\_simple , 63, 77, 79–84, 86–95, 97–103, 105, 106  
save\_matrix , 198, 199, 204  
sc , 249  
schema\_ch , 300  
schema\_ns , 300  
scturb , 258  
segment , 321  
seuil , 198, 199, 204, 268, 271  
seuil\_convergence\_implicite , 113, 303–307  
seuil\_convergence\_solveur , 113, 303–307  
seuil\_convergence\_uzawa , 176  
seuil\_cv\_iterations\_ptfixe , 316  
seuil\_diffusion\_implicite , 113, 265, 267, 269, 271, 273, 275, 276, 278, 280, 281, 283, 285, 287, 289, 292, 294, 296, 298, 300, 302  
seuil\_divU , 137, 166, 167, 169, 171, 173  
seuil\_dvolume\_residuel , 179  
seuil\_generation\_solveur , 303–307  
seuil\_residu\_gmresnl , 316  
seuil\_residu\_ptfixe , 316  
seuil\_statio , 265, 267, 269, 271, 273, 274, 276, 278, 279, 281, 283, 285, 287, 289, 292, 294, 296, 298, 300, 302  
seuil\_statio\_relatif\_deconseille , 265, 267, 269, 271, 273, 274, 276, 278, 280, 281, 283, 285, 287, 289, 292, 294, 296, 298, 300, 302  
seuil\_test\_preliminaire\_solveur , 303–307  
seuil\_verification , 55  
seuil\_verification\_solveur , 303–307  
solveur , 55, 78, 113, 266, 286, 289, 291, 293, 296, 298, 303–307  
solveur0 , 198  
solveur1 , 198  
solveur\_bar , 137, 166, 167, 169, 171, 173  
solveur\_pression , 137, 166, 167, 169, 171, 173  
sonde\_tble , 325, 329  
source , 185–195  
source\_reference , 185–195  
sources , 107, 114, 121–124, 126–130, 132–134, 137, 166, 168, 170, 171, 173, 177, 181, 183, 185–195  
sources\_reference , 185–195  
sous\_zone , 27, 233, 234, 240, 312, 313  
sous\_zones , 262  
splitting , 39  
stabilise , 154, 259  
standard , 109  
stationnaire , 325  
statistiques , 64, 74  
statistiques\_en\_serie , 64, 74  
stats , 325, 328, 329  
steady\_global\_dt , 266  
steady\_security\_facteur , 266  
stencil\_width , 132  
surface , 217  
surfacique , 44  
sutherland , 253  
symx , 36  
symy , 36  
symz , 36  
t0 , 309  
t\_deb , 164, 187, 188, 192  
t\_debut\_injection , 184  
t\_fin , 164, 187, 188, 192  
tanh , 36  
tanh\_dilatation , 36  
tanh\_taille\_premiere\_maille , 36  
tcpumax , 264, 266, 269, 271, 272, 274, 276, 277, 279, 281, 283, 284, 287, 289, 291, 294, 296, 298, 300, 301  
tdivu , 117  
temps\_d\_affichage , 316  
temps\_debut\_prise\_en\_compte\_drho\_dt , 253  
terme\_gravite , 136  
test , 117  
thi , 147  
tinf , 216, 217  
tinit , 264, 266, 268, 271, 272, 274, 276, 277, 279, 281, 283, 284, 287, 289, 291, 294, 296, 298, 300, 301  
tmax , 264, 266, 268, 271, 272, 274, 276, 277, 279, 281, 283, 284, 287, 289, 291, 294, 296, 298, 300, 301  
traitement\_coins , 42  
traitement\_particulier , 137, 166, 168, 169, 171, 173  
traitement\_pth , 253  
traitement\_rho\_gravite , 253  
tranches , 262  
transformation\_bulles , 182  
transport\_k\_epsilon , 157  
triangle , 27  
trois\_tetra , 29  
tsup , 216, 217  
tube , 321  
turbulence\_paroie , 138, 141–144, 146–148, 150–157, 257–259  
tuyauz , 145  
type , 191  
type\_vitesse\_imposee , 176  
u , 237, 239  
u\_star\_impose , 323  
u\_tau , 326  
ubar\_umprim\_cible , 318

- ucent** , 233
- union** , 321
- use\_weights** , 261
- val\_Ec** , 162
- verif\_boussinesq** , 309
- verif\_dparoi** , 145
- via\_extraire\_surface** , 27
- vingt\_tetra** , 29
- viscosite\_dynamique\_constante** , 165
- vitesse** , 250, 308
- vitesse\_fluide\_explicite** , 180
- vitesse\_imposee\_regularisee** , 177
- volume** , 216
- volume\_impose\_phase\_1** , 176
- volumes\_etendus** , 117
- volumes\_non\_etendus** , 117
- volumique** , 44
- with\_nu** , 181
- xinf** , 217
- xsup** , 217
- zmax** , 33
- zmin** , 33
- zones\_name** , 43
  
- acceleration, 308
- ale, 120
- algo\_base, 184
- algo\_couple\_1, 184
- amont, 115
- amont\_old, 115
- analyse\_angle, 17
- associate, 17
- associer\_algo, 18
- associer\_pbm\_g\_pbf\_in, 18
- associer\_pbm\_g\_pbg\_global, 18
- axi, 18
  
- base, 180
- bidim\_axi, 19
- bord, 36
- bord\_base, 36
- boundary\_field\_inward, 238
- boundary\_field\_uniform\_keps\_from\_ud, 239
- boussinesq\_concentration, 309
- boussinesq\_temperature, 309
- brech, 163
- btd, 119
  
- calcul, 19
- calculer\_moments, 19
- canal, 160
- canal\_perio, 309
- ceg, 163
- centre, 115
- centre4, 115
- centre\_de\_gravite, 19
- centre\_old, 115
- ch\_front\_input, 239
- ch\_front\_input\_uniforme, 240
- champ\_base, 229
- champ\_don\_base, 229
- champ\_don\_lu, 229
- champ\_fonc\_fonction, 229
- champ\_fonc\_fonction\_txyz, 230
- champ\_fonc\_med, 230
- Champ\_Fonc\_MEDfile, 229
- champ\_fonc\_reprise, 230
- champ\_fonc\_t, 231
- champ\_fonc\_tabule, 231
- champ\_fonc\_txyz, 236
- champ\_fonc\_xyz, 236
- champ\_front\_ale, 240
- champ\_front\_base, 238
- champ\_front\_bruite, 240
- champ\_front\_calc, 241
- champ\_front\_contact\_rayo\_semi\_transp\_vef, 241
- champ\_front\_contact\_rayo\_transp\_vef, 242
- champ\_front\_contact\_vef, 242
- champ\_front\_debit, 242
- Champ\_front\_debit\_QC\_VDF, 238
- champ\_front\_fonc\_pois\_ipsn, 242
- champ\_front\_fonc\_pois\_tube, 243
- champ\_front\_fonc\_t, 243
- champ\_front\_fonc\_txyz, 243
- champ\_front\_fonc\_xyz, 243
- champ\_front\_fonction, 244
- champ\_front\_lu, 244
- champ\_front\_MED, 240
- champ\_front\_normal\_vef, 244
- champ\_front\_pression\_from\_u, 245
- champ\_front\_recyclage, 245
- champ\_front\_tabule, 247
- champ\_front\_tangentiel\_vef, 247
- champ\_front\_uniforme, 247
- champ\_front\_vortex, 248
- champ\_front\_zoom, 248
- champ\_generique\_base, 184
- champ\_init\_canal\_sinal, 232
- champ\_input\_base, 233
- champ\_input\_p0, 233
- champ\_ostwald, 234
- champ\_post\_de\_champs\_post, 184
- champ\_post\_extraction, 189
- champ\_post\_interpolation, 190
- champ\_post\_morceau\_equation, 190
- champ\_post\_operateur\_base, 185
- champ\_post\_operateur\_divergence, 188
- champ\_post\_operateur\_eqn, 186



champ\_post\_operateur\_gradient, 189  
 champ\_post\_reduction\_0d, 192  
 champ\_post\_refchamp, 193  
 champ\_post\_statistiques\_base, 186  
 champ\_post\_tparoi\_vdf, 193  
 champ\_post\_transformation, 194  
 champ\_som\_lu\_vdf, 234  
 champ\_som\_lu\_vdf, 234  
 champ\_tabule\_temps, 235  
 champ\_uniforme\_morceaux, 235  
 champ\_uniforme\_morceaux\_tabule\_temps, 235  
 Champ\_front\_fonc\_txyz, 13  
 chimie, 195  
 chmoy\_faceperio, 163  
 Cholesky, 200–202  
 cholesky, 196  
 circle, 68  
 circle\_3, 68  
 class\_generic, 196  
 combinaison, 143  
 Concentration, 70, 72  
 condlim\_base, 205  
 condlims, 78  
 conduction, 107  
 conduction\_milieu\_variable, 113  
 constant, 223  
 constituant, 251  
 contact\_vdf\_vdf, 205  
 contact\_vdf\_vdf, 206  
 convection\_deriv, 114  
 convection\_diffusion\_chaleur\_qc, 120  
 convection\_diffusion\_chaleur\_turbulent\_qc, 121  
 convection\_diffusion\_concentration, 122  
 convection\_diffusion\_concentration\_ft\_disc, 124  
 convection\_diffusion\_concentration\_turbulent, 125  
 convection\_diffusion\_fraction\_massique\_qc, 126  
 convection\_diffusion\_fraction\_massique\_turbulent\_qc, 127  
 convection\_diffusion\_phase\_field, 128  
 convection\_diffusion\_temperature, 130  
 convection\_diffusion\_temperature\_ft\_disc, 131  
 convection\_diffusion\_temperature\_turbulent, 133  
 coriolis, 310  
 Correlation, 69, 70  
 correlation, 72, 187  
 corriger\_frontiere\_periodique, 20  
 create\_domain\_from\_sous\_zone, 20  
 darcy, 310  
 debog, 21  
 decoupebord\_pour\_rayonnement, 21  
 decouper\_bord\_coincident, 22  
 di\_12, 116  
 diffusion\_deriv, 108  
 dilate, 22  
 dimension, 23  
 dirac, 311  
 dirichlet, 206  
 disable\_TU, 23  
 discretisation\_base, 227  
 discretiser\_domaine, 23  
 discretize, 23  
 distance\_parois, 24  
 domain, 38  
 domaine, 228  
 domaine\_ale, 228  
 dt\_calc, 197  
 dt\_fixe, 197  
 dt\_min, 197  
 dt\_start, 197  
 Dt\_post, 69, 70  
 EASM\_Baglietto, 158  
 ec, 161  
 ecart\_type, 71, 188  
 Ecart\_type, 69, 70, 72  
 echange\_contact\_rayo\_transp\_vdf, 206  
 echange\_contact\_vdf\_ft\_disc, 206  
 echange\_contact\_vdf\_ft\_disc\_solid, 207  
 ecrire, 61  
 ecrire\_champ\_med, 24  
 ecrire\_fichier\_bin, 61  
 ecrire\_fichier\_formatte, 24  
 ecrire\_med, 62  
 ecrire\_medfile, 62  
 ecriturelecturespecial, 25  
 ef, 116, 227  
 ef\_stab, 117  
 end, 31  
 entree\_temperature\_imposee\_h, 207  
 epsilon, 38  
 eqn\_base, 134  
 execute\_parallel, 25  
 export, 25  
 extract\_2d\_from\_3d, 26  
 extract\_2daxi\_from\_3d, 26  
 extraire\_domaine, 26  
 extraire\_plan, 27  
 extraire\_surface, 27  
 extrudebord, 28  
 extrudeparoi, 29  
 extruder, 29  
 extruder\_en20, 30  
 extruder\_en3, 30  
 fichier\_decoupage, 260  
 field\_uniform\_keps\_from\_ud, 236  
 flottabilite, 316



fluide\_diphasique, 255  
 fluide\_incompressible, 251  
 fluide\_ostwald, 252  
 fluide\_quasi\_compressible, 253  
 flux\_radiatif, 208  
 flux\_radiatif\_vdf, 208  
 flux\_radiatif\_vef, 208  
 forchheimer, 311  
 frontiere\_ouverte, 208  
 frontiere\_ouverte\_concentration\_imposee, 209  
 frontiere\_ouverte\_fraction\_massique\_imposee, 209  
 frontiere\_ouverte\_gradient\_pression\_impose, 209  
 frontiere\_ouverte\_gradient\_pression\_impose\_vefprep1b, 209  
 frontiere\_ouverte\_gradient\_pression\_libre\_vef, 210  
 frontiere\_ouverte\_gradient\_pression\_libre\_vefprep1b, 210  
 frontiere\_ouverte\_k\_eps\_impose, 210  
 frontiere\_ouverte\_pression\_imposee, 210  
 frontiere\_ouverte\_pression\_imposee\_orlansky, 211  
 frontiere\_ouverte\_pression\_moyenne\_imposee, 211  
 frontiere\_ouverte\_rayo\_semi\_transp, 211  
 frontiere\_ouverte\_rayo\_transp, 211  
 frontiere\_ouverte\_rayo\_transp\_vdf, 212  
 frontiere\_ouverte\_rayo\_transp\_vef, 212  
 frontiere\_ouverte\_rho\_u\_impose, 212  
 frontiere\_ouverte\_temperature\_imposee, 212  
 frontiere\_ouverte\_temperature\_imposee\_rayo\_semi\_transp, 213  
 frontiere\_ouverte\_temperature\_imposee\_rayo\_transp, 213  
 frontiere\_ouverte\_vitesse\_imposee, 213  
 frontiere\_ouverte\_vitesse\_imposee\_sortie, 213  
  
 gaz\_parfait, 249  
 gaz\_reel\_rhot, 248  
 GCP, 200, 203  
 gcp, 203  
 gcp\_ns, 197  
 gen, 198  
 generic, 118  
 gmres, 198  
 Gradient, 200  
  
 IBICGSTAB, 200  
 implicit\_euler\_steady\_scheme, 265  
 implicit\_steady, 303  
 implice, 304  
 imposer\_vit\_bords\_ale, 31  
 imprimer\_flux, 32  
 imprimer\_flux\_sum, 32  
 init\_par\_partie, 237  
 integrer\_champ\_med, 32  
 Interface, 201  
  
 internes, 38  
 interpolation\_champ\_face\_deriv, 180  
 interprete, 15  
 interprete\_geometrique\_base, 33  
  
 Jones\_Launders, 159  
  
 k\_epsilon, 156  
 kquick, 118  
  
 Lam\_Bremhorst, 157  
 lata\_to\_med, 33  
 lata\_to\_other, 34  
 Launder\_Sharma, 158  
 leap\_frog, 273  
 lineaire, 180  
 lire\_ideas, 34  
 lire\_medfile, 17  
 lire\_tgrid, 48  
 list\_bloc\_mailler, 34  
 list\_bord, 36  
 list\_nom, 54  
 list\_nom\_virgule, 185  
 liste\_post, 74  
 liste\_post\_ok, 73  
 listobj, 330  
 listobj\_impl, 329  
 local, 202  
 loi\_analytique\_scalaire, 327  
 loi\_ciofalo\_hydr, 322  
 loi\_etat\_base, 248  
 loi\_expert\_hydr, 323  
 loi\_expert\_scalaire, 327  
 loi\_fermeture\_base, 250  
 loi\_fermeture\_test, 250  
 loi\_horaire, 177, 250  
 loi\_odvm, 327  
 loi\_paroie\_nu\_impose, 328  
 loi\_puissance\_hydr, 323  
 loi\_standard\_hydr, 323  
 loi\_standard\_hydr\_old, 324  
 loi\_standard\_hydr\_scalaire, 328  
 loi\_ww\_hydr, 324  
 loi\_WW\_scalaire, 326  
 longitudinale, 314  
 longueur\_melange, 145  
  
 mailler, 34  
 mailler\_base, 35  
 maillerparallel, 38  
 masse\_ajoutee, 317  
 melange\_gaz\_parfait, 249  
 methode\_transport\_deriv, 177  
 metis, 260

milieu\_base, 251  
milieu\_v2\_base, 255  
mod\_turb\_hyd\_rans, 155  
mod\_turb\_hyd\_ss\_maille, 140  
modele\_fonction\_bas\_reynolds\_base, 157  
modele\_rayo\_semi\_transp, 76  
modele\_rayonnement\_base, 255  
modele\_rayonnement\_milieu\_transparent, 255  
modele\_turbulence\_hyd\_deriv, 138  
modele\_turbulence\_scal\_base, 257  
modif\_bord\_to\_raccord, 40  
mor\_eqn, 107  
Moyenne, 69, 70, 72  
moyenne, 71, 191  
moyenne\_volumique, 40  
muscl, 119  
muscl3, 117  
muscl\_new, 119  
muscl\_old, 119  
  
N, 201  
navier\_stokes\_ft\_disc, 135  
navier\_stokes\_phase\_field, 165  
navier\_stokes\_qc, 167  
navier\_stokes\_standard, 168  
navier\_stokes\_turbulent, 170  
navier\_stokes\_turbulent\_qc, 172  
negligeable, 108, 119, 324  
negligeable\_scalaire, 328  
nettoiepasnoeuds, 41  
neumann, 214  
Neumann\_homogene, 205  
Neumann\_paro\_adiabatique, 205  
nom, 259  
NUL, 139  
NULL, 202  
numero\_elem\_sur\_maitre, 66  
  
objet\_lecture, 330  
optimal, 199  
option, 110  
option\_vdf, 41  
orientefacesbord, 42  
orienter\_simplexes, 49  
  
p1b, 108  
p1ncp1b, 108  
parametre\_diffusion\_implicite, 112  
parametre\_equation\_base, 112  
parametre\_implicite, 113  
Paroi, 205  
paroi\_adiabatique, 214  
paroi\_contact, 214  
paroi\_contact\_fictif, 215  
  
paroi\_decalee\_robin, 215  
paroi\_defilante, 215  
paroi\_echange\_contact\_correlation\_vdf, 216  
paroi\_echange\_contact\_correlation\_vef, 217  
paroi\_echange\_contact\_odvm\_vdf, 218  
paroi\_echange\_contact\_rayo\_semi\_transp\_vdf, 218  
paroi\_echange\_contact\_vdf, 218  
paroi\_echange\_contact\_vdf\_ft, 219  
paroi\_echange\_contact\_vdf\_zoom\_fin, 219  
paroi\_echange\_contact\_vdf\_zoom\_grossier, 219  
paroi\_echange\_externer\_impose, 220  
paroi\_echange\_externer\_impose\_h, 220  
paroi\_echange\_externer\_impose\_rayo\_semi\_transp, 220  
paroi\_echange\_externer\_impose\_rayo\_transp, 221  
paroi\_echange\_global\_impose, 221  
paroi\_fixe, 221  
paroi\_fixe\_iso\_Genepi2\_sans\_contribution\_aux\_vitesses-  
\_sommets, 222  
paroi\_flux\_impose, 222  
paroi\_flux\_impose\_rayo\_semi\_transp\_vdf, 222  
paroi\_flux\_impose\_rayo\_semi\_transp\_vef, 222  
paroi\_flux\_impose\_rayo\_transp, 222  
paroi\_ft\_disc, 223  
paroi\_ft\_disc\_deriv, 223  
paroi\_knudsen\_non\_negligeable, 223  
paroi\_rugueuse, 224  
paroi\_tble, 324  
paroi\_tble\_scal, 328  
paroi\_temperature\_imposee, 224  
paroi\_temperature\_imposee\_rayo\_semi\_transp, 224  
paroi\_temperature\_imposee\_rayo\_transp, 225  
partition, 42, 261  
partitionneur\_deriv, 260  
pave, 35  
pb\_avec\_passif, 78  
Pb\_base, 62  
pb\_conduction, 79  
pb\_conduction\_milieu\_variable, 80  
pb\_couple\_rayo\_semi\_transp, 81  
pb\_couple\_rayonnement, 105  
pb\_gen\_base, 62  
pb\_hydraulique, 82  
pb\_hydraulique\_concentration, 83  
pb\_hydraulique\_concentration\_scalaires\_passifs, 84  
pb\_hydraulique\_concentration\_turbulent, 85  
pb\_hydraulique\_concentration\_turbulent\_scalaires\_passifs,  
86  
pb\_hydraulique\_turbulent, 87  
pb\_mg, 88  
pb\_phase\_field, 88  
pb\_thermohydraulique, 90  
pb\_thermohydraulique\_concentration, 91  
pb\_thermohydraulique\_concentration\_scalaires\_passifs,  
92

pb\_thermohydraulique\_concentration\_turbulent, 93  
 pb\_thermohydraulique\_concentration\_turbulent\_scalaires, 47  
     \_passifs, 94  
 pb\_thermohydraulique\_qc, 96  
 pb\_thermohydraulique\_qc\_fraction\_massique, 97  
 pb\_thermohydraulique\_scalaires\_passifs, 98  
 pb\_thermohydraulique\_turbulent, 99  
 pb\_thermohydraulique\_turbulent\_qc, 100  
 pb\_thermohydraulique\_turbulent\_qc\_fraction\_massique, 101  
 pb\_thermohydraulique\_turbulent\_scalaires\_passifs, 102  
 pbc\_med, 104  
 periodique, 225  
 perte\_charge\_anisotrope, 311  
 perte\_charge\_circulaire, 312  
 perte\_charge\_directionnelle, 312  
 perte\_charge\_isotrope, 313  
 perte\_charge\_reguliere, 313  
 perte\_charge\_singuliere, 314  
 Petsc, 200, 202  
 petsc, 200  
 pilote\_icoco, 43  
 piso, 304  
 plan, 67  
 point, 66  
 points, 66  
 porosites, 44  
 porosites\_champ, 44  
 position\_like, 67  
 post\_processing, 73  
 post\_processings, 72  
 postraitement\_base, 73  
 postraitement\_ft\_lata, 74  
 postraiter\_domaine, 45  
 pp, 131  
 prandtl, 257  
 precisiongeom, 45  
 Precond, 200, 202  
 precondition\_base, 262  
 precondition\_solv, 262  
 predefini, 192  
 Pression, 70, 72  
 Print, 201  
 problem\_read\_generic, 104  
 probleme\_couple, 76  
 probleme\_ft\_disc\_gen, 105  
 profils\_thermo, 163  
 puissance\_thermique, 315  
 quick, 119  
 raccord, 37  
 raffiner\_anisotrope, 45  
 raffiner\_isotrope, 46  
 Raffiner\_isotrope\_parallele, 15  
 read, 47  
 read\_file, 47  
 read\_file\_binary, 48  
 read\_med, 16  
 read\_unsupported\_ascii\_file\_from\_icem, 48  
 redresser\_hexaedres\_vdf, 49  
 refine\_mesh, 49  
 regroupebord, 49  
 remove\_elem, 50  
 remove\_invalid\_internal\_boundaries, 51  
 reordonner, 52  
 reordonner\_faces\_periodiques, 51  
 reorienter\_tetraedres, 51  
 reorienter\_triangles, 51  
 rk3\_ft, 275  
 rotation, 52  
 runge\_kutta\_ordre\_3, 277  
 runge\_kutta\_ordre\_4\_d3p, 278  
 runge\_kutta\_rationnel\_ordre\_2, 280  
 scalaire\_impose\_parois, 225  
 scatter, 52  
 scatterformatte, 53  
 scattermed, 53  
 Sch\_CN\_EX\_iteratif, 267  
 Sch\_CN\_iteratif, 269  
 schema\_adams\_bashforth\_order\_2, 282  
 schema\_adams\_bashforth\_order\_3, 284  
 schema\_adams\_moulton\_order\_2, 285  
 schema\_adams\_moulton\_order\_3, 288  
 schema\_backward\_differentiation\_order\_2, 290  
 schema\_backward\_differentiation\_order\_3, 292  
 schema\_implicite\_base, 297  
 schema\_phase\_field, 299  
 schema\_predictor\_corrector, 301  
 schema\_temps\_base, 264  
 scheme\_euler\_explicit, 272  
 scheme\_euler\_implicit, 295  
 schmidt, 258  
 segment, 67  
 segmentpoints, 66  
 simple, 305  
 simpler, 306  
 solide, 254  
 solide\_milieu\_variable, 254  
 solve, 53  
 Solver, 200, 203  
 Solveur, 200, 202  
 solveur\_implicite\_base, 303  
 solveur\_lineaire\_std, 307  
 solveur\_sys\_base, 204  
 Solveur\_pression, 200, 202  
 sonde\_base, 65

sortie\_libre\_rho\_variable, 225  
 sortie\_libre\_temperature\_imposee\_h, 226  
 source\_base, 307  
 source\_con\_phase\_field, 315  
 source\_constituant, 316  
 source\_generique, 316  
 source\_qdm, 317  
 source\_qdm\_lambdaup, 317  
 source\_qdm\_phase\_field, 318  
 source\_rayo\_semi\_transp, 318  
 source\_robin, 318  
 source\_robin\_scalaire, 318  
 source\_th\_tdivu, 319  
 source\_transport\_k\_eps, 319  
 source\_transport\_k\_eps\_aniso\_concen, 320  
 source\_transport\_k\_eps\_aniso\_therm\_concen, 320  
 Source\_Transport\_K\_Eps\_anisotherme, 308  
 sources, 111  
 sous\_maille, 146  
 sous\_maille\_1elt, 150  
 sous\_maille\_1elt\_selectif\_mod, 151  
 sous\_maille\_axi, 152  
 sous\_maille\_dyn, 258  
 sous\_maille\_selectif, 149  
 sous\_maille\_selectif\_mod, 147  
 sous\_maille\_smago, 142  
 sous\_maille\_smago\_dyn, 154  
 sous\_maille\_smago\_filtre, 153  
 sous\_maille\_wale, 141  
 sous\_zone, 320  
 sous\_zones, 261  
 Spai, 202  
 spec\_pdc\_r\_base, 313  
 SSOR, 202, 203  
 ssor, 263  
 ssor\_bloc, 263  
 stab, 108  
 standard, 109  
 standard\_KEps, 158  
 stat\_post\_deriv, 70  
 Statistiques, 70, 72  
 Statistiques\_en\_serie, 72  
 sugr, 120  
 supprime\_bord, 53  
 symetrie, 223, 226  
 system, 54  
  
 t\_deb, 71  
 t\_fin, 71  
 tayl\_green, 237  
 Temperature, 70, 72  
 temperature, 160  
 temperature\_imposee\_pari, 226  
 test\_solveur, 54  
  
 testeur, 55  
 testeur\_medcoupling, 55  
 tetraedriser, 55  
 tetraedriser\_homogene, 56  
 tetraedriser\_homogene\_compact, 56  
 tetraedriser\_homogene\_fin, 57  
 tetraedriser\_par\_prisme, 58  
 thi, 161  
 thi\_thermo, 162  
 trainee, 319  
 traitement\_particulier\_base, 160  
 tranche, 262  
 transformer, 58  
 transport\_interfaces\_ft\_disc, 174  
 transport\_k\_epsilon, 180  
 transport\_marqueur\_ft, 182  
 transversale, 314  
 trianguler, 58  
 trianguler\_fin, 59  
 trianguler\_h, 59  
 turbulence\_pari\_base, 322  
 turbulence\_pari\_scalaire\_base, 326  
 type, 69, 70, 72, 201, 202  
  
 uniform\_field, 237  
 utau\_imp, 326  
  
 valeur\_totale\_sur\_volume, 237  
 vdf, 227  
 vect\_nom, 60  
 vef, 227  
 vefprep1b, 227  
 verifier\_qualite\_raffinements, 60  
 verifier\_simplexes, 60  
 verifiercoin, 61  
 Vitesse, 70, 72  
 vitesse\_imposee, 177  
 vitesse\_interpolee, 178  
 volume, 67  
  
 xyz, 13