

# **TRUST Reference Manual V1.7.5**

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

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

June 12, 2017

# Contents

<b>1</b>	<b>Syntax to define a mathematical function</b>	<b>12</b>
<b>2</b>	<b>Existing &amp; predefined fields names</b>	<b>13</b>
<b>3</b>	<b>interprete</b>	<b>15</b>
3.1	Raffiner_isotrope_parallele . . . . .	15
3.2	analyse_angle . . . . .	16
3.3	associate . . . . .	16
3.4	axi . . . . .	17
3.5	bidim_axi . . . . .	17
3.6	calculer_moments . . . . .	17
3.7	lecture_bloc_moment_base . . . . .	17
3.7.1	calcul . . . . .	17
3.7.2	centre_de_gravite . . . . .	17
3.7.3	un_point . . . . .	18
3.8	corriger_frontiere_periodique . . . . .	18
3.9	create_domain_from_sous_zone . . . . .	18
3.10	debog . . . . .	19
3.11	{ . . . . .	19
3.12	decoupebord_pour_rayonnement . . . . .	20
3.13	decouper_bord_coincident . . . . .	20
3.14	dilate . . . . .	21
3.15	dimension . . . . .	21
3.16	discretiser_domaine . . . . .	21
3.17	discretize . . . . .	21
3.18	distance_paroι . . . . .	22
3.19	ecrire_champ_med . . . . .	22
3.20	ecrire_fichier_formatte . . . . .	22
3.21	ecriturelecturespecial . . . . .	23
3.22	execute_parallel . . . . .	23
3.23	export . . . . .	23
3.24	extract_2d_from_3d . . . . .	23
3.25	extract_2daxi_from_3d . . . . .	24
3.26	extraire_domaine . . . . .	24
3.27	extraire_plan . . . . .	25
3.28	extraire_surface . . . . .	25
3.29	extrudebord . . . . .	26
3.30	extrudeparoi . . . . .	27
3.31	extruder . . . . .	27
3.32	troisf . . . . .	28
3.33	extruder_en20 . . . . .	28
3.34	extruder_en3 . . . . .	28
3.35	end . . . . .	29
3.36	} . . . . .	29
3.37	imprimer_flux . . . . .	29
3.38	bloc_lecture . . . . .	30
3.39	imprimer_flux_sum . . . . .	30
3.40	integrer_champ_med . . . . .	30
3.41	lata_to_med . . . . .	31
3.42	format_lata_to_med . . . . .	31
3.43	lata_to_other . . . . .	31
3.44	lire_ideas . . . . .	32

3.45	mailler	32
3.46	list_bloc_mailler	32
3.46.1	mailler_base	32
3.46.2	pave	32
3.46.3	bloc_pave	33
3.46.4	list_bord	33
3.46.5	bord_base	34
3.46.6	bord	34
3.46.7	defbord	34
3.46.8	defbord_2	34
3.46.9	defbord_3	35
3.46.10	raccord	35
3.46.11	internes	35
3.46.12	epsilon	36
3.46.13	domain	36
3.47	maillerparallel	36
3.48	modif_bord_to_raccord	37
3.49	moyenne_volumique	38
3.50	nettoiepasnoeuds	39
3.51	option_vdf	39
3.52	orientefacesbord	39
3.53	partition	39
3.54	bloc_decouper	40
3.55	pilote_icoco	41
3.56	porosites	41
3.57	bloc_lecture_poro	41
3.58	porosites_champ	42
3.59	postraiter_domaine	42
3.60	precisiongeom	43
3.61	raffiner_anisotrope	43
3.62	raffiner_isotrope	43
3.63	read	44
3.64	read_file	44
3.65	read_file_binary	45
3.66	lire_tgrid	45
3.67	read_unsupported_ascii_file_from_icem	45
3.68	read_med	46
3.69	orienter_simplexes	46
3.70	redresser_hexaedres_vdf	47
3.71	refine_mesh	47
3.72	regroupebord	47
3.73	remove_elem	47
3.74	remove_elem_bloc	48
3.75	remove_invalid_internal_boundaries	48
3.76	reordonner_faces_periodiques	49
3.77	reorienter_tetraedres	49
3.78	reorienter_triangles	49
3.79	reordonner	49
3.80	rotation	50
3.81	scatter	50
3.82	scatterformatte	50
3.83	scattermed	51
3.84	solve	51
3.85	supprime_bord	51

3.86	list_nom	51
3.87	system	52
3.88	test_solveur	52
3.89	testeur	52
3.90	testeur_medcoupling	53
3.91	tetraedriser	53
3.92	tetraedriser_homogene	54
3.93	tetraedriser_homogene_compact	54
3.94	tetraedriser_homogene_fin	55
3.95	tetraedriser_par_prisme	55
3.96	transformer	56
3.97	triangler	56
3.98	triangler_fin	57
3.99	triangler_h	57
3.100	verifier_qualite_raffinements	58
3.101	vect_nom	58
3.102	verifier_simplexes	58
3.103	verifiercoin	58
3.104	ecrire	59
3.105	ecrire_fichier_bin	59
3.106	ecrire_med	59
<b>4</b>	<b>pb_gen_base</b>	<b>60</b>
4.1	Pb_base	60
4.2	corps_postraitement	61
4.2.1	definition_champs	61
4.2.2	definition_champ	62
4.2.3	sondes	62
4.2.4	sonde	62
4.2.5	sonde_base	62
4.2.6	points	63
4.2.7	listpoints	63
4.2.8	point	63
4.2.9	segmentpoints	63
4.2.10	numero_elem_sur_maitre	64
4.2.11	position_like	64
4.2.12	segment	64
4.2.13	plan	64
4.2.14	volume	65
4.2.15	circle	65
4.2.16	circle_3	65
4.2.17	champs_posts	66
4.2.18	champs_a_post	66
4.2.19	champ_a_post	66
4.2.20	stats_posts	66
4.2.21	list_stat_post	67
4.2.22	stat_post_deriv	67
4.2.23	t_deb	68
4.2.24	t_fin	68
4.2.25	moyenne	68
4.2.26	ecart_type	68
4.2.27	correlation	69
4.2.28	stats_serie_posts	69
4.3	post_processings	70

4.3.1	un_postraitement	70
4.4	liste_post_ok	70
4.4.1	nom_postraitement	70
4.4.2	postraitement_base	70
4.4.3	post_processing	71
4.5	liste_post	71
4.5.1	un_postraitement_spec	72
4.5.2	type_un_post	72
4.5.3	type_postraitement_ft_lata	72
4.6	format_file	72
4.7	probleme_couple	73
4.8	list_list_nom	73
4.9	pb_avec_passif	73
4.10	listeqn	74
4.11	pb_conduction	75
4.12	pb_hydraulique	75
4.13	pb_hydraulique_concentration	76
4.14	pb_hydraulique_concentration_scalaires_passifs	77
4.15	pb_hydraulique_concentration_turbulent	78
4.16	pb_hydraulique_concentration_turbulent_scalaires_passifs	80
4.17	pb_hydraulique_turbulent	81
4.18	pb_post	82
4.19	pb_thermohydraulique	83
4.20	pb_thermohydraulique_concentration	84
4.21	pb_thermohydraulique_concentration_scalaires_passifs	85
4.22	pb_thermohydraulique_concentration_turbulent	86
4.23	pb_thermohydraulique_concentration_turbulent_scalaires_passifs	87
4.24	pb_thermohydraulique_qc	88
4.25	pb_thermohydraulique_qc_fraction_massique	89
4.26	pb_thermohydraulique_scalaires_passifs	90
4.27	pb_thermohydraulique_turbulent	91
4.28	pb_thermohydraulique_turbulent_qc	93
4.29	pb_thermohydraulique_turbulent_qc_fraction_massique	94
4.30	pb_thermohydraulique_turbulent_scalaires_passifs	95
4.31	pb_med	96
4.32	list_info_med	96
4.32.1	info_med	96
4.33	problem_read_generic	97
<b>5</b>	<b>mor_eqn</b>	<b>98</b>
5.1	conduction	98
5.2	bloc_diffusion	99
5.2.1	diffusion_deriv	99
5.2.2	negligeable	99
5.2.3	p1b	99
5.2.4	p1ncp1b	99
5.2.5	stab	100
5.2.6	standard	100
5.2.7	bloc_diffusion_standard	101
5.2.8	option	101
5.2.9	op_implicite	101
5.3	condinits	102
5.3.1	condinit	102
5.4	condlims	102

5.4.1	condlimlu	102
5.5	sources	103
5.6	ecrire_fichier_xyz_valeur_param	103
5.6.1	ecrire_fichier_xyz_valeur_item	103
5.6.2	bords_ecrire	103
5.7	parametre_equation_base	104
5.7.1	parametre_diffusion_implicite	104
5.7.2	parametre_implicite	104
5.8	convection_diffusion_chaleur_qc	105
5.9	bloc_convection	106
5.9.1	convection_deriv	106
5.9.2	amont	106
5.9.3	amont_old	107
5.9.4	centre	107
5.9.5	centre4	107
5.9.6	centre_old	107
5.9.7	di_l2	107
5.9.8	ef	107
5.9.9	bloc_ef	108
5.9.10	muscl3	108
5.9.11	ef_stab	109
5.9.12	listsous_zone_valeur	109
5.9.13	sous_zone_valeur	109
5.9.14	generic	110
5.9.15	kquick	110
5.9.16	muscl	110
5.9.17	muscl_old	110
5.9.18	muscl_new	111
5.9.19	negligeable	111
5.9.20	quick	111
5.9.21	btd	111
5.9.22	supg	112
5.10	convection_diffusion_chaleur_turbulent_qc	112
5.11	convection_diffusion_concentration	113
5.12	convection_diffusion_concentration_turbulent	114
5.13	convection_diffusion_fraction_massique_qc	115
5.14	convection_diffusion_fraction_massique_turbulent_qc	116
5.15	convection_diffusion_temperature	117
5.16	pp	118
5.16.1	penalisation_l2_ftd_lec	119
5.17	convection_diffusion_temperature_turbulent	119
5.18	eqn_base	120
5.19	navier_stokes_qc	121
5.20	deuxmots	123
5.21	floatfloat	123
5.22	traitement_particulier	123
5.22.1	traitement_particulier_base	123
5.22.2	temperature	123
5.22.3	canal	124
5.22.4	ec	124
5.22.5	thi	125
5.22.6	chmoy_faceperio	126
5.23	navier_stokes_standard	126
5.24	navier_stokes_turbulent	128

5.25	modele_turbulence_hyd_deriv	129
5.25.1	dt_impr_ustar_mean_only	130
5.25.2	NUL	131
5.25.3	mod_turb_hyd_ss_maille	131
5.25.4	form_a_nb_points	132
5.25.5	sous_maille_wale	133
5.25.6	sous_maille_smago	134
5.25.7	combinaison	135
5.25.8	longueur_melange	137
5.25.9	sous_maille	138
5.25.10	k_epsilon	139
5.25.11	modele_fonction_bas_reynolds_base	141
5.26	navier_stokes_turbulent_qc	141
5.27	transport_k_epsilon	143
<b>6</b>	<b>/*</b>	<b>144</b>
6.1	/*	144
<b>7</b>	<b>champ_generique_base</b>	<b>144</b>
7.1	champ_post_de_champs_post	144
7.2	list_nom_virgule	145
7.3	listchamp_generique	145
7.4	champ_post_operateur_base	145
7.5	champ_post_operateur_eqn	145
7.6	champ_post_statistiques_base	146
7.7	correlation	147
7.8	champ_post_operateur_divergence	147
7.9	ecart_type	148
7.10	champ_post_extraction	148
7.11	champ_post_operateur_gradient	149
7.12	champ_post_interpolation	149
7.13	champ_post_morceau_equation	150
7.14	moyenne	151
7.15	predefini	151
7.16	champ_post_reduction_0d	152
7.17	champ_post_refchamp	152
7.18	champ_post_tparoi_vef	153
7.19	champ_post_transformation	153
<b>8</b>	<b>chimie</b>	<b>154</b>
8.1	reactions	154
8.1.1	reaction	155
<b>9</b>	<b>class_generic</b>	<b>155</b>
9.1	cholesky	155
9.2	dt_calc	156
9.3	dt_fixe	156
9.4	dt_min	156
9.5	dt_start	156
9.6	gcp_ns	156
9.7	gen	157
9.8	gmres	158
9.9	optimal	158
9.10	petsc	159

9.11	gcp	162
9.12	solveur_sys_base	163
<b>10</b>	<b>#</b>	<b>163</b>
10.1	#	163
<b>11</b>	<b>condlim_base</b>	<b>164</b>
11.1	Paroi	164
11.2	dirichlet	164
11.3	entree_temperature_imposee_h	164
11.4	frontiere_ouverte	165
11.5	frontiere_ouverte_concentration_imposee	165
11.6	frontiere_ouverte_fraction_massique_imposee	165
11.7	frontiere_ouverte_gradient_pression_impose	165
11.8	frontiere_ouverte_gradient_pression_impose_vef	166
11.9	frontiere_ouverte_gradient_pression_impose_vefprep1b	166
11.10	frontiere_ouverte_gradient_pression_libre_vef	166
11.11	frontiere_ouverte_gradient_pression_libre_vefprep1b	166
11.12	frontiere_ouverte_k_eps_impose	166
11.13	frontiere_ouverte_pression_imposee	167
11.14	frontiere_ouverte_pression_imposee_orlansky	167
11.15	frontiere_ouverte_pression_moyenne_imposee	167
11.16	frontiere_ouverte_rho_u_impose	167
11.17	frontiere_ouverte_temperature_imposee	168
11.18	frontiere_ouverte_vitesse_imposee	168
11.19	frontiere_ouverte_vitesse_imposee_sortie	168
11.20	neumann	168
11.21	paroi_adiabatique	169
11.22	paroi_contact	169
11.23	paroi_contact_fictif	170
11.24	paroi_couple	170
11.25	paroi_decalee_robin	170
11.26	paroi_defilante	171
11.27	paroi_echange_contact_correlation_vdf	171
11.28	paroi_echange_contact_correlation_vef	172
11.29	paroi_echange_contact_vdf	173
11.30	paroi_echange_externe_impose	173
11.31	paroi_echange_externe_impose_h	173
11.32	paroi_echange_global_impose	174
11.33	paroi_fixe	174
11.34	paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets	174
11.35	paroi_flux_impose	174
11.36	paroi_knudsen_non_negligeable	175
11.37	paroi_rugueuse	175
11.38	paroi_temperature_imposee	175
11.39	periodique	176
11.40	scalaire_impose_paro	176
11.41	sortie_libre_temperature_imposee_h	176
11.42	symetrie	176
11.43	temperature_imposee_paro	176



<b>12 discretisation_base</b>	<b>177</b>
12.1 ef	177
12.2 vdf	177
12.3 vef	177
12.4 vefprep1b	177
<b>13 domaine</b>	<b>178</b>
<b>14 espece</b>	<b>178</b>
<b>15 champ_base</b>	<b>179</b>
15.1 champ_base	179
15.2 champ_don_base	179
15.3 champ_don_lu	179
15.4 champ_fonc_fonction	179
15.5 champ_fonc_fonction_txyz	180
15.6 champ_fonc_med	180
15.7 champ_fonc_reprise	180
15.8 fonction_champ_reprise	181
15.9 champ_fonc_t	181
15.10 champ_fonc_tabule	181
15.11 champ_init_canal_sinal	182
15.12 bloc_lec_champ_init_canal_sinal	182
15.13 champ_input_base	183
15.14 champ_input_p0	183
15.15 champ_ostwald	184
15.16 champ_som_lu_vdf	184
15.17 champ_som_lu_vef	184
15.18 champ_tabule_temps	185
15.19 champ_uniforme_morceaux	185
15.20 champ_uniforme_morceaux_tabule_temps	185
15.21 champ_fonc_txyz	186
15.22 champ_fonc_xyz	186
15.23 field_uniform_keps_from_ud	186
15.24 init_par_partie	187
15.25 tayl_green	187
15.26 uniform_field	187
15.27 valeur_totale_sur_volume	187
<b>16 champ_front_base</b>	<b>188</b>
16.1 champ_front_base	188
16.2 boundary_field_inward	188
16.3 boundary_field_uniform_keps_from_ud	188
16.4 ch_front_input	189
16.5 ch_front_input_uniforme	189
16.6 champ_front_bruite	190
16.7 champ_front_calc	190
16.8 champ_front_contact_vef	190
16.9 champ_front_debit	191
16.10 champ_front_fonc_pois_ipsn	191
16.11 champ_front_fonc_pois_tube	191
16.12 champ_front_fonc_txyz	191
16.13 champ_front_fonc_xyz	192
16.14 champ_front_fonction	192

16.15	champ_front_lu	192
16.16	champ_front_normal_vef	192
16.17	champ_front_pression_from_u	193
16.18	champ_front_recyclage	193
16.19	champ_front_tabule	195
16.20	champ_front_tangentiel_vef	195
16.21	champ_front_uniforme	195
<b>17</b>	<b>loi_etat_base</b>	<b>195</b>
17.1	gaz_reel_rhot	196
17.2	melange_gaz_parfait	196
17.3	gaz_parfait	196
<b>18</b>	<b>loi_fermeture_base</b>	<b>197</b>
18.1	loi_fermeture_test	197
<b>19</b>	<b>loi_horaire</b>	<b>197</b>
<b>20</b>	<b>milieu_base</b>	<b>197</b>
20.1	constituant	198
20.2	fluide_incompressible	198
20.3	fluide_ostwald	199
20.4	fluide_quasi_compressible	199
20.5	bloc_sutherland	200
20.6	solide	201
<b>21</b>	<b>modele_turbulence_scal_base</b>	<b>201</b>
21.1	prandtl	202
21.2	schmidt	202
<b>22</b>	<b>nom</b>	<b>203</b>
22.1	nom_anonyme	203
<b>23</b>	<b>partitionneur_deriv</b>	<b>203</b>
23.1	fichier_decoupage	204
23.2	metis	204
23.3	partition	205
23.4	sous_zones	205
23.5	tranche	206
<b>24</b>	<b>precond_base</b>	<b>206</b>
24.1	precond_local	206
24.2	precondsolv	206
24.3	ssor	207
24.4	ssor_bloc	207
<b>25</b>	<b>schema_temps_base</b>	<b>208</b>
25.1	Sch_CN_EX_iteratif	209
25.2	Sch_CN_iteratif	211
25.3	scheme_euler_explicit	213
25.4	leap_frog	215
25.5	runge_kutta_ordre_3	217
25.6	runge_kutta_ordre_4_d3p	218
25.7	runge_kutta_rationnel_ordre_2	220
25.8	schema_adams_bashforth_order_2	221

25.9	schema_adams_bashforth_order_3	223
25.10	schema_adams_moulton_order_2	225
25.11	schema_adams_moulton_order_3	227
25.12	schema_backward_differentiation_order_2	229
25.13	schema_backward_differentiation_order_3	231
25.14	scheme_euler_implicit	234
25.15	schema_implicite_base	236
25.16	schema_predictor_corrector	238
<b>26</b>	<b>solveur_implicite_base</b>	<b>239</b>
26.1	implicite	240
26.2	piso	240
26.3	simple	241
26.4	simpler	242
26.5	solveur_lineaire_std	243
<b>27</b>	<b>source_base</b>	<b>243</b>
27.1	Source_Transport_K_Eps_anisotherme	244
27.2	acceleration	244
27.3	boussinesq_concentration	245
27.4	boussinesq_temperature	245
27.5	canal_perio	245
27.6	coriolis	246
27.7	darcy	246
27.8	dirac	247
27.9	forchheimer	247
27.10	perte_charge_anisotrope	247
27.11	perte_charge_circulaire	248
27.12	perte_charge_directionnelle	248
27.13	perte_charge_isotrope	249
27.14	perte_charge_reguliere	249
27.15	spec_pdc_base	249
27.15.1	longitudinale	250
27.15.2	transversale	250
27.16	perte_charge_singuliere	250
27.17	puissance_thermique	251
27.18	source_constituant	251
27.19	source_generique	251
27.20	source_qdm	252
27.21	source_qdm_lambdaup	252
27.22	source_robin	252
27.23	source_robin_scalaire	253
27.24	listdeuxmots_sacc	253
27.25	source_th_tdivu	253
27.26	source_transport_k_eps	253
27.27	source_transport_k_eps_aniso_concen	254
27.28	source_transport_k_eps_aniso_therm_concen	254
<b>28</b>	<b>sous_zone</b>	<b>254</b>
28.1	bloc_origine_cotes	255
28.2	deuxentiers	256
28.3	bloc_couronne	256
28.4	bloc_tube	256

<b>29</b>	<b>turbulence_paroι_base</b>	<b>257</b>
29.1	loi_expert_hydr	257
29.2	loi_standard_hydr	257
29.3	loi_standard_hydr_old	258
29.4	negligeable	258
29.5	paroi_tble	258
29.6	twofloat	259
29.7	liste_sonde_tble	259
29.7.1	sonde_tble	259
29.8	entierfloat	259
29.9	utau_imp	260
<b>30</b>	<b>turbulence_paroι_scalaire_base</b>	<b>260</b>
30.1	loi_analytique_scalaire	260
30.2	loi_expert_scalaire	260
30.3	loi_paroι_nu_impose	261
30.4	loi_standard_hydr_scalaire	261
30.5	negligeable_scalaire	261
30.6	paroi_tble_scal	262
30.7	fourfloat	262
<b>31</b>	<b>listobj_impl</b>	<b>262</b>
31.1	list_un_pb	263
31.2	un_pb	263
31.3	listobj	263
<b>32</b>	<b>objet_lecture</b>	<b>263</b>
32.1	paroi_ft_disc_deriv	264
32.1.1	symetrie	264
32.2	methode_transport_deriv	264
32.2.1	loi_horaire	264
<b>33</b>	<b>index</b>	<b>264</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 : cosinus function  
 SIN : sinus function  
 TAN : tan function  
 ATAN : arctan function  
 EXP : exponential function  
 LN : neperian logaithm function  
 SQRT : root mean square function  
 INT : integer function  
 ERF : erf function  
 RND(x) : random function (values between 0 and x)  
 COSH : hyperbolic cosinus function  
 SINH : hyperbolic sinus function  
 TANH : hyperbolic tangent function  
 ACOS : inverse cosinus function  
 ATANH : inverse hyperbolic tangent function

NOT(x) : not equal to x  
 x\_AND\_y : and function (returns 1 if x and y true else 0)  
 x\_OR\_y : or function (returns 1 if x or y true else 0)  
 x\_GT\_y : greater to (returns 1 if x>y else 0)  
 x\_GE\_y : greater or equal to (returns 1 if x>=y else 0)  
 x\_LT\_y : lesser to (returns 1 if x<y else 0)  
 x\_LE\_y : lesser or equal to (returns 1 if x<=y else 0)  
 x\_MIN\_y : minimum of x and y  
 x\_MAX\_y : maximum of x and y  
 x\_MOD\_y : modular division of x per y  
 x\_EQ\_y : equal to (returns 1 if x=y else 0)  
 x\_NEQ\_y : not equal to (returns 1 if x!=y else 0)

You can also use the following operations:

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

You can also use the following constants:

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

The variables which can be used are:

x,y,z : coordinates  
 t : time

#### Examples:

Champ\_front\_fonc\_txyz 2 cos(y+x^2) t+ln(y)  
 Champ\_fonc\_xyz dom 2 tanh(4\*y)\*(0.95+0.1\*rnd(1)) 0.

#### Possible error:

Champ\_fonc\_txyz 1 cos(10\*t)\*(1<x<2)\*(1<y<2)  
 Previous line is wrong. It should be written:  
 Champ\_fonc\_txyz 1 cos(10\*t)\*(1<x)\*(x<2)\*(1<y)\*(y<2)

## 2 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
Speed	Vitesse or Velocity	$m.s^{-1}$
Kinetic energy per elements ( $0.5\rho  u_i  ^2$ )	Energie_cinetique_elem	$kg.m^{-1}.s^{-2}$
... continued on next page ...		

Physical values	Keyword for field_name	Unit
Total kinetic energy $\left( \frac{\sum_{i=1}^{nb\_elem} 0.5\rho  u_i  ^2 vol_i}{\sum_{i=1}^{nb\_elem} vol_i} \right)$	Energie_cinetique_totale	$kg.m^{-1}.s^{-2}$
Vorticity	Vorticite	$s^{-1}$
Pressure in incompressible flow $(P/\rho + gz)$ For Front Tracking probleme $(P + \rho gz)$	Pression <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	$kg.m.s^{-1}$
Turbulent kinetic energy	K	$m^2.s^{-2}$
Turbulent dissipation rate	Eps	$m^3.s^{-1}$
Turbulent quantities K and Epsilon	K_Eps	$(m^2.s^{-2}, m^3.s^{-1})$
Constituent concentration	Concentration	
Component velocity along X	VitesseX	$m.s^{-1}$
Component velocity along Y	VitesseY	$m.s^{-1}$
Component velocity along Z	VitesseZ	$m.s^{-1}$
Mass balance on each cell	Divergence_U	$m^3.s^{-1}$
Irradiancy	Irradiance	$W.m^{-2}$
Q-criteria	Critere_Q	$s^{-1}$
Distance to the wall $Y^+ = yU/\nu$ (only computed on boundaries of wall type)	Y_plus	dimensionless
Friction velocity	U_star	$m.s^{-1}$
... 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
Cell volumes	Volume_maille	$m^3$
Chemical potential	Potentiel_Chimique_Generalise	
Source term in non Galilean referential	Acceleration_terme_source	$m.s^{-2}$
Stability time steps	Pas_de_temps	S
Boundary fluxes	Flux_bords	
Volumetric porosity	Porosite_volumique	dimensionless
Distance to the wall	Distance_Paroi <sup>3</sup>	$m$
Volumic thermal power	Puissance_volumique	$W.m^{-3}$
Local shear strain rate defined as $\sqrt{(2S_{ij}S_{ij})}$	Taux_cisaillement	$s^{-1}$
Cell Courant number (VDF only)	Courant_maille	dimensionless
Cell Reynolds number (VDF only)	Reynolds_maille	dimensionless

### 3 interpret

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

See also: objet\_u (3.3) read (3.63) associate (3.3) discretize (3.17) mailler (3.45) maillerparallel (3.47) ecrire\_fichier\_bin (3.105) ecrire (3.104) read\_file (3.64) lire\_tgrid (3.66) solve (3.84) execute\_parallel (3.22) end (3.35) dimension (3.15) bidim\_axi (3.5) axi (3.4) transformer (3.96) rotation (3.80) dilate (3.14) testeur (3.89) test\_solveur (3.88) postraiter\_domaine (3.59) modif\_bord\_to\_raccord (3.48) remove\_elem (3.73) regroupebord (3.72) supprime\_bord (3.85) calculer\_moments (3.6) imprimer\_flux (3.37) decouper\_bord\_coincident (3.13) raffiner\_anisotrope (3.61) raffiner\_isotrope (3.62) trianguler (3.97) tetraedriser (3.91) orientefacesbord (3.52) reorienter\_tetraedres (3.77) reorienter\_triangles (3.78) verifiercoin (3.103) porosites (3.56) porosites\_champ (3.58) discretiser\_domaine (3.16) { (3.11) } (3.36) export (3.23) debug (3.10) pilote\_icoco (3.55) moyenne\_volumique (3.49) ecrire\_champ\_med (3.19) read\_med (3.68) lire\_ideas (3.44) ecrire\_med (3.106) system (3.87) redresser\_hexaedres\_vdf (3.70) analyse\_angle (3.2) remove\_invalid\_internal\_boundaries (3.75) reordonner (3.79) option\_vdf (3.51) precisiongeom (3.60) nettoiepasnoeuds (3.50) scatter (3.81) partition (3.53) reordonner\_faces\_periodiques (3.76) corriger\_frontiere\_periodique (3.8) distance\_paroi (3.18) extrudebord (3.29) extruder (3.31) extract\_2d\_from\_3d (3.24) extruder\_en20 (3.33) extrudeparoi (3.30) ecrirelecturespecial (3.21) lata\_to\_med (3.41) lata\_to\_other (3.43) decoupebord\_pour\_rayonnement (3.12) extraire\_plan (3.27) create\_domain\_from\_sous\_zone (3.9) extraire\_domaine (3.26) extraire\_surface (3.28) integrer\_champ\_med (3.40) orienter\_simplexes (3.69) verifier\_simplexes (3.102) verifier\_qualite\_raffinements (3.100) testeur\_medcoupling (3.90) Raffiner\_isotrope\_parallele (3.1) refine\_mesh (3.71)

Usage:  
**interpret**

#### 3.1 Raffiner\_isotrope\_parallele

Description: Refine parallel mesh in parallel

See also: interpret (3)

Usage:  
**Raffiner\_isotrope\_parallele** {

<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.

```

    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 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*

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

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

Usage:

```

associate objet_1 objet_2
where

```

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



### 3.4 axi

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

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

Usage:

**axi**

### 3.5 bidim\_axi

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

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

Usage:

**bidim\_axi**

### 3.6 calculer\_moments

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

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

Usage:

**calculer\_moments nom\_dom mot**

where

- **nom\_dom** *str*: Name of domain.
- **mot** *lecture\_bloc\_moment\_base (3.7)*: Keyword.

### 3.7 lecture\_bloc\_moment\_base

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

See also: [objet\\_lecture \(32\)](#) [calcul \(3.7.1\)](#) [centre\\_de\\_gravite \(3.7.2\)](#)

Usage:

#### 3.7.1 calcul

Description: The centre of gravity will be calculated.

See also: [\(3.7\)](#)

Usage:

**calcul**

#### 3.7.2 centre\_de\_gravite

Description: To specify a specific centre of gravity.

See also: [\(3.7\)](#)

Usage:

**centre\_de\_gravite point**  
where

- **point** *un\_point* [\(3.7.3\)](#): A centre of gravity.

### 3.7.3 un\_point

Description: A point.

See also: [objet\\_lecture \(32\)](#)

Usage:

**pos**  
where

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

## 3.8 corriger\_frontiere\_periodique

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

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

Usage:

```
corriger_frontiere_periodique {  
    domaine str  
    bord str  
    [ direction n x1 x2 ... xn ]  
    [ fichier_post str ]  
}
```

where

- **domaine** *str*: Name of domain.
- **bord** *str*: the name of the boundary (which must contain two opposite sides of the domain)
- **direction** *n x1 x2 ... xn*: defines the periodicity direction vector (a vector that points from one node on one side to the opposite node on the other side. This vector must be given if the automatic algorithm fails, that is:
  - when the node coordinates are not perfectly periodic
  - when the periodic direction is not aligned with the normal vector of the boundary faces
- **fichier\_post** *str*: see `corriger_coordonnees`

## 3.9 create\_domain\_from\_sous\_zone

Description: These keyword fills the domain `domaine_final` with the subzone `par_sous_zone` from the domain `domaine_init`. It is very useful when meshing several mediums with Gmsh. Each medium will be defined as a subzone into Gmsh. A MED mesh file will be saved from Gmsh and read with `Lire_Med`

keyword by the TRUST data file. And with this keyword, a domain will be created for each medium in the TRUST data file.

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

Usage:

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

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

}

where

- **domaine\_final** *str*: domaine dans lequel stocke les faces
- **par\_sous\_zone** *str*: sous zone permettant de choisir les elements
- **domaine\_init** *str*: domaine d origine

### 3.10 debog

Description: Class to debug some differences between two TRUST versions on a same data file.

If you want to compare the results of the same code in sequential and parallel calculation, first run (mode=0) in sequential mode (the files fichier1 and fichier2 will be written first) then the second run in parallel calculation (mode=1).

During the first run (mode=0), it prints into the file DEBOG, values at different points of the code thanks to the C++ instruction call. see for example in Noyau/Resoudre.cpp file the instruction: `Debug::verifier(msg,value);` Where msg is a string and value may be a double, integer or array.

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

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

Usage:

**debog pb fichier1 fichier2 seuil mode**

where

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

### 3.11 {

Description: Block's beginning.

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

Usage:

{

### 3.12 decoupebord\_pour\_rayonnement

Description: To subdivide the external boundary of a domain in several parts (may be useful for better accuracy when using radiation model in transparent medium). to specify the boundaries of the fine\_domain\_name domain to be splitted. These boundaries will be cut according the coarse mesh defined by either the keyword `domaine_grossier` (each boundary face of the coarse mesh `coarse_domain_name` will be used to group boundary faces of the fine mesh to define a new boundary), either by the keyword `nb_parts_naif` (each boundary of the fine mesh is splitted into a partition with  $n_x \times n_y \times n_z$  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*

### 3.13 decouper\_bord\_coincident

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

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

Usage:

```
decouper_bord_coincident domain_name bord
```

where

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

### 3.14 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.

### 3.15 dimension

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

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

Usage:

**dimension dim**

where

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

### 3.16 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.

### 3.17 discretize

Synonymous: **discretiser**

Description: Keyword to discretise a problem problem\_name according to the discretisation dis.

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

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

Usage:

**discretize problem\_name dis**

where

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

### 3.18 distance\_pari

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

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

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).

### 3.19 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

### 3.20 ecrire\_fichier\_formatte

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

See also: [ecrire\\_fichier\\_bin \(3.105\)](#)

Usage:

**ecrire\_fichier\_formatte name\_obj filename**

where

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

### 3.21 **ecriturelecturespecial**

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

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

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).

### 3.22 **execute\_parallel**

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

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

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.

### 3.23 **export**

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

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

Usage:

**export**

### 3.24 **extract\_2d\_from\_3d**

Description: Keyword to extract a 2D mesh by selecting a boundary of the 3D mesh. To generate a 2D axisymmetric mesh prefer Extract\_2Daxi\_from\_3D keyword.

See also: [interpret \(3\)](#) [extract\\_2daxi\\_from\\_3d \(3.25\)](#)

Usage:

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

where

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

### 3.25 extract\_2daxi\_from\_3d

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

See also: `extract_2d_from_3d` ([3.24](#))

Usage:

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

where

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

### 3.26 extraire\_domaine

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

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

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

Usage:

**extraire\_domaine {**

**domaine** *str*  
**probleme** *str*  
[ **condition\_elements** *str*]  
[ **sous\_zone** *str*]

**}**

where

- **domaine** *str*: domaine dans lequel stocke les faces
- **probleme** *str*: Probleme duquel il faut extraire les faces
- **condition\_elements** *str*
- **sous\_zone** *str*



### 3.27 extraire\_plan

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

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

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

Usage:

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

### 3.28 extraire\_surface

Description: This keyword extract a surface mesh named `domain_name` (this domain should have be declared before) from the mesh of the `pb_name` problem. The surface mesh is defined by one or two conditions. The first condition is about elements with `Condition_elements`. For example: `Condition_elements x*x+y*y+z*z<1`

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

By default, the faces from the boundaries are not added to the surface mesh excepted if option `avec_les_bords` is given (all the boundaries are added), or if the option `avec_certaines_bords` is used to add only

some boundaries.

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

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

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*: domaine dans lequel stocke les faces
- **probleme** *str*: Probleme duquel il faut extraire les faces
- **condition\_elements** *str*
- **condition\_faces** *str*
- **avec\_les\_bords**
- **avec\_certains\_bords** *n word1 word2 ... wordn*

### 3.29 extrudebord

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

Warning: If the initial domain is an tetrahedral mesh, the boundary will be moved in the XY plan then extrusion will be applied (you should may be use the Transformer keyword on the final domain to have the domain you really want). You can use the keyword `Ecrire_Fichier_Meshtv` to generate a meshtv file to visualize your initial and final meshes.

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

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

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

Usage:

```
extrudebord {  
    domaine_init str  
    [ direction x1 x2 (x3) ]  
    [ nb_tranches int ]  
    [ domaine_final str ]  
    [ nom_bord str ]  
    [ non_perio ]  
    [ hexa_old ]  
    [ trois_tetra ]  
    [ vingt_tetra ]  
    [ sans_passer_par_le2D int ]  
}
```

where

- **domaine\_init** *str*: Initial domain with hexaedras or tetrahedras.
- **direction** *x1 x2 (x3)*: Directions for the extrusion.
- **nb\_tranches** *int*: Number of elements in the extrusion direction.
- **domaine\_final** *str*: Extruded domain.
- **nom\_bord** *str*: Name of the boundary of the initial domain where extrusion will be applied.
- **non\_perio** : Extruded domain will not have periodic boundaries. So, the boundaries will be named DEVANT and DERRIERE instead of PERIO.
- **hexa\_old** : Old algorithm for boundary extrusion from a hexahedral mesh.
- **trois\_tetra** : To extrude in 3 tetrahedras instead of 14 tetrahedras.
- **vingt\_tetra** : To extrude in 20 tetrahedras instead of 14 tetrahedras.
- **sans\_passer\_par\_le2D** *int*: Only for non regression

### 3.30 extrudeparoi

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

### 3.31 extruder

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

See also: [interpret \(3\)](#) [extruder\\_en3 \(3.34\)](#)

Usage:

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

}

where

- **domaine** *str*: Name of the domain.
- **direction** *troisf* (3.32): Direction of the extrude operation.
- **nb\_tranches** *int*: Number of elements in the extrusion direction.

### 3.32 troisf

Description: Auxiliary class to extrude.

See also: objet\_lecture (32)

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.

### 3.33 extruder\_en20

Description: It does the same task as Extruder except a prism is cut in 20 instead of 3. The name of the boundaries will be *devant* and *derriere*. But you can change this name 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* (3.32): 0 Direction of the extrude operation.
- **nb\_tranches** *int*: Number of elements in the extrusion direction.

### 3.34 extruder\_en3

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

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

See also: extruder (3.31)

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* (3.32) for inheritance: Direction of the extrude operation.
- **nb\_tranches** *int* for inheritance: Number of elements in the extrusion direction.

### 3.35 end

Synonymous: **fin**

Description: Keyword which must complete the data file.

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

Usage:

**end**

### 3.36 }

Description: Block's end.

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

Usage:

}

### 3.37 imprimer\_flux

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

See also: [interprete \(3\)](#) `imprimer_flux_sum (3.39)`

Usage:

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

where

- **domain\_name** *str*: Name of the domain.
- **noms\_bord** *bloc\_lecture (3.38)*: Liste des noms des bords ex: { Bord1 Bord2 }

### 3.38 bloc\_lecture

Description: pour lire entre deux accolades

See also: objet\_lecture (32)

Usage:

**bloc\_lecture**

where

- **bloc\_lecture** *str*

### 3.39 imprimer\_flux\_sum

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

See also: imprimer\_flux (3.37)

Usage:

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

where

- **domain\_name** *str*: Name of the domain.
- **noms\_bord** *bloc\_lecture* (3.38): Liste des noms des bords ex: { Bord1 Bord2 }

### 3.40 integrer\_champ\_med

Description: his keyword is used to calculate a flow rate from a velocity MED field read before. The method is either debit\_total to calculate the flow rate on the whole surface, either integrale\_en\_z to calculate flow rates between  $z=z_{min}$  and  $z=z_{max}$  on nb\_tranche surfaces. The output file indicates first the flow rate for the whole surface and then lists for each tranche : the height z, the surface average value, the surface area and the flow rate. For the debit\_total method case, only one tranche is considered.

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

See also: interprete (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']: permet de choisir si l on veut l integrale suivant z ou sur toute la hauteur (debit\_total correspond a zmin=-DMAXFLOAT, ZMax=DMAXFLOAT, nb\_tranche=1)

- **zmin** *float*
- **zmax** *float*
- **nb\_tranche** *int*
- **fichier\_sortie** *str*: nom du fichier de sortie par défaut : integrale.

### 3.41 lata\_to\_med

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

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

Usage:

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

where

- **format** *format\_lata\_to\_med* ([3.42](#)): generated file post\_med.data use format (MED or MESHTV or LML keyword).
- **file** *str*: LATA file to convert to the new format.
- **file\_med** *str*: Name of file med.

### 3.42 format\_lata\_to\_med

Description: not\_set

See also: [objet\\_lecture \(32\)](#)

Usage:

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

where

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

### 3.43 lata\_to\_other

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

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

Usage:

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

where

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

### 3.44 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.

### 3.45 mailler

Description: The Mailler (Mesh) interpreter allows a Domain type object *domaine* to be meshed with objects *objet\_1*, *objet\_2*, etc...

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

Usage:

**mailler domaine bloc**

where

- **domaine** *str*: Name of domain.
- **bloc** *list\_bloc\_mailler (3.46)*: Instructions to mesh.

### 3.46 list\_bloc\_mailler

Description: List of block mesh.

See also: [listobj \(31.3\)](#)

Usage:

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

list of *mailler\_base (3.46.1)* separated with ,

#### 3.46.1 mailler\_base

Description: Basic class to mesh.

See also: [objet\\_lecture \(32\)](#) [pave \(3.46.2\)](#) [epsilon \(3.46.12\)](#) [domain \(3.46.13\)](#)

Usage:

#### 3.46.2 pave

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

See also: [mailler\\_base \(3.46.1\)](#)

Usage:

**pave name bloc list\_bord**

where



- **name** *str*: Name of the pave (block).
- **bloc** *bloc\_pave* (3.46.3): Definition of the pave (block).
- **list\_bord** *list\_bord* (3.46.4): Definition of boundaries of domain.

### 3.46.3 bloc\_pave

Description: Class to create a pave.

See also: [objet\\_lecture](#) (32)

Usage:

```
{
    [ Origine x1 x2 (x3)]
    [ longueurs x1 x2 (x3)]
    [ nombre_de_noeuds n1 n2 (n3)]
    [ facteurs x1 x2 (x3)]
    [ symx ]
    [ symy ]
    [ symz ]
    [ tanh float]
    [ tanh_dilatation int into [-1, 0, 1]]
    [ tanh_taille_premiere_maille float]
```

}

where

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

### 3.46.4 list\_bord

Description: The block sides.

See also: [listobj \(31.3\)](#)

Usage:

```
{ object1 object2 .... }  
list of bord_base \(3.46.5\)
```

### 3.46.5 **bord\_base**

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

See also: [objet\\_lecture \(32\)](#) [bord \(3.46.6\)](#) [raccord \(3.46.10\)](#) [internes \(3.46.11\)](#)

Usage:

### 3.46.6 **bord**

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

See also: [bord\\_base \(3.46.5\)](#)

Usage:

**bord nom defbord**

where

- **nom** *str*: Name of block side.
- **defbord** *defbord* [\(3.46.7\)](#): Definition of block side.

### 3.46.7 **defbord**

Description: Class to define an edge.

See also: [objet\\_lecture \(32\)](#) [defbord\\_2 \(3.46.8\)](#) [defbord\\_3 \(3.46.9\)](#)

Usage:

### 3.46.8 **defbord\_2**

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

See also: [\(3.46.7\)](#)

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*: Value minimal.
- **inf1** *str* into [*'<='*]: Less or equal sign.
- **dir2** *str* into [*'X'*, *'Y'*]: Edge is parallel to this direction.
- **inf2** *str* into [*'<='*]: Less or equal sign.
- **pos2\_max** *float*: Value maximal.

### 3.46.9 defbord\_3

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

See also: (3.46.7)

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*: Value minimal.
- **inf1** *str* into ['<=']: Less or equal sign.
- **dir2** *str* into ['X', 'Y']: Edge is parallel to this direction.
- **inf2** *str* into ['<=']: Less or equal sign.
- **pos2\_max** *float*: Value maximal.
- **pos3\_min** *float*: Value minimal.
- **inf3** *str* into ['<=']: Less or equal sign.
- **dir3** *str* into ['Y', 'Z']: Edge is parallel to this direction.
- **inf4** *str* into ['<=']: Less or equal sign.
- **pos3\_max** *float*: Value maximal.

### 3.46.10 raccord

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

See also: bord\_base (3.46.5)

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* (3.46.7): Definition of block side.

### 3.46.11 internes

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

Two boundaries with the same limitation conditions may be given the same name (whether or not they belong to the same block).

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

See also: bord\_base (3.46.5)

Usage:

**internes nom defbord**  
where

- **nom** *str*: Name of block side.
- **defbord** *defbord* (3.46.7): Definition of block side.

### 3.46.12 epsilon

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

See also: *mailler\_base* (3.46.1)

Usage:

**epsilon** *eps*

where

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

### 3.46.13 domain

Description: Class to reuse a domain.

See also: *mailler\_base* (3.46.1)

Usage:

**domain** *domain\_name*

where

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

## 3.47 mailerparallel

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

See also: *interpret* (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*

### 3.48 **modif\_bord\_to\_raccord**

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

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

Usage:

**modif\_bord\_to\_raccord** **domaine** **nom\_bord**

where

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

### 3.49 moyenne\_volumique

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

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

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 postraitements) N source\_field1 source\_field2 ... source\_fieldN
- **nom\_fichier\_post** *str*: indicates the filename where the result is written
- **format\_post** *str*: gives the fileformat for the result (by default : lata)
- **localisation** *str* into ['elem', 'som']: indicates where the convolution product should be computed: either on the elements or on the nodes of the destination domain.
- **fonction\_filtre** *bloc\_lecture* (3.38): 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.

### 3.50 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.

### 3.51 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).
- **p\_imposee\_aux\_faces** *str* into ['oui', 'non']: Pressure imposed at the faces (yes or no).

### 3.52 orientefacesbord

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

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

Usage:

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

where

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

### 3.53 partition

Synonymous: **decouper**

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

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

Usage:

**partition** **domaine** **bloc\_decouper**

where

- **domaine** *str*: Name of the domain to be cut.
- **bloc\_decouper** *bloc\_decouper* (3.54): Description how to cut a domain.

### 3.54 bloc\_decouper

Description: Auxiliary class to cut a domain.

See also: [objet\\_lecture](#) (32)

Usage:

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

}

where

- **Partition\_tool|partitionneur partitionneur\_deriv** (23): Defines the partitionning algorithm (the effective C++ object used is 'Partitionneur\_ALGORITHM\_NAME').
- **larg\_joint** *int*: This keyword specifies the thickness of the virtual ghost zone (data known by one processor though not owned by it). The default value is 1 and is generally correct for all algorithms except the QUICK convection scheme that require a thickness of 2. Since the 1.5.5 version, the VEF discretization imply also a thickness of 2 (except VEF P0). Any non-zero positive value can be used, but the amount of data to store and exchange between processors grows quickly with the thickness.
- **zones\_namelnom\_zones** *str*: Name of the files containing the different partition of the domain. The files will be :  
 name\_0001.Zones  
 name\_0002.Zones  
 ...  
 name\_000n.Zones. If this keyword is not specified, the geometry is not written on disc (you might just want to generate a 'ecrire\_decoupage' or 'ecrire\_lata').
- **ecrire\_decoupage** *str*: After having called the partitionning algorithm, the resulting partition is written on disc in the specified filename. See also [partitionneur Fichier\\_Decoupage](#). This keyword is useful to change the partition numbers (for example, to do manually the task of the keyword [Echange\\_domcut](#)): first, you write the partition into a file with the option [ecrire\\_decoupage](#). This file contains the zone number for each element's mesh. Then you can easily permute zone numbers in this file. Then read the new partition to create the .Zones files with the [Fichier\\_Decoupage](#) keyword.
- **ecrire\_lata** *str*
- **nb\_parts\_tot** *int*: Keyword to generates N .Zone files, instead of the default number M obtained after the 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.

### 3.55 pilote\_icoco

Description: not\_set

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

Usage:

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

where

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

### 3.56 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* ([3.57](#)): Surface and volume porosity values.

### 3.57 bloc\_lecture\_poro

Description: Surface and volume porosity values.

See also: `objet_lecture` (32)

Usage:

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

where

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

### 3.58 porosites\_champ

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

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

See also: `interpret` (3)

Usage:

```
porosites_champ pb ch  
where
```

- **pb** *str*: Name of the problem to which the sub-area is attached.
- **ch** *champ\_base* (15.1): field used to define the porosity field

### 3.59 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', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med']  
    [ filefichier str ]  
    [ domaine str ]  
    [ domaines bloc_lecture ]  
    [ joints_non_postraites int into [0, 1] ]  
    [ binaire int into [0, 1] ]  
    [ ecrire_frontiere int into [0, 1] ]  
}
```

where

- **format** *str* into ['lml', 'meshtv', 'lata', 'lata\_v1', 'lata\_v2', 'med']: File format.
- **filefichier** *str*: The file name can be changed with the `fichier` option.
- **domaine** *str*: Name of domain
- **domaines** *bloc\_lecture* (3.38): 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)

### 3.60 precisiongeom

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

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

Usage:

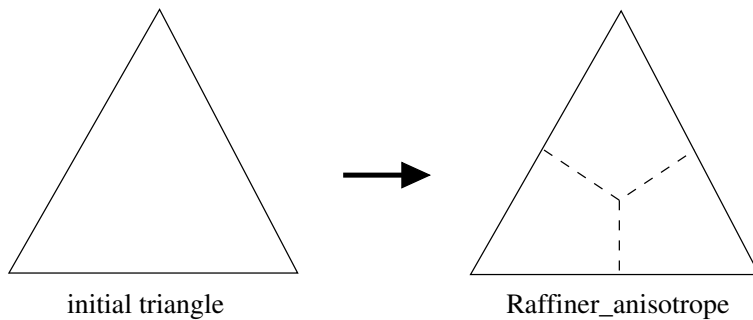
**precisiongeom** **precision**

where

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

### 3.61 raffiner\_anisotrope

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



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

Usage:

**raffiner\_anisotrope** **domain\_name**

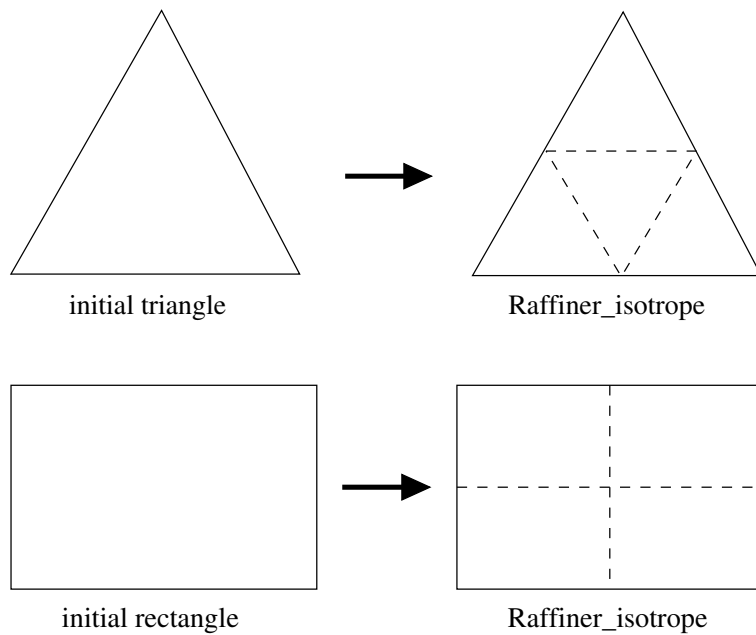
where

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

### 3.62 raffiner\_isotrope

Synonymous: **raffiner\_simplexes**

Description: To allows to cut triangles/quadrangles or tetrahedral/hexaedras elements respectively in 4 or 8 new ones by defining new summits located at the middle of edges (and center of faces and elements



for quadrangles and hexaedra). Such a cut preserves the shape of original elements (isotropic).

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

Usage:

**raffiner\_isotrope** **domain\_name**

where

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

### 3.63 read

Synonymous: **lire**

Description: Interpreter to read the object objet 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.

### 3.64 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 lire\_fichier dom filename (where dom is the name of the meshed domain).

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

See also: [interpret \(3\)](#) [read\\_unsupported\\_ascii\\_file\\_from\\_icem \(3.67\)](#) [read\\_file\\_binary \(3.65\)](#)

Usage:

**read\_file name\_obj filename**

where

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

### 3.65 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 \(3.64\)](#)

Usage:

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

where

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

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

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

Description: not\_set

See also: [read\\_file \(3.64\)](#)

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.

### 3.68 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 Lire\_Med to read the mesh then use Create\_domain\_from\_sous\_zone keyword.

NB: If the MED file contains one or several subzone defined as a group of volumes, then Lire\_MED will read it and will create two files domain\_name\_ssz.geo and domain\_name\_ssz\_par.geo defining the subzones for sequential and/or parallel calculations. These subzones will be read in sequential in the datafile by including (after Lire\_Med keyword) something like:

Lire\_Med ....

Read\_file domain\_name\_ssz.geo ;

During the parallel calculation, you will include something:

Scatter { ... }

Read\_file domain\_name\_ssz\_par.geo ;

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

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

### 3.69 orienter\_simplexes

Synonymous: **rectify\_mesh**

Description: Keyword to raffine a mesh

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

Usage:

**orienter\_simplexes** **domain\_name**

where

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

### 3.70 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.

### 3.71 refine\_mesh

Description: not\_set

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

Usage:

**refine\_mesh domaine**

where

- **domaine** *str*

### 3.72 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

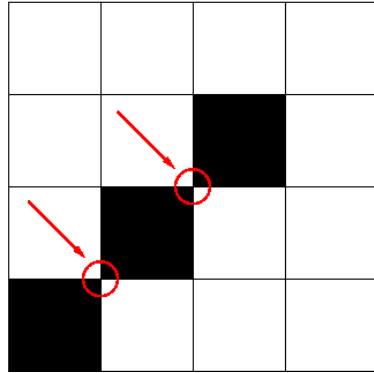
- **domaine** *str*: Name of domain
- **new\_bord** *str*: Name of the new boundary
- **bords** *bloc\_lecture (3.38)*: { Bound1 Bound2 }

### 3.73 remove\_elem

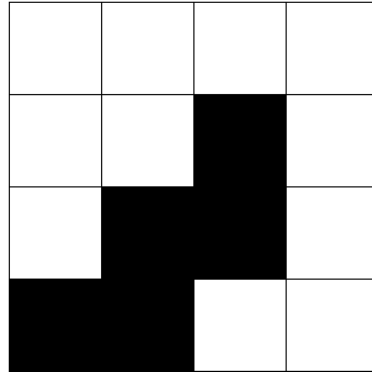
Description: Keyword to remove element from a VDF mesh (named domaine\_name), either from an explicit list of elements or from a geometric condition defined by a condition  $f(x,y)>0$  in 2D and  $f(x,y,z)>0$  in 3D. All the new borders generated are gathered in one boundary called : newBord (to rename 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  $0.2 * 0.2 - (x - 0.5)^2 - (y - 0.5)^2 > 0$  }

UNCORRECT – 2 SINGULAR NODES



CORRECT



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

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

Usage:

**remove\_elem** **domaine** **bloc**

where

- **domaine** *str*: Name of domain
- **bloc** *remove\_elem\_bloc* ([3.74](#))

### 3.74 remove\_elem\_bloc

Description: not\_set

See also: [objet\\_lecture \(32\)](#)

Usage:

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

where

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

### 3.75 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.

### 3.76 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.

### 3.77 reorienter\_tetraedres

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

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

Usage:

**reorienter\_tetraedres** **domain\_name**

where

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

### 3.78 reorienter\_triangles

Description: `not_set`

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

Usage:

**reorienter\_triangles** **domain\_name**

where

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

### 3.79 reordonner

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

`Lire_Fichier dom fichier.geom`

`Reordonner dom`

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

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

Usage:

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

where

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

### 3.80 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)

### 3.81 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 \(3.82\)](#) [scattermed \(3.83\)](#)

Usage:

**scatter** **file** **domaine**

where

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

### 3.82 scatterformatte

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

See also: [scatter \(3.81\)](#)

Usage:

**scatterformatte** **file** **domaine**

where

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

### 3.83 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 ([3.81](#))

Usage:

**scattermed file domaine**

where

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

### 3.84 solve

Synonymous: **resoudre**

Description: Interpreter to start calculation with TRUST.

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

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

Usage:

**solve pb**

where

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

### 3.85 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* ([3.86](#)): { Boundary\_name1 Boundary\_name2 }

### 3.86 list\_nom

Description: List of name.

See also: listobj ([31.3](#))

Usage:

{ object1 object2 .... }

list of *nom\_anonyme* ([22.1](#))

### 3.87 system

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

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

Usage:

**system cmd**

where

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

### 3.88 test\_solveur

Description: To test several solvers

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

Usage:

**test\_solveur {**

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

**}**

where

- **fichier\_secmem** *str*: Filename containing the second member B
- **fichier\_matrice** *str*: Filename containing the matrix A
- **fichier\_solution** *str*: Filename containing the solution x
- **nb\_test** *int*: Number of tests to measure the time resolution (one preconditionnement)
- **impr** : To print the convergence solver
- **solveur** *solveur\_sys\_base* (9.12): 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

### 3.89 testeur

Description: not\_set

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

Usage:

**testeur data**

where

- **data** *bloc\_lecture* (3.38)

### 3.90 testeur\_medcoupling

Description: not\_set

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

Usage:

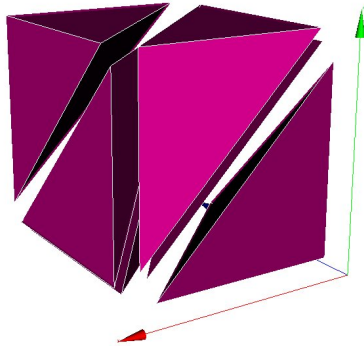
**testeur\_medcoupling pb\_name field\_name**

where

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

### 3.91 tetraedriser

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



See also: [interprete \(3\)](#) [tetraedriser\\_homogene \(3.92\)](#) [tetraedriser\\_homogene\\_fin \(3.94\)](#) [tetraedriser\\_homogene\\_compact \(3.93\)](#) [tetraedriser\\_par\\_prisme \(3.95\)](#)

Usage:

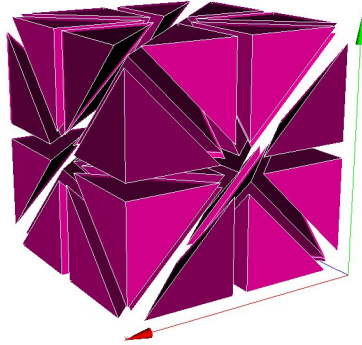
**tetraedriser domain\_name**

where

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

### 3.92 tetraedriser\_homogene

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



See also: tetraedriser ([3.91](#))

Usage:

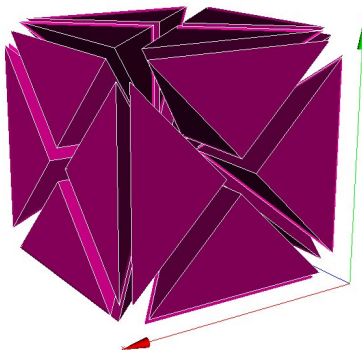
**tetraedriser\_homogene** **domain\_name**

where

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

### 3.93 tetraedriser\_homogene\_compact

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



See also: tetraedriser ([3.91](#))

Usage:

**tetraedriser\_homogeneous\_compact domain\_name**

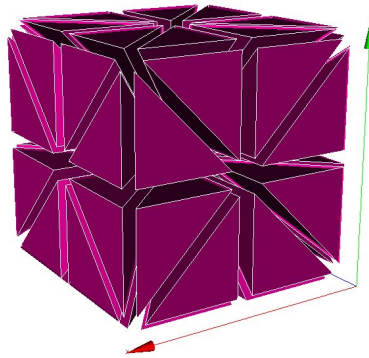
where

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

### 3.94 tetraedriser\_homogeneous\_fin

Description: Tetraedriser\_homogeneous\_fin is the recommended option to tetrahedralise blocks. As an extension (subdivision) of Tetraedriser\_homogeneous\_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\_homogeneous\_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 ([3.91](#))

Usage:

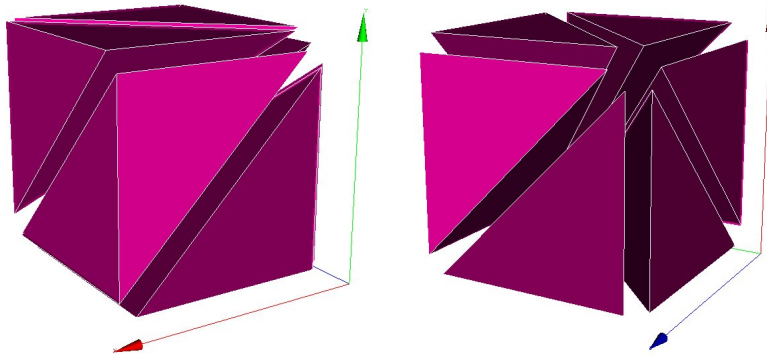
**tetraedriser\_homogeneous\_fin domain\_name**

where

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

### 3.95 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 \(3.91\)](#)

Usage:

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

where

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

### 3.96 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*

### 3.97 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 \(3.99\)](#) [trianguler\\_fin \(3.98\)](#)

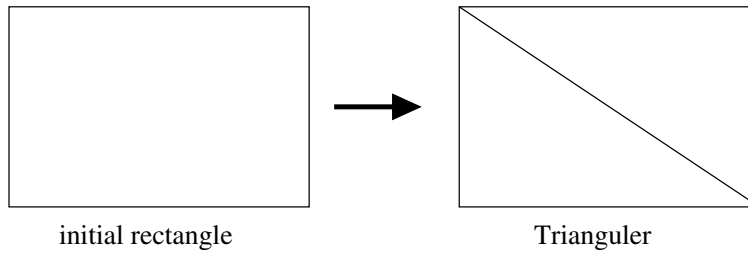
Usage:

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

where

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



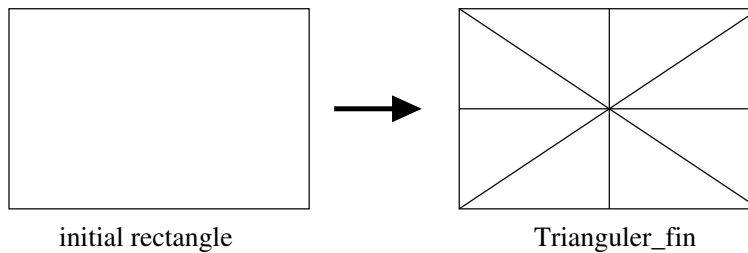


### 3.98 **triangler\_fin**

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

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

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



See also: [triangler \(3.97\)](#)

Usage:

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

where

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

### 3.99 **triangler\_h**

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

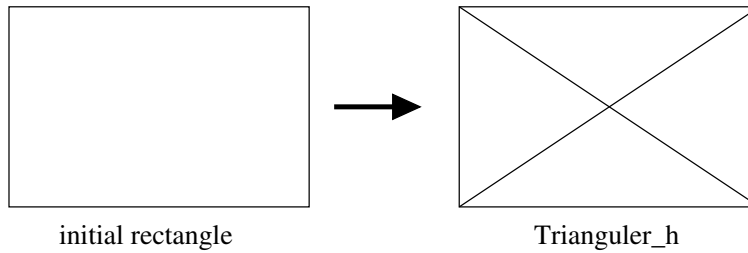
See also: [triangler \(3.97\)](#)

Usage:

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

where

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



### 3.100 **verifier\_qualite\_raffinements**

Description: not\_set

See also: interpret (3)

Usage:

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

- **domain\_names** *vect\_nom* (3.101)

### 3.101 **vect\_nom**

Description: Vect of name.

See also: listobj (31.3)

Usage:

n object1 object2 ....  
list of *nom\_anonyme* (22.1)

### 3.102 **verifier\_simplexes**

Description: Keyword to raffine a simplexes

See also: interpret (3)

Usage:

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

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

### 3.103 **verfiercoin**

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

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

See also: interpret (3)

Usage:

**verifiercoin dom**

where

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

### 3.104 ecrire

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

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

Usage:

**ecrire name\_obj**

where

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

### 3.105 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: [interpret \(3\)](#) [ecrire\\_fichier\\_formatte \(3.20\)](#)

Usage:

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

where

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

### 3.106 ecrire\_med

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

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

Usage:

**ecrire\_med 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 (33) Pb\_base (4.1) probleme\_couple (4.7) pbc\_med (4.31)

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 Discretiser should have already be used to read the object.

See also: pb\_gen\_base (4) pb\_thermohydraulique (4.19) pb\_hydraulique (4.12) pb\_hydraulique\_turbulent (4.17) pb\_thermohydraulique\_turbulent (4.27) pb\_conduction (4.11) pb\_thermohydraulique\_qc (4.24) pb\_thermohydraulique\_turbulent\_qc (4.28) pb\_hydraulique\_concentration (4.13) pb\_hydraulique\_concentration\_turbulent (4.15) pb\_thermohydraulique\_concentration (4.20) pb\_thermohydraulique\_concentration\_turbulent (4.22) pb\_avec\_passif (4.9) pb\_post (4.18) problem\_read\_generic (4.33)

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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be

restarted, values for the tinit (see `schema_temps_base`) time fields are taken from the `name_file` file. If there is no backup corresponding to this time in the `name_file`, TRUST exits in error.

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

## 4.2 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', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med']]
  [ fields/champs champs_posts]
  [ statistiques stats_posts]
  [ fichier str]
  [ statistiques_en_serie stats_serie_posts]
  [ interfaces champs_posts]
}
```

where

- **definition\_champs** *definition\_champs* (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', 'meshtv', 'lata', 'lata\_v1', 'lata\_v2', 'med'] for inheritance: This optional parameter specifies the format of the output file. The basename used for the output file is the basename of the data file. For the `fmt` parameter, choices are `lml`, `lata`, or `meshtv`. A short description of each format can be found below. The default value is `lml`.
- **fields/champs** *champs\_posts* (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` (31.3)

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 \(32\)](#)

Usage:

**name champ\_generique**

where

- **name** *str*: The name of the new created field.
- **champ\_generique** *champ\_generique\_base* [\(7\)](#)

#### 4.2.3 sondes

Description: List of probes.

See also: [listobj \(31.3\)](#)

Usage:

{ object1 object2 .... }

list of *sonde* [\(4.2.4\)](#)

#### 4.2.4 sonde

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

See also: [objet\\_lecture \(32\)](#)

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 PINC 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 \(32\)](#) [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 \(31.3\)](#)

Usage:

n object1 object2 ....

list of *un\_point* [\(3.7.3\)](#)

#### 4.2.8 point

Description: Point as class-daughter of Points.

See also: [points \(4.2.6\)](#)

Usage:

**point points**

where

- **points** *listpoints* [\(4.2.7\)](#): Probe points.

#### 4.2.9 segmentpoints

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

See also: [points \(4.2.6\)](#)

Usage:

**segmentpoints points**

where

- **points** *listpoints* [\(4.2.7\)](#): Probe points.

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

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

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

Usage:

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

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

#### 4.2.11 position\_like

Description: Keyword to define a probe at the same position of another probe named autre\_sonde.

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

Usage:

**position\_like** **autre\_sonde**  
where

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

#### 4.2.12 segment

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

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

Usage:

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

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point\_deb** *un\_point* ([3.7.3](#)): First outer probe segment point.
- **point\_fin** *un\_point* ([3.7.3](#)): 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* ([3.7.3](#)): First point defining the angle. This angle should be positive.
- **point\_fin** *un\_point* ([3.7.3](#)): Second point defining the angle. This angle should be positive.
- **point\_fin\_2** *un\_point* ([3.7.3](#)): 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* ([3.7.3](#)): Point of origin.
- **point\_fin** *un\_point* ([3.7.3](#)): Point defining the first direction (from point of origin).
- **point\_fin\_2** *un\_point* ([3.7.3](#)): Point defining the second direction (from point of origin).
- **point\_fin\_3** *un\_point* ([3.7.3](#)): 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* ([3.7.3](#)): 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 theta1 and theta2 (angles given in degrees).
- **point\_deb** *un\_point* ([3.7.3](#)): 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 (32)

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 (31.3)

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 (32)

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 (speed)**, **Pression (pressure)**, **Temperature**, **Concentration**,...

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

Example:

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

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

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

See also: [objet\\_lecture \(32\)](#)

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 \(31.3\)](#)

Usage:

{ object1 object2 .... }

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

#### 4.2.22 stat\_post\_deriv

Description: not\_set

See also: [objet\\_lecture \(32\)](#) [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 (speed)**, **Pression (pressure)**, **Temperature**, **Concentration**,...

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

*Example*:

```
Statistiques_en_serie Dt_integr dtst {
Moyenne Pression
}
```

Will calculate and write every dtst seconds the mean value:

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

See also: objet\_lecture (32)

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 (31.3)

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 (32)

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 (31.3)

Usage:

{ object1 object2 .... }

list of *nom\_postraitement* (4.4.1)

#### 4.4.1 nom\_postraitement

Description:

See also: objet\_lecture (32)

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 (32) post\_processing (4.4.3)

Usage:

### 4.4.3 post\_processing

Synonymous: **postraitement**

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

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

Usage:

```
post_processing {  
    [ definition_champs definition_champs]  
    [ Probeslsondes sondes]  
    [ domaine str]  
    [ format str into ['lml', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med']]  
    [ 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', 'meshtv', 'lata', 'lata\_v1', 'lata\_v2', 'med']: This optional parameter specifies the format of the output file. The basename used for the output file is the basename of the data file. For the fmt parameter, choices are lml, lata, or meshtv. A short description of each format can be found below. The default value is lml.
- **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.

## 4.5 liste\_post

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

See also: `listobj` (31.3)

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` ([32](#))

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` ([32](#))

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` ([32](#))

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` ([32](#))

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

`Probleme_Couple pbc`

`Lire pbc { groupes { { pb1 , pb2 , pb3 , pb4 } } }`

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

Each problem is resolved in a domain.

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

See also: `pb_gen_base` (4)

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` (31.3)

Usage:

{ object1 , object2 .... }

list of *list\_un\_pb* (31.1) separated with ,

## 4.9 pb\_avec\_passif

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

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

See also: `Pb_base` (4.1) `pb_thermohydraulique_concentration_turbulent_scalaires_passifs` (4.23) `pb_thermohydraulique_concentration_scalaires_passifs` (4.21) `pb_thermohydraulique_turbulent_scalaires_passifs` (4.30) `pb_thermohydraulique_scalaires_passifs` (4.26) `pb_hydraulique_concentration_turbulent_scalaires_passifs` (4.16) `pb_hydraulique_concentration_scalaires_passifs` (4.14) `pb_thermohydraulique_qc_fraction_massique` (4.25) `pb_thermohydraulique_turbulent_qc_fraction_massique` (4.29)

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

## 4.10 listeqn

Description: List of equations.

See also: listobj (31.3)

Usage:

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

list of *eqn\_base* (5.18)

## 4.11 pb\_conduction

Description: Resolution of the heat equation.

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

## 4.12 pb\_hydraulique

Description: Resolution of the NAVIER STOKES equations.

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

See also: `Pb_base` (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* (5.23): 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

### 4.13 pb\_hydraulique\_concentration

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

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

See also: `Pb_base` (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* (5.23): NAVIER STOKES equations.
- **convection\_diffusion\_concentration** *convection\_diffusion\_concentration* (5.11): Constituent transportation 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

#### 4.14 pb\_hydraulique\_concentration\_scalaires\_passifs

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

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

See also: pb\_avec\_passif (4.9)

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* (5.23): NAVIER STOKES equations.
- **convection\_diffusion\_concentration** *convection\_diffusion\_concentration* (5.11): Constituent transportation equations (concentration diffusion convection).
- **equations\_scalaires\_passifs** *listeqn* (4.10) 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

#### 4.15 pb\_hydraulique\_concentration\_turbulent

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

Keyword Discretiser should have already be used to read the object.  
See also: Pb\_base (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_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* (5.24): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection\_diffusion\_concentration\_turbulent** *convection\_diffusion\_concentration\_turbulent* (5.12): Constituent transportation 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).



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

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

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

See also: pb\_avec\_passif (4.9)

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* (5.24): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection\_diffusion\_concentration\_turbulent** *convection\_diffusion\_concentration\_turbulent* (5.12): Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **equations\_scalaires\_passifs** *listeqn* (4.10) 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on



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

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

## 4.17 pb\_hydraulique\_turbulent

Description: Resolution of NAVIER STOKES equations with turbulence modelling.

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

See also: Pb\_base (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* (5.24): 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N \leq P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the

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

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

## 4.18 pb\_post

Description: not\_set

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

## 4.19 pb\_thermohydraulique

Description: Resolution of thermohydraulic problem.

Keyword Discretiser should have already be 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* (5.23): NAVIER STOKES equations.
- **convection\_diffusion\_temperature** *convection\_diffusion\_temperature* (5.15): 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.20 pb\_thermohydraulique\_concentration

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

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

See also: Pb\_base (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* (5.23): NAVIER STOKES equations.
- **convection\_diffusion\_concentration** *convection\_diffusion\_concentration* (5.11): Constituent transportation equations (concentration diffusion convection).
- **convection\_diffusion\_temperature** *convection\_diffusion\_temperature* (5.15): 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.21 pb\_thermohydraulique\_concentration\_scalaires\_passifs

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

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

See also: pb\_avec\_passif (4.9)

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* (5.23): NAVIER STOKES equations.
- **convection\_diffusion\_concentration** *convection\_diffusion\_concentration* (5.11): Constituent transportation equations (concentration diffusion convection).
- **convection\_diffusion\_temperature** *convection\_diffusion\_temperature* (5.15): Energy equations (temperature diffusion convection).
- **equations\_scalaires\_passifs** *listeqn* (4.10) 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file

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

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

## 4.22 pb\_thermohydraulique\_concentration\_turbulent

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

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

See also: Pb\_base (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* (5.24): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection\_diffusion\_concentration\_turbulent** *convection\_diffusion\_concentration\_turbulent* (5.12): Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection\_diffusion\_temperature\_turbulent** *convection\_diffusion\_temperature\_turbulent* (5.17): 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N \leq P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

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

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

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

See also: pb\_avec\_passif (4.9)

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* (5.24): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection\_diffusion\_concentration\_turbulent** *convection\_diffusion\_concentration\_turbulent* (5.12): Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection\_diffusion\_temperature\_turbulent** *convection\_diffusion\_temperature\_turbulent* (5.17): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations\_scalaires\_passifs** *listeqn* (4.10) 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N \leq P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.24 pb\_thermohydraulique\_qc

Description: Resolution of thermohydraulic problem under small Mach number.

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

masse\_volumique : density

enthalpie : enthalpy

pression : reduced pressure

pression\_tot : total pressure.

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

See also: Pb\_base (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* (5.19): NAVIER STOKES equations under small Mach number.
- **convection\_diffusion\_chaleur\_qc** *convection\_diffusion\_chaleur\_qc* (5.8): Energy equation under small 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.25 pb\_thermohydraulique\_qc\_fraction\_massique

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

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

See also: pb\_avec\_passif (4.9)

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* (5.19): NAVIER STOKES equations under small Mach number.
- **convection\_diffusion\_chaleur\_qc** *convection\_diffusion\_chaleur\_qc* (5.8): Energy equation under small Mach number.
- **equations\_scalaires\_passifs** *listeqn* (4.10) 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N \leq P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.26 pb\_thermohydraulique\_scalaires\_passifs

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

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

See also: pb\_avec\_passif (4.9)

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* (5.23): NAVIER STOKES equations.
- **convection\_diffusion\_temperature** *convection\_diffusion\_temperature* (5.15): Energy equations (temperature diffusion convection).
- **equations\_scalaires\_passifs** *listeqn* (4.10) 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N < P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.27 pb\_thermohydraulique\_turbulent

Description: Resolution of thermohydraulic problem, with turbulence modelling.

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

See also: `Pb_base` (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* (5.24): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection\_diffusion\_temperature\_turbulent** *convection\_diffusion\_temperature\_turbulent* (5.17): 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N <> P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

## 4.28 pb\_thermohydraulique\_turbulent\_qc

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

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

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

See also: Pb\_base (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* (5.26): NAVIER STOKES equations under small Mach number as well as the associated turbulence model equations.
- **convection\_diffusion\_chaleur\_turbulent\_qc** *convection\_diffusion\_chaleur\_turbulent\_qc* (5.10): Energy equation under small 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N \leq P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

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

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

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

See also: pb\_avec\_passif (4.9)

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* (5.26): NAVIER STOKES equations under small Mach number as well as the associated turbulence model equations.
- **convection\_diffusion\_chaleur\_turbulent\_qc** *convection\_diffusion\_chaleur\_turbulent\_qc* (5.10): Energy equation under small Mach number as well as the associated turbulence model equations.
- **equations\_scalaires\_passifs** *listeqn* (4.10) 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on

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

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

### 4.30 pb\_thermohydraulique\_turbulent\_scalaires\_passifs

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

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

See also: pb\_avec\_passif (4.9)

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* (5.24): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection\_diffusion\_temperature\_turbulent** *convection\_diffusion\_temperature\_turbulent* (5.17): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations\_scalaires\_passifs** *listeqn* (4.10) 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 restarting 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 restart a calculation based on the name\_file file (see the class format\_file). If format\_reprise is xyz, the name\_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ( $N \leq P$ ) processors. Should the calculation be restarted, values for the tinit (see schema\_temps\_base) time fields are taken from the name\_file file. If there is no backup corresponding to this time in the name\_file, TRUST exits in error.
- **resume\_last\_time** *format\_file* (4.6) for inheritance: Keyword to restart a calculation based on the name\_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

### 4.31 pbc\_med

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

See also: pb\_gen\_base (4)

Usage:

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

where

- **list\_info\_med** *list\_info\_med* (4.32)

### 4.32 list\_info\_med

Description: not\_set

See also: listobj (31.3)

Usage:

{ object1 , object2 .... }

list of *info\_med* (4.32.1) separated with ,

#### 4.32.1 info\_med

Description: not\_set

See also: objet\_lecture (32)

Usage:

**file\_med domaine pb\_post**

where

- **file\_med** *str*: Name of file med.
- **domaine** *str*: Name of domain.
- **pb\_post** *pb\_post* (4.18)



### 4.33 problem\_read\_generic

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

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

See also: `Pb_base` (4.1)

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

## 5 mor\_eqn

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

See also: `objet_u` (33) `eqn_base` (5.18)

Usage:

### 5.1 conduction

Description: Heat equation.

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

See also: `eqn_base` (5.18)

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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if `equation_non_resolue` keyword is used. Example: The Navier Stokes is not solved between time t0 and t1.

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

## 5.2 bloc\_diffusion

Description: not\_set

See also: objet\_lecture (32)

Usage:

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

where

- **aco** *str* into [' ']: Open accodance sign.
- **opérateur** *diffusion\_deriv* (5.2.1): if none is specified, the diffusive scheme used is an order 2 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 [' ']: Closed accodance sign.

### 5.2.1 diffusion\_deriv

Description: not\_set

See also: objet\_lecture (32) negligable (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 negligable

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 [*'default\_bar'*]: equivalent to `grad_Ubar 1 nu 1 nut 1 nu_transp 1 nut_transp 1 filtrer_resu 1`
- **bloc\_diffusion\_standard** *bloc\_diffusion\_standard* (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` (32)

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* (3.38)

### 5.2.9 op\_implicite

Description: `not_set`

See also: `objet_lecture` (32)

Usage:

**implicite mot solveur**

where

- **implicite** *str* into ['implicite']
- **mot** *str* into ['solveur']
- **solveur** *solveur\_sys\_base* (9.12)

### 5.3 condinits

Description: Initial conditions.

See also: [objet\\_lecture \(32\)](#)

Usage:

**aco condinit acof**

where

- **aco** *str* into [' ']: Open accodance sign.
- **condinit** *condinit* ([5.3.1](#)): *CI*
- **acof** *str* into [' ']: Closed accodance sign.

#### 5.3.1 condinit

Description: Initial condition.

See also: [objet\\_lecture \(32\)](#)

Usage:

**nom ch**

where

- **nom** *str*: Name of initial condition field.
- **ch** *champ\_base* ([15.1](#)): Type field and the initial values.

### 5.4 condlims

Description: Boundary conditions.

See also: [listobj \(31.3\)](#)

Usage:

{ object1 object2 .... }

list of *condlimlu* ([5.4.1](#))

#### 5.4.1 condlimlu

Description: Boundary condition specified.

See also: [objet\\_lecture \(32\)](#)

Usage:

**bord cl**

where

- **bord** *str*: Name of the edge where the boundary condition applies.
- **cl** *condlim\_base* ([11](#)): Boundary condition at the boundary called bord (edge).

## 5.5 sources

Description: The sources.

See also: [listobj \(31.3\)](#)

Usage:

{ object1 , object2 .... }

list of *source\_base* ([27](#)) separated with ,

## 5.6 ecrire\_fichier\_xyz\_valeur\_param

Description: not\_set

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

See also: [listobj \(31.3\)](#)

Usage:

n object1 , object2 ....

list of *ecrire\_fichier\_xyz\_valeur\_item* ([5.6.1](#)) separated with ,

### 5.6.1 ecrire\_fichier\_xyz\_valeur\_item

Description: To write the values of a field for some boundaries in a text file.

The name of the files is pb\_name\_field\_name\_time.dat

Several *Ecrire\_fichier\_xyz\_valeur* keywords may be written into an equation to write several fields. This kind of files may be read by *Champ\_don\_lu* or *Champ\_front\_lu* for example.

See also: [objet\\_lecture \(32\)](#)

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* ([5.6.2](#)): to post-process only on some boundaries

### 5.6.2 bords\_ecrire

Description: not\_set

See also: [objet\\_lecture \(32\)](#)

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.

## 5.7 parametre\_equation\_base

Description: Basic class for parametre\_equation

See also: objet\_lecture (32) parametre\_diffusion\_implicite (5.7.1) parametre\_implicite (5.7.2)

Usage:

### 5.7.1 parametre\_diffusion\_implicite

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

See also: parametre\_equation\_base (5.7)

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.

### 5.7.2 parametre\_implicite

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

See also: parametre\_equation\_base (5.7)

Usage:

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



- **seuil\_convergence\_implicit** *float*: Keyword to change for this equation only the value of `seuil_convergence_implicit` used in the implicit scheme.
- **seuil\_convergence\_solveur** *float*: Keyword to change for this equation only the value of `seuil_convergence_solveur` used in the implicit scheme
- **solveur** *solveur\_sys\_base* (9.12): 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)*).

## 5.8 convection\_diffusion\_chaleur\_qc

Description: Energy equation under small Mach number.

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

See also: `eqn_base` (5.18) `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* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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 : `pname_fieldname_[boundaryname]_time.dat`

- **ecrire\_fichier\_xyz\_valeur\_bin** *ecrire\_fichier\_xyz\_valeur\_param* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Example: The Navier Stokes is not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.9 bloc\_convection

Description: not\_set

See also: objet\_lecture (32)

Usage:

**aco operateur acof**

where

- **aco** *str* into [' ']: Open accodance sign.
- **operateur** *convection\_deriv* (5.9.1)
- **acof** *str* into [' ']: Closed accodance sign.

### 5.9.1 convection\_deriv

Description: not\_set

See also: objet\_lecture (32) [amont](#) (5.9.2) [amont\\_old](#) (5.9.3) [centre](#) (5.9.4) [centre4](#) (5.9.5) [centre\\_old](#) (5.9.6) [di\\_l2](#) (5.9.7) [ef](#) (5.9.8) [muscl3](#) (5.9.10) [ef\\_stab](#) (5.9.11) [generic](#) (5.9.14) [kquick](#) (5.9.15) [muscl](#) (5.9.16) [muscl\\_old](#) (5.9.17) [muscl\\_new](#) (5.9.18) [negligeable](#) (5.9.19) [quick](#) (5.9.20) [btd](#) (5.9.21) [supg](#) (5.9.22)

Usage:

**convection\_deriv**

### 5.9.2 amont

Description: Keyword for upwind scheme 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](#) (5.9.1)

Usage:

**amont**

### 5.9.3 **amont\_old**

Description: not\_set

See also: convection\_deriv ([5.9.1](#))

Usage:

**amont\_old**

### 5.9.4 **centre**

Description: not\_set

See also: convection\_deriv ([5.9.1](#))

Usage:

**centre**

### 5.9.5 **centre4**

Description: not\_set

See also: convection\_deriv ([5.9.1](#))

Usage:

**centre4**

### 5.9.6 **centre\_old**

Description: not\_set

See also: convection\_deriv ([5.9.1](#))

Usage:

**centre\_old**

### 5.9.7 **di\_l2**

Description: not\_set

See also: convection\_deriv ([5.9.1](#))

Usage:

**di\_l2**

### 5.9.8 **ef**

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

Convection { EF transportant\_bar val transporte\_bar val antisym val filtrer\_resu val }

This scheme is 2nd order accuracy (and get better the property of kinetic energy conservation). Due to possible problems of instabilities phenomena, this scheme has to be coupled with stabilisation process (see Source\_Qdm\_lambdaup). These two last data are equivalent from a theoretical point of view in variationnal

writing to :  $\text{div}((u \cdot \text{grad } ub, vb) - (u \cdot \text{grad } vb, ub))$ , where  $vb$  corresponds to the filtered reference test functions.

Remark:

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

See also: `convection_deriv` ([5.9.1](#))

Usage:

**ef** [ **mot1** ] [ **bloc\_ef** ]

where

- **mot1** *str* into [*'default\_bar'*]: equivalent to `transportant_bar 0 transporte_bar 1 filtrer_resu 1 antisym 1`
- **bloc\_ef** *bloc\_ef* ([5.9.9](#))

### 5.9.9 bloc\_ef

Description: `not_set`

See also: `objet_lecture` ([32](#))

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*]

### 5.9.10 muscl3

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

See also: `convection_deriv` ([5.9.1](#))

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).

### 5.9.11 ef\_stab

Description: Keyword for a VEF convective scheme.

See also: convection\_deriv ([5.9.1](#))

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  $\text{div}(\mathbf{TU}) - \mathbf{T} \text{div} \mathbf{U} (= \mathbf{U} \text{grad} \mathbf{T})$ .
- **old** : To use old version of EF\_stab scheme (default no).
- **volumes\_etendus** : Option for the scheme to use the extended volumes (default, yes).
- **volumes\_non\_etendus** : Option for the scheme to not use the extended volumes (default, no).
- **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* ([5.9.12](#)): 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.

### 5.9.12 listsous\_zone\_valeur

Description: List of groups of two words.

See also: listobj ([31.3](#))

Usage:

n object1 object2 ....

list of *sous\_zone\_valeur* ([5.9.13](#))

### 5.9.13 sous\_zone\_valeur

Description: Two words.

See also: objet\_lecture ([32](#))

Usage:

**sous\_zone valeur**

where

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

#### 5.9.14 generic

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

Examples:

```
convection { generic amount }
convection { generic muscl minmod 1 }
convection { generic muscl vanleer 2 }
```

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

See also: convection\_deriv ([5.9.1](#))

Usage:

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

where

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

#### 5.9.15 kquick

Description: not\_set

See also: convection\_deriv ([5.9.1](#))

Usage:

**kquick**

#### 5.9.16 muscl

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

See also: convection\_deriv ([5.9.1](#))

Usage:

**muscl**

#### 5.9.17 muscl\_old

Description: not\_set

See also: convection\_deriv ([5.9.1](#))

Usage:

**muscl\_old**

#### **5.9.18 muscl\_new**

Description: not\_set

See also: convection\_deriv ([5.9.1](#))

Usage:

**muscl\_new**

#### **5.9.19 negligeable**

Description: suppresses the convection operator.

See also: convection\_deriv ([5.9.1](#))

Usage:

**negligeable**

#### **5.9.20 quick**

Description: not\_set

See also: convection\_deriv ([5.9.1](#))

Usage:

**quick**

#### **5.9.21 btd**

Description: not\_set

See also: convection\_deriv ([5.9.1](#))

Usage:

**btd** {

**btd** *float*

**facteur** *float*

}

where

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

### 5.9.22 supg

Description: not\_set

See also: convection\_deriv (5.9.1)

Usage:

```
supg {  
    facteur float  
}  
where  
    • facteur float
```

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

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

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

See also: convection\_diffusion\_chaleur\_qc (5.8)

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* (21): Turbulence model for the energy equation.
- **mode\_calcul\_convection** *str* into ['ancien', 'divuT\_moins\_Tdivu', 'divrhout\_moins\_Tdivrhout']  
for inheritance: Option to set the form of the convective operator  
divrhout\_moins\_Tdivrhout (the default since 1.6.8):  $\rho \cdot u \cdot \text{grad} T = \text{div}(\rho \cdot u \cdot T) - T \text{div}(\rho \cdot u \cdot 1)$   
ancien:  $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \cdot \text{div}(u)$   
divuT\_moins\_Tdivu :  $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \text{div}(u \cdot 1)$
- **convection** *bloc\_convection* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.11 convection\_diffusion\_concentration

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

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

See also: eqn\_base (5.18) convection\_diffusion\_concentration\_turbulent (5.12)

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_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

- **nom\_inconnue** *str*: Keyword Nom\_inconnue will rename the unknown of this equation with the given name. In the postprocessing part, the concentration field will be accessible with this name. This is usefull if you want to track more than one concentration (otherwise, only the concentration field in the first concentration equation can be accessed).
- **masse\_molaire** *float*
- **alias** *str*

- **convection** *bloc\_convection* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Example: The Navier Stokes is not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.12 convection\_diffusion\_concentration\_turbulent

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

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

See also: convection\_diffusion\_concentration (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* (21): Turbulence model to be used in the constituent transportation equations. The only model currently available is Schmidt.
- **nom\_inconnue** *str* for inheritance: Keyword Nom\_inconnue will rename the unknown of this equation with the given name. In the postprocessing part, the concentration field will be accessible with this name. This is useful if you want to track more than one concentration (otherwise, only the concentration field in the first concentration equation can be accessed).
- **masse\_molaire** *float* for inheritance
- **alias** *str* for inheritance
- **convection** *bloc\_convection* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

### 5.13 convection\_diffusion\_fraction\_massique\_qc

Description: not\_set

Keyword Discretiser should have already be used to read the object.  
See also: eqn\_base (5.18)

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_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

```

- **espece** *espece* (14)
- **convection** *bloc\_convection* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if *equation\_non\_resolue* keyword is used. Exemple: The Navier Stokes is not solved between time *t0* and *t1*.  
*Navier\_Sokes\_Standard*  
{ *equation\_non\_resolue* (*t>t0*)\*(*t<t1*) }

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

Description: *not\_set*

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

See also: *eqn\_base* (5.18)

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* (21): Turbulence model to be used.
- **espece** *espece* (14)
- **convection** *bloc\_convection* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if *equation\_non\_resolue* keyword is used. Exemple: The Navier Stokes is not solved between time *t0* and *t1*.  

```

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

```

## 5.15 convection\_diffusion\_temperature

Description: Energy equation (temperature diffusion convection).

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

See also: *eqn\_base* (5.18)

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* (5.16): 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* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if *equation\_non\_resolue* keyword is used. Exemple: The Navier Stokes is not solved between time *t0* and *t1*.  

```

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

```

## 5.16 pp

Description: *not\_set*

See also: *listobj* (31.3)

Usage:

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

list of *penalisation\_l2\_ftd\_lec* (5.16.1)

### 5.16.1 penalisation\_l2\_ftd\_lec

Description: not\_set

See also: objet\_lecture (32)

Usage:

**bord** val

where

- **bord** str
- **val**  $n \times 1 \times 2 \dots \times n$

## 5.17 convection\_diffusion\_temperature\_turbulent

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

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

See also: eqn\_base (5.18)

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* (21): Turbulence model for the energy equation.
- **convection** *bloc\_convection* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n\_valeur  
x\_1 y\_1 [z\_1] val\_1  
...  
x\_n y\_n [z\_n] val\_n  
The created files are named : pbyname\_fieldname\_[boundaryname]\_time.dat

- **ecrire\_fichier\_xyz\_valeur\_bin** *ecrire\_fichier\_xyz\_valeur\_param* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.18 eqn\_base

Description: Basic class for equations.

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

See also: mor\_eqn (5) navier\_stokes\_standard (5.23) convection\_diffusion\_temperature (5.15) convection\_diffusion\_temperature\_turbulent (5.17) conduction (5.1) convection\_diffusion\_chaleur\_qc (5.8) transport\_k\_epsilon (5.27) convection\_diffusion\_concentration (5.11) convection\_diffusion\_fraction\_massique\_qc (5.13) convection\_diffusion\_fraction\_massique\_turbulent\_qc (5.14)

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* (5.9): 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* (5.4): Boundary conditions.
- **sources** *sources* (5.5): 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* (5.6): 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* (5.6): 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* (5.7): 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 is not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.19 navier\_stokes\_qc

Description: NAVIER STOKES equations under small Mach number.

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

See also: navier\_stokes\_standard (5.23)

Usage:

```
navier_stokes_qc obj Lire obj {
    [ methode_calcul_pression_initiale str into ['avec_les_cl', 'avec_sources', 'avec_sources_et-
    _operateurs', 'sans_rien']]
    [ projection_initiale int]
    [ solveur_pression solveur_sys_base]
    [ solveur_bar solveur_sys_base]
    [ dt_projection deuxmots]
    [ seuil_divU floatfloat]
    [ traitement_particulier traitement_particulier]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ initial_conditions|conditions_initiales condinits]
    [ boundary_conditions|conditions_limites condlims]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
    [ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
```

```
}
```

where

- **methode\_calcul\_pression\_initiale** *str into ['avec\_les\_cl', 'avec\_sources', 'avec\_sources\_et\_operateurs', 'sans\_rien']* for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : avec\_les\_cl (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), avec\_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier Stokes equation) and avec\_sources\_et\_operateurs (lapP=f is solved as with the previous option avec\_sources but f integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.

- **projection\_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks  $\text{DivU}=0$ . By default, boolean equals 1.
- **solveur\_pression** *solveur\_sys\_base* (9.12) for inheritance: Linear pressure system resolution method.
- **solveur\_bar** *solveur\_sys\_base* (9.12) 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* (5.20) 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* (5.21) 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 (  $\text{lmax}(\text{DivU}) * dt < \text{value}$  )  
 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* (5.22) for inheritance: Keyword to post-process particular values.
- **convection** *bloc\_convection* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if *equation\_non\_resolue* keyword is used. Example: The Navier Stokes is not solved between time  $t_0$  and  $t_1$ .  
 Navier\_Sokes\_Standard  
 { *equation\_non\_resolue* ( $t > t_0$ )\*( $t < t_1$ ) }

## 5.20 deuxmots

Description: Two words.

See also: [objet\\_lecture \(32\)](#)

Usage:

**mot\_1 mot\_2**

where

- **mot\_1** *str*: First word.
- **mot\_2** *str*: Second word.

## 5.21 floatfloat

Description: Two reals.

See also: [objet\\_lecture \(32\)](#)

Usage:

**a b**

where

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

## 5.22 traitement\_particulier

Description: Auxiliary class to post-process particular values.

See also: [objet\\_lecture \(32\)](#)

Usage:

**aco trait\_part acof**

where

- **aco** *str* into [' ']: Open accodance sign.
- **trait\_part** *traitement\_particulier\_base* ([5.22.1](#)): Type of *traitement\_particulier*.
- **acof** *str* into [' ']: Closed accodance sign.

### 5.22.1 traitement\_particulier\_base

Description: Basic class to post-process particular values.

See also: [objet\\_lecture \(32\)](#) [temperature \(5.22.2\)](#) [canal \(5.22.3\)](#) [ec \(5.22.4\)](#) [thi \(5.22.5\)](#) [chmoy\\_faceperio \(5.22.6\)](#)

Usage:

### 5.22.2 temperature

Description: `not_set`

See also: [traitement\\_particulier\\_base \(5.22.1\)](#)

Usage:

```
temperature {  
  
    bord str  
    direction int  
  
}
```

where

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

### 5.22.3 canal

Description: Keyword for statistics on a periodic plane channel.

See also: `traitement_particulier_base` ([5.22.1](#))

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 restart a calculation with previous average quantities.

Note that for thermal and turbulent problems, averages on temperature and turbulent viscosity are automatically calculated. To restart a calculation with phase averaging, `val_moy_temp_XXXXXX.sauv_phase` file is required on the directory where the job is submitted (this last file will be then automatically loaded by TRUST).

### 5.22.4 ec

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

See also: `traitement_particulier_base` ([5.22.1](#))

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`.

### 5.22.5 thi

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

See also: `traitement_particulier_base` ([5.22.1](#))

Usage:

```
thi {  
    init_Ec int  
    [ val_Ec float ]  
    [ facon_init int into [0, 1] ]  
    [ calc_spectre int into [0, 1] ]  
    [ periode_calc_spectre float ]  
    [ 3D int into [0, 1] ]  
    [ 1D int into [0, 1] ]  
    [ conservation_Ec ]  
    [ longueur_boite float ]  
}
```

where

- **init\_Ec** *int*: Keyword to renormalize initial velocity so as kinetic energy equals to the value given by keyword `val_Ec`.
- **val\_Ec** *float*: Keyword to impose a value for kinetic energy by velocity renormalized if `init_Ec` value is 1.
- **facon\_init** *int into [0, 1]*: Keyword to specify how kinetic energy is computed (0 or 1).
- **calc\_spectre** *int into [0, 1]*: Calculate or not the spectrum of kinetic energy.  
Files called `Sorties_THI` are written with inside four columns :  
`time:t global_kinetic_energy:Ec enstrophy:D skewness:S`  
If `calc_spectre` is set to 1, a file `Sorties_THI2_2` is written with three columns :  
`time:t kinetic_energy_at_kc=32 enstrophy_at_kc=32`  
If `calc_spectre` is set to 1, a file `spectre_XXXXX` is written with two columns at each time `XXXXX` :  
`frequency:k energy:E(k)`.
- **periode\_calc\_spectre** *float*: Period for calculating spectrum of kinetic energy
- **3D** *int into [0, 1]*: Calculate or not the 3D spectrum
- **1D** *int into [0, 1]*: Calculate or not the 1D spectrum

- **conservation\_Ec** : If set to 1, velocity field will be changed as to have a constant kinetic energy (default 0)
- **longueur\_boite** *float*: Length of the calculation domain

### 5.22.6 chmoy\_faceperio

Description: non documente

See also: `traitement_particulier_base` (5.22.1)

Usage:

**chmoy\_faceperio** **bloc**

where

- **bloc** *bloc\_lecture* (3.38)

### 5.23 navier\_stokes\_standard

Description: NAVIER STOKES equations.

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

See also: `eqn_base` (5.18) `navier_stokes_turbulent` (5.24) `navier_stokes_qc` (5.19)

Usage:

**navier\_stokes\_standard** **obj** Lire obj {

```
[ methode_calcul_pression_initiale str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_
_operateurs', 'sans_rien']]
[ projection_initiale int]
[ solveur_pression solveur_sys_base]
[ solveur_bar solveur_sys_base]
[ dt_projection deuxmots]
[ seuil_divU floatfloat]
[ traitement_particulier traitement_particulier]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ initial_conditions|conditions_initiales condinits]
[ boundary_conditions|conditions_limites condlims]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
[ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
```

}

where

- **methode\_calcul\_pression\_initiale** *str* into [*'avec\_les\_cl'*, *'avec\_sources'*, *'avec\_sources\_et\_operateurs'*, *'sans\_rien'*]: Keyword to select an option for the pressure calculation before the first time step. Options are : *avec\_les\_cl* (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), *avec\_sources* (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier Stokes equation) and *avec\_sources\_et\_operateurs* (lapP=f is solved as with the previous option *avec\_sources* but f integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.

- **projection\_initiale** *int*: Keyword to suppress, if boolean equals 0, the initial projection which checks  $\text{DivU}=0$ . By default, boolean equals 1.
- **solveur\_pression** *solveur\_sys\_base* (9.12): Linear pressure system resolution method.
- **solveur\_bar** *solveur\_sys\_base* (9.12): 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* (5.20): 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* (5.21): value factor : this keyword is intended to minimise the number of iterations during the pressure system resolution. The convergence criteria during this step ('seuil' in *solveur\_pression*) is dynamically adapted according to the mass conservation. At  $t_n$ , the linear system  $Ax=B$  is considered as solved if the residual  $\|Ax-B\| < \text{seuil}(t_n)$ . For  $t_{n+1}$ , the threshold value  $\text{seuil}(t_{n+1})$  will be evaluated as:  
 If (  $\text{lmax}(\text{DivU}) * dt < \text{value}$  )  
 Seuil( $t_{n+1}$ ) = Seuil( $t_n$ ) \* factor  
 Else  
 Seuil( $t_{n+1}$ ) = Seuil( $t_n$ ) \* factor  
 Endif  
 The first parameter (value) is the mass evolution the user is ready to accept per timestep, and the second one (factor) is the factor of evolution for 'seuil' (for example 1.1, so 10)
- **traitement\_particulier** *traitement\_particulier* (5.22): Keyword to post-process particular values.
- **convection** *bloc\_convection* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if *equation\_non\_resolue* keyword is used. Example: The Navier Stokes is not solved between time  $t_0$  and  $t_1$ .  
 Navier\_Sokes\_Standard  
 { equation\_non\_resolue ( $t > t_0$ )\*( $t < t_1$ ) }

## 5.24 navier\_stokes\_turbulent

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

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

See also: `navier_stokes_standard` (5.23) `navier_stokes_turbulent_qc` (5.26)

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  $\text{lapP}=0$  is solved with Neuman boundary conditions on pressure if any), *avec\_sources* ( $\text{lapP}=f$  is solved with Neuman boundaries conditions and  $f$  integrating the source terms of the Navier Stokes equation) and *avec\_sources\_et\_operateurs* ( $\text{lapP}=f$  is solved as with the previous option *avec\_sources* but  $f$  integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.
- **projection\_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks  $\text{DivU}=0$ . By default, boolean equals 1.
- **solveur\_pression** *solveur\_sys\_base* (9.12) for inheritance: Linear pressure system resolution method.
- **solveur\_bar** *solveur\_sys\_base* (9.12) 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* (5.20) 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* (5.21) 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\_precession) 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)*dt<value )
Seuil(tn+1)= Seuil(tn)*factor
Else
Seuil(tn+1)= Seuil(tn)*factor
Endif
```

The first parameter (value) is the mass evolution the user is ready to accept per timestep, and the second one (factor) is the factor of evolution for 'seuil' (for example 1.1, so 10

- **traitement\_particulier** *traitement\_particulier* (5.22) for inheritance: Keyword to post-process particular values.
- **convection** *bloc\_convection* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.  
Navier\_Sokes\_Standard  
{ equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.25 modele\_turbulence\_hyd\_deriv

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

See also: objet\_lecture (32) NUL (5.25.2) mod\_turb\_hyd\_ss\_maille (5.25.3) k\_epsilon (5.25.10)

Usage:

```
modele_turbulence_hyd_deriv {
    [ 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]
[ eps_min float]
[ k_min float]
[ prandtl_k float]
[ prandtl_eps float]
}
where

```

- **correction\_visco\_turb\_pour\_controle\_pas\_de\_temps** : Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction\_visco\_turb\_pour\_controle\_pas\_de\_temps\_parametre** *float*: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence\_paro** *turbulence\_paro\_base* (29): 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).
- **eps\_min** *float*: Lower limitation of epsilon (default value 1.e-10).
- **k\_min** *float*: Lower limitation of k (default value 1.e-10).
- **prandtl\_k** *float*: Keyword to change the Prk value (default 1.0).
- **prandtl\_eps** *float*: Keyword to change the Pre value (default 1.3).

### 5.25.1 dt\_impr\_ustar\_mean\_only

Description: `not_set`

See also: `objet_lecture` (32)

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_paro ] [ dt_impr_ustar ] [ dt_impr_ustar_mean_only ] [  
nut_max ] [ eps_min ] [ k_min ] [ prandtl_k ] [ prandtl_eps ]
```

where

- **correction\_visco\_turb\_pour\_controle\_pas\_de\_temps** : Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr\_visco\_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction\_visco\_turb\_pour\_controle\_pas\_de\_temps\_parametre** *float*: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence\_paro** *turbulence\_paro\_base* (29): 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).
- **eps\_min** *float*: Lower limitation of epsilon (default value 1.e-10).
- **k\_min** *float*: Lower limitation of k (default value 1.e-10).
- **prandtl\_k** *float*: Keyword to change the Prk value (default 1.0).
- **prandtl\_eps** *float*: Keyword to change the Pre value (default 1.3).

### 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)

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_paro turbulence_paro_base]  
    [ dt_impr_ustar float]
```

```

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

```

- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (5.25.4): The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur\_maille** *str into ['volume', 'volume\_sans\_lissage', 'scotti', 'arrete']*: different ways to calculate the characteristic length may be specified :  
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.  
 volume\_sans\_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).  
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.  
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction\_visco\_turb\_pour\_controle\_pas\_de\_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr\_visco\_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction\_visco\_turb\_pour\_controle\_pas\_de\_temps\_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence\_paro** *turbulence\_paro\_base* (29) 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).
- **eps\_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k\_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl\_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl\_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

#### 5.25.4 form\_a\_nb\_points

Description: The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.

See also: `objet_lecture` ([32](#))

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  $\mathcal{O}(y^3)$  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_paroit turbulence_paroit_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
[ eps_min float]
[ k_min float]
[ prandtl_k float]
[ prandtl_eps float]
```

}

where

- **cw** *float*: The unique parameter (constant) of the WALE-model (by default value 0.5).
- **formulation\_a\_nb\_points** *form\_a\_nb\_points* ([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* (29) 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).
- **eps\_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k\_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl\_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl\_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

#### 5.25.6 sous\_maille\_smago

Description: Smagorinsky sub-grid turbulence model.

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

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

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]
    [ eps_min float]
    [ k_min float]
    [ prandtl_k float]
    [ prandtl_eps float]
}
```

where

- **cs** *float*: This is an optional keyword and the value is used to set the constant used in the Smagorinsky model (This is currently only valid for Smagorinsky models and it is set to 0.18 by default) .
- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (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* (29) 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).
- **eps\_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k\_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl\_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl\_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

### 5.25.7 combinaison

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

See also: mod\_turb\_hyd\_ss\_maille (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_paroit turbulence_paroit_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
[ eps_min float]
[ k_min float]
[ prandtl_k float]
[ prandtl_eps float]
}

```

where

- **nb\_var** *n word1 word2 ... wordn*: Number and names of variables which will be used in the turbulent viscosity definition (by default 0)
- **fonction** *str*: Fonction for turbulent viscosity. X,Y,Z and variables defined previously can be used.
- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (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* (29) 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).
- **eps\_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k\_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).



- **prandtl\_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl\_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

### 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_paro turbulence_paro_base]
    [ dt_impr_ustar float]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
    [ nut_max float]
    [ eps_min float]
    [ k_min float]
    [ prandtl_k float]
    [ prandtl_eps float]
}
```

where

- **canalx** *float*: [height] : plane channel according to Ox direction (for the moment, formulation in the code relies on fixed height : H=2).
- **tuyauz** *float*: [diameter] : pipe according to Oz direction (for the moment, formulation in the code relies on fixed diameter : D=2).
- **verif\_dparoi** *str*
- **dmax** *float*: Maximum distance.
- **fichier** *str*
- **fichier\_ecriture\_K\_Eps** *str*: When a restart with k-epsilon model is envisaged, this keyword allows to generate external MED-format file with evaluation of k and epsilon quantities (based on eddy turbulent viscosity and turbulent characteristic length returned by mixing length model). The frequency of the MED file print is set equal to dt\_impr\_ustar. Moreover, k-eps MED field is automatically saved at the last time step. MED file is then used for the restarting K-Epsilon calculation with the Champ\_Fonc\_Med keyword.
- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (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* (29) 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).
- **eps\_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k\_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl\_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl\_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

### 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_paro turbulence_paro_base]
    [ dt_impr_ustar float]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
    [ nut_max float]
```

```

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

```

- **formulation\_a\_nb\_points** *form\_a\_nb\_points* (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* (29) 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).
- **eps\_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k\_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl\_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl\_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

### 5.25.10 k\_epsilon

Description: Turbulence model (k-eps).

See also: *modele\_turbulence\_hyd\_deriv* (5.25)

Usage:

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

where

- **cmu** *float*: Keyword to modify the Cmu constant of k-eps model :  $Nut = Cmu * k^2 / \epsilon$  Default value is 0.09
- **transport\_k\_epsilon** *transport\_k\_epsilon* (5.27): Keyword to define the (k-eps) transportation equation.
- **modele\_fonc\_bas\_reynolds** *modele\_fonction\_bas\_reynolds\_base* (5.25.11): This keyword is used to set the bas Reynolds model used.
- **correction\_visco\_turb\_pour\_controle\_pas\_de\_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction\_visco\_turb\_pour\_controle\_pas\_de\_temps\_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence\_paro** *turbulence\_paro\_base* (29) 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).
- **eps\_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k\_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl\_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl\_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

### 5.25.11 modele\_fonction\_bas\_reynolds\_base

Description: not\_set

See also: objet\_lecture (32)

Usage:

### 5.26 navier\_stokes\_turbulent\_qc

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

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

See also: navier\_stokes\_turbulent (5.24)

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  $\text{lapP}=0$  is solved with Neuman boundary conditions on pressure if any), *avec\_sources* ( $\text{lapP}=f$  is solved with Neuman boundaries conditions and  $f$  integrating the source terms of the Navier Stokes equation) and *avec\_sources\_et\_operateurs* ( $\text{lapP}=f$  is solved as with the previous option *avec\_sources* but  $f$  integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicit when using an implicit time scheme to solve the Navier Stokes equation.
- **projection\_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks  $\text{DivU}=0$ . By default, boolean equals 1.
- **solveur\_pression** *solveur\_sys\_base* (9.12) for inheritance: Linear pressure system resolution method.

- **solveur\_bar** *solveur\_sys\_base* (9.12) 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* (5.20) 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* (5.21) 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 (  $\text{lmax}(\text{DivU}) * dt < \text{value}$  )  
 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* (5.22) for inheritance: Keyword to post-process particular values.
- **convection** *bloc\_convection* (5.9) 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* (5.4) for inheritance: Boundary conditions.
- **sources** *sources* (5.5) 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* (5.6) 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* (5.6) 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* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Example: The Navier Stokes is not solved between time t0 and t1.  
 Navier\_Sokes\_Standard  
 { equation\_non\_resolue (t>t0)\*(t<t1) }

## 5.27 transport\_k\_epsilon

Description: The (k-eps) transportation equation. To restart from a previous mixing length calculation, an external MED-format file containing reconstructed K and Epsilon quantities can be read (see fichier\_écriture\_k\_eps) thanks to the Champ\_fonc\_MED keyword.

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

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

See also: eqn\_base (5.18)

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_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
```

- **with\_nu** str into ['yes', 'no']: yes/no
- **convection** bloc\_convection (5.9) 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** (5.4) for inheritance: Boundary conditions.
- **sources** sources (5.5) 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** (5.6) 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** (5.6) 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** (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation\_non\_resolue** str for inheritance: The equation will not be solved while condition(t) is verified if equation\_non\_resolue keyword is used. Exemple: The Navier Stokes is not solved between

```

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

```

## 6 /\*

### 6.1 /\*

Description: bloc of Comment in a data file.

See also: [objet\\_u \(33\)](#)

Usage:

```
/* comm
```

where

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

## 7 champ\_generique\_base

Description: not\_set

See also: [objet\\_u \(33\)](#) [champ\\_post\\_de\\_champs\\_post \(7.1\)](#) [predefini \(7.15\)](#) [champ\\_post\\_refchamp \(7.17\)](#)

Usage:

### 7.1 champ\_post\_de\_champs\_post

Description: not\_set

See also: [champ\\_generique\\_base \(7\)](#) [champ\\_post\\_operateur\\_eqn \(7.5\)](#) [champ\\_post\\_transformation \(7.19\)](#) [champ\\_post\\_reduction\\_0d \(7.16\)](#) [champ\\_post\\_operateur\\_base \(7.4\)](#) [champ\\_post\\_statistiques\\_base \(7.6\)](#) [champ\\_post\\_extraction \(7.10\)](#) [champ\\_post\\_morceau\\_equation \(7.13\)](#) [champ\\_post\\_tparoi\\_vef \(7.18\)](#) [champ\\_post\\_interpolation \(7.12\)](#)

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 (7)*: the source field.
- **nom\_source** *str*: To name a source field with the nom\_source keyword
- **source\_reference** *str*
- **sources\_reference** *list\_nom\_virgule (7.2)*
- **sources** *listchamp\_generique (7.3)*: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }



## 7.2 list\_nom\_virgule

Description: List of name.

See also: listobj (31.3)

Usage:

{ object1 , object2 .... }  
list of *nom\_anonyme* (22.1) separated with ,

## 7.3 listchamp\_generique

Description: XXX

See also: listobj (31.3)

Usage:

{ object1 , object2 .... }  
list of *champ\_generique\_base* (7) separated with ,

## 7.4 champ\_post\_operateur\_base

Description: not\_set

See also: champ\_post\_de\_champs\_post (7.1) champ\_post\_operateur\_gradient (7.11) champ\_post\_operateur-divergence (7.8)

Usage:

**champ\_post\_operateur\_base** 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* (7) for inheritance: the source field.
- **nom\_source** *str* for inheritance: To name a source field with the *nom\_source* keyword
- **source\_reference** *str* for inheritance
- **sources\_reference** *list\_nom\_virgule* (7.2) for inheritance
- **sources** *listchamp\_generique* (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 7.5 champ\_post\_operateur\_eqn

Synonymous: **operateur\_eqn**

Description: not\_set

See also: champ\_post\_de\_champs\_post (7.1)

Usage:

```

champ_post_operateur_eqn 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* (7) for inheritance: the source field.
- **nom\_source** *str* for inheritance: To name a source field with the **nom\_source** keyword
- **source\_reference** *str* for inheritance
- **sources\_reference** *list\_nom\_virgule* (7.2) for inheritance
- **sources** *listchamp\_generique* (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 7.6 champ\_post\_statistiques\_base

Description: *not\_set*

See also: *champ\_post\_de\_champs\_post* (7.1) *correlation* (7.7) *moyenne* (7.14) *ecart\_type* (7.9)

Usage:

```

champ_post_statistiques_base 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* (7) for inheritance: the source field.
- **nom\_source** *str* for inheritance: To name a source field with the **nom\_source** keyword
- **source\_reference** *str* for inheritance
- **sources\_reference** *list\_nom\_virgule* (7.2) for inheritance
- **sources** *listchamp\_generique* (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 7.7 correlation

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

Description: to calculate the correlation between the two fields.

See also: `champ_post_statistiques_base` (7.6)

Usage:

```
correlation 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 (7) for inheritance: the source field.
- **nom\_source** str for inheritance: To name a source field with the nom\_source keyword
- **source\_reference** str for inheritance
- **sources\_reference** list\_nom\_virgule (7.2) for inheritance
- **sources** listchamp\_generique (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post..  
{ ... }}

## 7.8 champ\_post\_operateur\_divergence

Synonymous: **divergence**

Description: To calculate divergency of a given field.

See also: `champ_post_operateur_base` (7.4)

Usage:

```
champ_post_operateur_divergence 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 (7) for inheritance: the source field.
- **nom\_source** str for inheritance: To name a source field with the nom\_source keyword
- **source\_reference** str for inheritance
- **sources\_reference** list\_nom\_virgule (7.2) for inheritance
- **sources** listchamp\_generique (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post..  
{ ... }}

## 7.9 ecart\_type

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

Description: to calculate the standard deviation (statistic rms) of the field nom\_champ.

See also: champ\_post\_statistiques\_base (7.6)

Usage:

```
ecart_type 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 (7) for inheritance: the source field.
- **nom\_source** str for inheritance: To name a source field with the nom\_source keyword
- **source\_reference** str for inheritance
- **sources\_reference** list\_nom\_virgule (7.2) for inheritance
- **sources** listchamp\_generique (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post..  
{ ... }}

## 7.10 champ\_post\_extraction

Synonymous: **extraction**

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

See also: champ\_post\_de\_champs\_post (7.1)

Usage:

```
champ_post_extraction 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* (7) for inheritance: the source field.
- **nom\_source** *str* for inheritance: To name a source field with the nom\_source keyword
- **source\_reference** *str* for inheritance
- **sources\_reference** *list\_nom\_virgule* (7.2) for inheritance
- **sources** *listchamp\_generique* (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 7.11 champ\_post\_operateur\_gradient

Synonymous: **gradient**

Description: To calculate gradient of a given field.

See also: champ\_post\_operateur\_base (7.4)

Usage:

**champ\_post\_operateur\_gradient** 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* (7) for inheritance: the source field.
- **nom\_source** *str* for inheritance: To name a source field with the nom\_source keyword
- **source\_reference** *str* for inheritance
- **sources\_reference** *list\_nom\_virgule* (7.2) for inheritance
- **sources** *listchamp\_generique* (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 7.12 champ\_post\_interpolation

Synonymous: **interpolation**

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

See also: champ\_post\_de\_champs\_post (7.1)

Usage:

**champ\_post\_interpolation** 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* (7) for inheritance: the source field.
- **nom\_source** *str* for inheritance: To name a source field with the nom\_source keyword
- **source\_reference** *str* for inheritance
- **sources\_reference** *list\_nom\_virgule* (7.2) for inheritance
- **sources** *listchamp\_generique* (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

### 7.13 champ\_post\_morceau\_equation

Synonymous: **morceau\_equation**

Description: To calculate a field related to a piece of equation. For the moment, the field which can be calculated is the stability time step of an operator equation. The problem name and the unknown of the equation should be given by Source refChamp { Pb\_Champ problem\_name unknown\_field\_of\_equation }

See also: champ\_post\_de\_champs\_post (7.1)

Usage:

**champ\_post\_morceau\_equation** 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* (7) for inheritance: the source field.
- **nom\_source** *str* for inheritance: To name a source field with the nom\_source keyword
- **source\_reference** *str* for inheritance
- **sources\_reference** *list\_nom\_virgule* (7.2) for inheritance

- **sources** *listchamp\_generique* (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 7.14 moyenne

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

Description: to calculate the average of the field over time

See also: **champ\_post\_statistiques\_base** (7.6)

Usage:

```
moyenne 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* (15.1): This option allows to read a converged time averaged field in a .xyz file in order to calculate, when restarting the calculation, the statistics fields (rms, correlation) which depend on this average. In that case, the time averaged field is not updated during the restarting calculation. In this case, the time averaged field must be fully converged to avoid errors when calculating high order statistics.
- **t\_deb** *float* for inheritance: Start of integration time
- **t\_fin** *float* for inheritance: End of integration time
- **source** *champ\_generique\_base* (7) for inheritance: the source field.
- **nom\_source** *str* for inheritance: To name a source field with the **nom\_source** keyword
- **source\_reference** *str* for inheritance
- **sources\_reference** *list\_nom\_virgule* (7.2) for inheritance
- **sources** *listchamp\_generique* (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 7.15 predefini

Description: These keyword is used to post process predefined postprocessing fields. For the moment, only kinetic energy (**energie\_cinetique** keyword to use for **field\_name**) is available.

See also: **champ\_generique\_base** (7)

Usage:

```
predefini obj Lire obj {
    pb_champ deuxmots
}
```

where

- **pb\_champ** *deuxmots* (5.20): { Pb\_champ nom\_pb nom\_champ } : nom\_pb is the problem name and nom\_champ is the selected field name.

## 7.16 champ\_post\_reduction\_0d

Synonymous: **reduction\_0d**

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

See also: champ\_post\_de\_champs\_post (7.1)

Usage:

```
champ_post_reduction_0d obj Lire obj {
    methode str into ['min', 'max', 'moyenne', 'somme', 'moyenne_ponderee', 'somme_ponderee',
    'norme_l2', 'normalized_norm_l2']
    [ source champ_generique_base]
    [ nom_source str]
    [ source_reference str]
    [ sources_reference list_nom_virgule]
    [ sources listchamp_generique]
```

}

where

- **methode** str into ['min', 'max', 'moyenne', 'somme', 'moyenne\_ponderee', 'somme\_ponderee', 'norme\_l2', 'normalized\_norm\_l2']: name of the reduction method (min, max, somme for the sum, somme\_ponderee for a weighted sum (integral), norme\_L2 for the L2 norm, normalized\_norm\_L2 for the L2 norm normalized, moyenne for a mean and moyenne\_ponderee for a mean ponderated by integration volumes, e.g: cell volumes for temperature or pressure in VDF, volumes around faces for velocity and temperature in VEF)
- **source** champ\_generique\_base (7) for inheritance: the source field.
- **nom\_source** str for inheritance: To name a source field with the nom\_source keyword
- **source\_reference** str for inheritance
- **sources\_reference** list\_nom\_virgule (7.2) for inheritance
- **sources** listchamp\_generique (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 7.17 champ\_post\_refchamp

Synonymous: **refchamp**

Description: Field of prolem

See also: champ\_generique\_base (7)

Usage:

```
champ_post_refchamp obj Lire obj {
    pb_champ deuxmots
    [ nom_source str]
```

}

where



- **pb\_champ** *deuxmots* (5.20): { Pb\_champ nom\_pb nom\_champ } : nom\_pb is the problem name and nom\_champ is the selected field name.
- **nom\_source** *str*: The alias name for the field

## 7.18 champ\_post\_tparoi\_vef

Synonymous: **tparoi\_vef**

Description: These keyword is used to post process (only for VEF discretization) the temperature field with a slight difference on boundaries with Neumann condition where law of the wall is applied on the temperature field. nom\_pb is the problem name and field\_name is the selected field name. A keyword (temperature\_physique) is available to post process this field without using Definition\_champs.

See also: champ\_post\_de\_champs\_post (7.1)

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* (7) for inheritance: the source field.
- **nom\_source** *str* for inheritance: To name a source field with the nom\_source keyword
- **source\_reference** *str* for inheritance
- **sources\_reference** *list\_nom\_virgule* (7.2) for inheritance
- **sources** *listchamp\_generique* (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 7.19 champ\_post\_transformation

Synonymous: **transformation**

Description: To create a field with a transformation.

See also: champ\_post\_de\_champs\_post (7.1)

Usage:

**champ\_post\_transformation** 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* (7) for inheritance: the source field.
- **nom\_source** *str* for inheritance: To name a source field with the nom\_source keyword
- **source\_reference** *str* for inheritance
- **sources\_reference** *list\_nom\_virgule* (7.2) for inheritance
- **sources** *listchamp\_generique* (7.3) for inheritance: sources { Champ\_Post.... { ... } Champ\_Post.. { ... } }

## 8 chimie

Description: Keyword to describe the chmical reactions

See also: objet\_u (33)

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* (8.1): list of reactions
- **modele\_micro\_melange** *int*: modele\_micro\_melange (0 by default)
- **constante\_modele\_micro\_melange** *float*: constante of modele (1 by default)
- **espece\_en\_competition\_micro\_melange** *str*: espece in competition in reactions

### 8.1 reactions

Description: list of reactions

See also: listobj (31.3)

Usage:

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

list of *reaction* (8.1.1) separated with ,

### 8.1.1 reaction

Description: Keyword to describe reaction:

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

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

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

See also: `objet_lecture` ([32](#))

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* ([3.38](#)): coefficients of activity (exemple { CH<sub>4</sub> 1 O<sub>2</sub> 2 })
- **enthalpie\_reaction** *float*: DH
- **energie\_activation** *float*: E<sub>a</sub>
- **exposant\_beta** *float*: Beta
- **contre\_reaction** *float*: K<sub>inv</sub>
- **contre\_energie\_activation** *float*: c<sub>r</sub> E<sub>a</sub>

## 9 class\_generic

Description: `not_set`

See also: `objet_u` ([33](#)) `dt_start` ([9.5](#)) `solveur_sys_base` ([9.12](#))

Usage:

### 9.1 cholesky

Description: Cholesky direct method.

See also: `solveur_sys_base` ([9.12](#))

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

## 9.2 dt\_calc

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

See also: [dt\\_start \(9.5\)](#)

Usage:

**dt\_calc**

## 9.3 dt\_fixe

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

See also: [dt\\_start \(9.5\)](#)

Usage:

**dt\_fixe value**

where

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

## 9.4 dt\_min

Description: The first iteration is based on dt\_min.

See also: [dt\\_start \(9.5\)](#)

Usage:

**dt\_min**

## 9.5 dt\_start

Description: not\_set

See also: [class\\_generic \(9\)](#) [dt\\_calc \(9.2\)](#) [dt\\_min \(9.4\)](#) [dt\\_fixe \(9.3\)](#)

Usage:

**dt\_start**

## 9.6 gcp\_ns

Description: not\_set

See also: [gcp \(9.11\)](#)

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* (9.12): Solver type.
- **solveur1** *solveur\_sys\_base* (9.12): Solver type.
- **precond** *precond\_base* (24) for inheritance: Keyword to define system preconditioning in order to accelerate resolution by the conjugated gradient. Many parallel preconditioning methods are not equivalent to their sequential counterpart, and you should therefore expect differences, especially when you select a high value of the final residue (**seuil**). The result depends on the number of processors and on the mesh splitting. It is sometimes useful to run the solver with no preconditioning at all. In particular:
  - when the solver does not converge during initial projection,
  - when comparing sequential and parallel computations.
 With no preconditioning, except in some particular cases (no open boundary), the sequential and the parallel computations should provide exactly the same results within fpv accuracy. If not, there might be a coding error or the system of equations is singular.
- **precond\_nul** for inheritance: Keyword to not use a preconditioning method.
- **seuil** *float* for inheritance: Value of the final residue. The gradient ceases iteration when the Euclidean residue standard  $\|Ax-B\|$  is less than this value.
- **impr** for inheritance: Keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **quiet** for inheritance: To not displaying any outputs of the solver.
- **save\_matrix|save\_matrice** for inheritance: to save the matrix in a file.
- **optimized** for inheritance: This keyword triggers a memory and network optimized algorithms useful for strong scaling (when computing less than 100 000 elements per processor). The matrix and the vectors are duplicated, common items removed and only virtual items really used in the matrix are exchanged.  
Warning: this is experimental and known to fail in some VEF computations (L2 projection step will not converge). Works well in VDF.
- **nb\_it\_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gcp.

## 9.7 gen

Description: not\_set

See also: *solveur\_sys\_base* (9.12)

Usage:

**gen data**

where

- **data** *bloc\_lecture* (3.38)

## 9.8 gmres

Description: Gmres method (for non symmetric matrix).

See also: `solveur_sys_base` ([9.12](#))

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*

## 9.9 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` ([9.12](#))

Usage:

```
optimal obj Lire obj {  
    seuil float  
    [ impr ]  
    [ quiet ]  
    [ save_matrix|save_matrice ]  
    [ frequence_recalc int ]  
    [ nom_fichier_solveur str ]  
    [ fichier_solveur_non_recre ]  
}
```

where

- **seuil** *float*: Convergence threshold
- **impr** : To print the convergency of the fastest solver
- **quiet** : To disable printing of information

- **save\_matrix|save\_matrice** : To save the linear system (A, x, B) into a file
- **frequence\_recalc** *int*: To set a time step period (by default, 100) for re-checking the fastest solver
- **nom\_fichier\_solveur** *str*: To specify the file containing the list of the tested solvers
- **fichier\_solveur\_non\_recre** : To avoid the creation of the file containing the list

## 9.10 petsc

Description: Solveur via Petsc API

Usage:

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

*Solver* : Several solvers through PETSc API are available :

**GCP** : Conjugate Gradient

**PIPECG** : Pipelined Conjugate Gradient (possible reduced CPU cost during massive parallel calculation due to a single non-blocking reduction per iteration, if TRUST is built with a MPI-3 implementation).

**GMRES** : Generalized Minimal Residual

**BICGSTAB** : Stabilized Bi-Conjugate Gradient

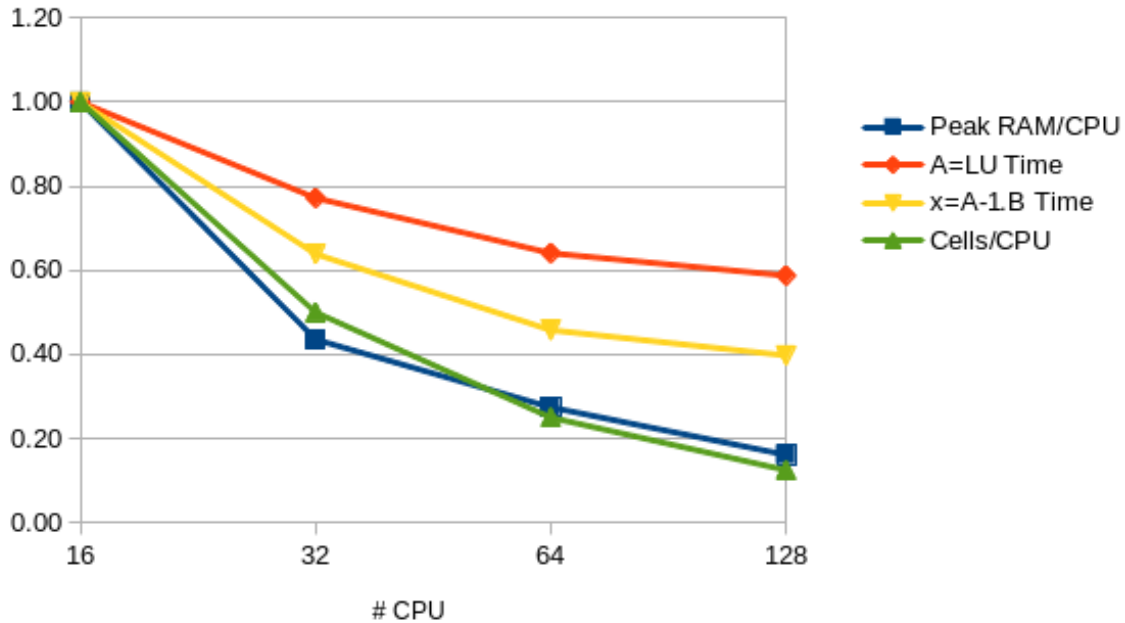
**IBICGSTAB** : Improved version of previous one for massive parallel computations (only a single global reduction operation instead of the usual 3 or 4).

**CHOLESKY** : Parallelized version of Cholesky from MUMPS library. This solver accepts since the 1.6.7 version an option to select a different ordering than the automatic selected one by MUMPS (and printed by using the **impr** option). The possible choices are **Metis** | **Scotch** | **PT-Scotch** | **Parmetis**. The two last options can't only be used during a parallel calculation, whereas the two first are available for sequential or parallel calculations. It seems that the CPU cost of A=LU factorization but also of the backward/forward elimination steps may sometimes be reduced by selecting a different ordering than the default one. Notice that this solver requires a huge amount of memory compared to iterative methods. To know how many RAM you will need by core, then use the **impr** option to have detailed informations during the analysis phase and before the factorisation phase (in the following output, you will learn that the largest memory is taken by the 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 disc (save RAM memory but add an extra CPU cost during  $Ax=B$  solve)

**CHOLESKY\_SUPERLU** : Parallelized Cholesky from SUPERLU\_DIST library (less CPU and RAM efficient than the previous one)

**CHOLESKY\_PASTIX** : Parallelized Cholesky from PASTIX library

**CHOLESKY\_UMFPACK** : Sequential Cholesky from UMFPACK library (seems fast).

**CLI** { string } : Command Line Interface. Should be used only by advanced users, to access the whole solver/preconditioners from the PETSC API. To find all the available options, run your calculation with the -ksp\_view -help options:

trust datafile [N] -ksp\_view -help

...

#### Preconditioner (PC) Options -----

-pc\_type Preconditioner: (one of) none jacobi pbjacobi bjacobi sor lu shell mg  
eisenstat ilu icc cholesky asm ksp composite redundant nn mat fieldsplit galerkin openmp spai hypre  
tfs (PCSetType)

HYPRE preconditioner options

-pc\_hypre\_type <pilut> (choose one of) pilut parasails boomeramg

HYPRE ParaSails Options

-pc\_hypre\_parasails\_nlevels <1>: Number of number of levels (None)

-pc\_hypre\_parasails\_thresh <0.1>: Threshold (None)

-pc\_hypre\_parasails\_filter <0.1>: filter (None)

-pc\_hypre\_parasails\_loadbal <0>: Load balance (None)

-pc\_hypre\_parasails\_logging: <FALSE> Print info to screen (None)



-pc\_hypre\_parasails\_reuse: <FALSE> Reuse nonzero pattern in preconditioner (None)  
 -pc\_hypre\_parasails\_sym <nonsymmetric> (choose one of) nonsymmetric SPD nonsymmetric, SPD

#### Krylov Method (KSP) Options -----

-ksp\_type Krylov method:(one of) cg cgne stcg gltr richardson chebychev gmres tcqmr  
 bcgs bcgsl cgs tfqmr cr lsqr preonly qcg bicg fgmres minres symmlq lgmres lcd (KSPSetType)  
 -ksp\_max\_it <10000>: Maximum number of iterations (KSPSetTolerances)  
 -ksp\_rtol <0>: Relative decrease in residual norm (KSPSetTolerances)  
 -ksp\_atol <1e-12>: Absolute value of residual norm (KSPSetTolerances)  
 -ksp\_divtol <10000>: Residual norm increase cause divergence (KSPSetTolerances)  
 -ksp\_converged\_use\_initial\_residual\_norm: Use initial residual residual norm for computing relative convergence  
 -ksp\_monitor\_singular\_value <stdout>: Monitor singular values (KSPMonitorSet)  
 -ksp\_monitor\_short <stdout>: Monitor preconditioned residual norm with fewer digits (KSPMonitorSet)  
 -ksp\_monitor\_draw: Monitor graphically preconditioned residual norm (KSPMonitorSet)  
 -ksp\_monitor\_draw\_true\_residual: Monitor graphically true residual norm (KSPMonitorSet)

Example to use the multigrid method as a solver, not only as a preconditioner:

**Solveur\_pression Petsc CLI** { -ksp\_type richardson -pc\_type hypre -pc\_hypre\_type boomeramg -ksp\_atol 1.e-7 }

*Precond* : Several preconditioners are available :

**NULL** { } : No preconditioner used

**BLOCK\_JACOBI\_ICC** { **level** k **ordering** *natural* | **rcm** } : Incomplete Cholesky factorization for symmetric matrix with the PETSc implementation. The integer k is the factorization level (default value, 1). In parallel, the factorization is done by block (one per processor by default). The ordering of the local matrix is **natural** by default, but **rcm** ordering, which reduces the bandwidth of the local matrix, may interestingly improve the quality of the decomposition and reduces the number of iterations.

**SSOR** { **omega** double } : Symmetric Successive Over Relaxation algorithm. **omega** (default value, 1.5) defines the relaxation factor.

**EISENTAT** { **omega** double } : SSOR version with Eisenstat trick which reduces the number of computations and thus CPU cost

**SPAI** { **level** nlevels **epsilon** thresh } : Spai Approximate Inverse algorithm from Parasails Hypre library. Two parameters are available, nlevels and thresh.

**PILUT** { **level** k **epsilon** thresh } : Dual Threshold Incomplete LU factorization. The integer k is the factorization level and **epsilon** is the drop tolerance.

**DIAG** { } : Diagonal (Jacobi) preconditioner.

**BOOMERAMG** { } : Multigrid preconditioner (no option is available yet, look at CLI command and Petsc documentation to try other options).

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

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

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

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

**save\_matrix/read\_matrix** are the keywords to save/read into a file the constant matrix A of the linear system  $Ax=B$  solved (eg: matrix from the pressure linear system for an incompressible flow). It is useful

when you want to minimize the MPI communications on massive parallel calculation. Indeed, in VEF discretization, the overlapping width (generally 2, specified with the **largeur\_joint** option in the partition keyword **partition**) can be reduced to 1, once the matrix has been properly assembled and saved. The cost of the MPI communications in TRUST itself (not in PETSc) will be reduced with length messages divided by 2. So the strategy is:

- I) Partition your VEF mesh with a **largeur\_joint** value of 2
- II) Run your parallel calculation on 0 time step, to build and save the matrix with the **save\_matrix** option. A file named *Matrix\_NBROWS\_rows\_NCPUS\_cpus.petsc* will be saved to the disc (where NBROWS is the number of rows of the matrix and NCPUS the number of CPUs used).
- III) Partition your VEF mesh with a **largeur\_joint** value of 1
- IV) Run your parallel calculation completely now and substitute the **save\_matrix** option by the **read\_matrix** option. Some interesting gains have been noticed when the cost of linear system solve with PETSc is small compared to all the other operations.

#### **TIPS:**

A) Solver for symmetric linear systems (e.g: Pressure system from Navier Stokes equation):

-The **CHOLSKY** parallel solver is from MUMPS library. It offers better performance than all others solvers if you have enough RAM for your calculation. A parallel calculation on a cluster with 4GBytes on each processor, 40000 cells/processor seems the upper limit. Seems to be very slow to initialize above 500 cpus/cores.

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

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

B) Solver for non symmetric linear systems (e.g.: Implicit schemes):

The **BICGSTAB/DIAG** solver seems to offer the best performances.

Additional information is available into the PETSC documentation available there: `$TRUST_ROOT/lib/src/LIBPETSC/petsc/*/do`

See also: `solveur_sys_base` (9.12)

Usage:

**petsc solveur option\_solveur**

where

- **solveur** *str*
- **option\_solveur** *bloc\_lecture* (3.38)

## **9.11 gcp**

Description: Preconditioned conjugated gradient.

See also: `solveur_sys_base` (9.12) `gcp_ns` (9.6)

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

## 9.12 solveur\_sys\_base

Description: Basic class to solve the linear system.

See also: [class\\_generic \(9\)](#) [optimal \(9.9\)](#) [gen \(9.7\)](#) [petsc \(9.10\)](#) [gcp \(9.11\)](#) [cholesky \(9.1\)](#) [gmres \(9.8\)](#)

Usage:

## 10 #

### 10.1 #

Description: Comments in a data file.

See also: [objet\\_u \(33\)](#)

Usage:

# **comm**

where

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

## 11 **condlim\_base**

Description: Basic class of boundary conditions.

See also: [objet\\_u \(33\)](#) [paroi\\_fixe \(11.33\)](#) [symetrie \(11.42\)](#) [periodique \(11.39\)](#) [paroi\\_decalee\\_robin \(11.25\)](#) [paroi\\_adiabatique \(11.21\)](#) [dirichlet \(11.2\)](#) [neumann \(11.20\)](#) [paroi\\_couple \(11.24\)](#) [paroi\\_contact \(11.22\)](#) [paroi\\_contact\\_fictif \(11.23\)](#) [paroi\\_echange\\_contact\\_vdf \(11.29\)](#) [paroi\\_echange\\_externe\\_impose \(11.30\)](#) [paroi\\_echange\\_global\\_impose \(11.32\)](#) [Paroi \(11.1\)](#) [frontiere\\_ouverte\\_k\\_eps\\_impose \(11.12\)](#) [paroi\\_flux-\\_impose \(11.35\)](#) [frontiere\\_ouverte\\_fraction\\_massique\\_imposee \(11.6\)](#) [paroi\\_echange\\_contact\\_correlation-\\_vdf \(11.27\)](#) [paroi\\_echange\\_contact\\_correlation\\_vdf \(11.28\)](#)

Usage:

**condlim\_base**

### 11.1 **Paroi**

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

See also: [condlim\\_base \(11\)](#)

Usage:

**Paroi**

### 11.2 **dirichlet**

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

See also: [condlim\\_base \(11\)](#) [paroi\\_defilante \(11.26\)](#) [paroi\\_knudsen\\_non\\_negligeable \(11.36\)](#) [paroi\\_rugueuse \(11.37\)](#) [frontiere\\_ouverte\\_vitesse\\_imposee \(11.18\)](#) [frontiere\\_ouverte\\_temperature\\_imposee \(11.17\)](#) [frontiere\\_ouverte\\_concentration\\_imposee \(11.5\)](#) [paroi\\_temperature\\_imposee \(11.38\)](#) [scalaire\\_impose\\_paro \(11.40\)](#)

Usage:

**dirichlet**

### 11.3 **entree\_temperature\_imposee\_h**

Description: Particular case of class [frontiere\\_ouverte\\_temperature\\_imposee](#) for enthalpy equation.

See also: [frontiere\\_ouverte\\_temperature\\_imposee \(11.17\)](#)

Usage:

**entree\_temperature\_imposee\_h ch**

where

- **ch** *champ\_front\_base* ([16.1](#)): Boundary field type.

## 11.4 `frontiere_ouverte`

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

See also: `neumann` ([11.20](#))

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* ([16.1](#)): Boundary field type.

## 11.5 `frontiere_ouverte_concentration_imposee`

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

See also: `dirichlet` ([11.2](#))

Usage:

**`frontiere_ouverte_concentration_imposee ch`**

where

- **`ch`** *champ\_front\_base* ([16.1](#)): Boundary field type.

## 11.6 `frontiere_ouverte_fraction_massique_imposee`

Description: `not_set`

See also: `condlim_base` ([11](#))

Usage:

**`frontiere_ouverte_fraction_massique_imposee ch`**

where

- **`ch`** *champ\_front\_base* ([16.1](#)): Boundary field type.

## 11.7 `frontiere_ouverte_gradient_pression_impose`

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

See also: `neumann` ([11.20](#))

Usage:

**`frontiere_ouverte_gradient_pression_impose ch`**

where

- **`ch`** *champ\_front\_base* ([16.1](#)): Boundary field type.

## 11.8 **frontiere\_ouverte\_gradient\_pression\_impose\_vef**

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

See also: [frontiere\\_ouverte\\_pression\\_imposee \(11.13\)](#) [frontiere\\_ouverte\\_gradient\\_pression\\_impose\\_vefprep1b \(11.9\)](#)

Usage:

**frontiere\_ouverte\_gradient\_pression\_impose\_vef ch**

where

- **ch** *champ\_front\_base* ([16.1](#)): Boundary field type.

## 11.9 **frontiere\_ouverte\_gradient\_pression\_impose\_vefprep1b**

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

See also: [frontiere\\_ouverte\\_gradient\\_pression\\_impose\\_vef \(11.8\)](#)

Usage:

**frontiere\_ouverte\_gradient\_pression\_impose\_vefprep1b ch**

where

- **ch** *champ\_front\_base* ([16.1](#)): Boundary field type.

## 11.10 **frontiere\_ouverte\_gradient\_pression\_libre\_vef**

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

See also: [neumann \(11.20\)](#)

Usage:

**frontiere\_ouverte\_gradient\_pression\_libre\_vef**

## 11.11 **frontiere\_ouverte\_gradient\_pression\_libre\_vefprep1b**

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

See also: [neumann \(11.20\)](#)

Usage:

**frontiere\_ouverte\_gradient\_pression\_libre\_vefprep1b**

## 11.12 **frontiere\_ouverte\_k\_eps\_impose**

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

See also: [condlim\\_base \(11\)](#)

Usage:

**frontiere\_ouverte\_k\_eps\_impose ch**

where

- **ch** *champ\_front\_base* ([16.1](#)): Boundary field type.

### 11.13 frontiere\_ouverte\_pression\_imposee

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

See also: [neumann \(11.20\)](#) [frontiere\\_ouverte\\_gradient\\_pression\\_impose\\_vef \(11.8\)](#)

Usage:

**frontiere\_ouverte\_pression\_imposee ch**

where

- **ch** *champ\_front\_base* ([16.1](#)): Boundary field type.

### 11.14 frontiere\_ouverte\_pression\_imposee\_orlansky

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

See also: [neumann \(11.20\)](#)

Usage:

**frontiere\_ouverte\_pression\_imposee\_orlansky**

### 11.15 frontiere\_ouverte\_pression\_moyenne\_imposee

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

See also: [neumann \(11.20\)](#)

Usage:

**frontiere\_ouverte\_pression\_moyenne\_imposee pext**

where

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

### 11.16 frontiere\_ouverte\_rho\_u\_impose

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

See also: [frontiere\\_ouverte\\_vitesse\\_imposee\\_sortie \(11.19\)](#)

Usage:

**frontiere\_ouverte\_rho\_u\_impose ch**

where

- **ch** *champ\_front\_base* (16.1): Boundary field type.

### 11.17 **frontiere\_ouverte\_temperature\_imposee**

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

See also: *dirichlet* (11.2) *entree\_temperature\_imposee\_h* (11.3)

Usage:

**frontiere\_ouverte\_temperature\_imposee ch**

where

- **ch** *champ\_front\_base* (16.1): Boundary field type.

### 11.18 **frontiere\_ouverte\_vitesse\_imposee**

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

See also: *dirichlet* (11.2) *frontiere\_ouverte\_vitesse\_imposee\_sortie* (11.19)

Usage:

**frontiere\_ouverte\_vitesse\_imposee ch**

where

- **ch** *champ\_front\_base* (16.1): Boundary field type.

### 11.19 **frontiere\_ouverte\_vitesse\_imposee\_sortie**

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

See also: *frontiere\_ouverte\_vitesse\_imposee* (11.18) *frontiere\_ouverte\_rho\_u\_impose* (11.16)

Usage:

**frontiere\_ouverte\_vitesse\_imposee\_sortie ch**

where

- **ch** *champ\_front\_base* (16.1): Boundary field type.

### 11.20 **neumann**

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



See also: `condlim_base` (11) `frontiere_ouverte_gradient_pression_libre_vef` (11.10) `frontiere_ouverte_gradient_pression_libre_vefprep1b` (11.11) `frontiere_ouverte_gradient_pression_impose` (11.7) `frontiere_ouverte_pression_imposee` (11.13) `frontiere_ouverte_pression_imposee_orlansky` (11.14) `frontiere_ouverte_pression_moyenne_imposee` (11.15) `frontiere_ouverte` (11.4) `sortie_libre_temperature_imposee_h` (11.41)

Usage:

**neumann**

## 11.21 paroi\_adiabatique

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

See also: `condlim_base` (11)

Usage:

**paroi\_adiabatique**

## 11.22 paroi\_contact

Description: Thermal condition between two domains. Important: the name of the boundaries in the two domains should be the same. (Warning: there is also an old limitation not yet fixed on the sequential algorithm in VDF to detect the matching faces on the two boundaries: faces should be ordered in the same way). The kind of condition depends on the discretization. In VDF, it is a heat exchange condition, and in VEF, a temperature condition.

Such a coupling requires coincident meshes for the moment. In case of non-coincident meshes, run is stopped and two external files are automatically generated in VEF (`connectivity_failed_boundary_name` and `connectivity_failed_pb_name.med`). In 2D, the keyword `Decouper_bord_coincident` associated to the `connectivity_failed_boundary_name` file allows to generate a new coincident mesh.

In 3D, for a first preliminary cut domain with HOMARD (fluid for instance), the second problem associated to `pb_name` (solide in a fluid/solid coupling problem) has to be submitted to HOMARD cutting procedure with `connectivity_failed_pb_name.med`.

Such a procedure works as while the primary refined mesh (fluid in our example) impacts the fluid/solid interface with a compact shape as described below (values 2 or 4 indicates the number of division from primary faces obtained in fluid domain at the interface after HOMARD cutting):

```
2-2-2-2-2-2
2-4-4-4-4-4-2  2-2-2
2-4-4-4-4-2  2-4-2
2-2-2-2-2  2-2
OK
```

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

See also: `condlim_base` (11)

Usage:

**paroi\_contact autrepb nameb**

where

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

### 11.23 paroi\_contact\_fictif

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

See also: condlim\_base ([11](#))

Usage:

**paroi\_contact\_fictif** **autrepb** **nameb** **conduct\_fictif** **ep\_fictive**  
where

- **autrepb** *str*: Name of other problem.
- **nameb** *str*: Name of bord.
- **conduct\_fictif** *float*: thermal conductivity
- **ep\_fictive** *float*: thickness of the fictitious media

### 11.24 paroi\_couple

Description: not\_set

See also: condlim\_base ([11](#))

Usage:

**paroi\_couple** **autrepb**  
where

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

### 11.25 paroi\_decalee\_robin

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

See also: condlim\_base ([11](#))

Usage:

**paroi\_decalee\_robin** obj Lire obj {  
    **delta** *float*  
}  
where

- **delta** *float*

## 11.26 paroi\_defilante

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

See also: [dirichlet \(11.2\)](#)

Usage:

**paroi\_defilante** **ch**

where

- **ch** *champ\_front\_base* ([16.1](#)): Boundary field type.

## 11.27 paroi\_echange\_contact\_correlation\_vdf

Description: Class to define a thermohydraulic 1D model which will apply to a boundary of 2D or 3D domain.

Warning : For parallel calculation, the only possible partition will be according the axis of the model with the keyword Tranche.

See also: [condlim\\_base \(11\)](#)

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 restarting calculation with this correlation.

## 11.28 paroi\_echange\_contact\_correlation\_vef

Description: Class to define a thermohydraulic 1D model which will apply to a boundary of 2D or 3D domain.

Warning : For parallel calculation, the only possible partition will be according the axis of the model with the keyword Tranche\_geom.

See also: condlim\_base ([11](#))

Usage:

```
paroi_echange_contact_correlation_vef obj Lire obj {
    dir int
    tinf float
    tsup float
    lambda str
    rho str
    cp float
    dt_impr float
    mu str
    debit float
    dh float
    n int
    surface str
    nu str
    xinf float
    xsup float
    [ emissivite_pour_rayonnement_entre_deux_plaques_quasi_infinies float]
    [ reprise_correlation ]
}
where
```

- **dir** *int*: Direction (0 : axis X, 1 : axis Y, 2 : axis Z) of the 1D model.
- **tinf** *float*: Inlet fluid temperature of the 1D model (oC or K).
- **tsup** *float*: Outlet fluid temperature of the 1D model (oC or K).
- **lambda** *str*: Thermal conductivity of the fluid (W.m-1.K-1).
- **rho** *str*: Mass density of the fluid (kg.m-3) which may be a function of the temperature T.
- **cp** *float*: Calorific capacity value at a constant pressure of the fluid (J.kg-1.K-1).
- **dt\_impr** *float*: Printing period in name\_of\_data\_file\_time.dat files of the 1D model results.
- **mu** *str*: Dynamic viscosity of the fluid (kg.m-1.s-1) which may be a function of the temperature T.
- **debit** *float*: Surface flow rate (kg.s-1.m-2) of the fluid into the channel.
- **dh** *float*: Hydraulic diameter may be a function f(x) with x position along the 1D axis (xinf <= x <= xsup)
- **n** *int*: Number of 1D cells of the 1D mesh.
- **surface** *str*: Section surface of the channel which may be function f(Dh,x) of the hydraulic diameter (Dh) and x position along the 1D axis (xinf <= x <= xsup)
- **nu** *str*: Nusselt number which may be a function of the Reynolds number (Re) and the Prandtl number (Pr).
- **xinf** *float*: Position of the inlet of the 1D mesh on the axis direction.

- **xsup** *float*: Position of the outlet of the 1D mesh on the axis direction.
- **emissivite\_pour\_rayonnement\_entre\_deux\_plaques\_quasi\_infinies** *float*: Coefficient of emissivity for radiation between two quasi infinite plates.
- **reprise\_correlation** : Keyword in the case of a restarting calculation with this correlation.

## 11.29 paroi\_echange\_contact\_vdf

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

See also: [condlim\\_base \(11\)](#)

Usage:

**paroi\_echange\_contact\_vdf** **autrepb** **nameb** **temp** **h**  
where

- **autrepb** *str*: Name of other problem.
- **nameb** *str*: Name of bord.
- **temp** *str*: Name of field.
- **h** *float*: Value assigned to a coefficient (expressed in W.K-1m-2) that characterises the contact between the two mediums. In order to model perfect contact, h must be taken to be infinite. This value must obviously be the same in both the two problems blocks.  
The surface thermal flux exchanged between the two mediums is represented by :  
$$f_i = h (T_1 - T_2)$$
 where  $1/h = d_1/\lambda_{a1} + 1/\text{val\_h\_contact} + d_2/\lambda_{a2}$   
where  $d_i$  : distance between the node where  $T_i$  and the wall is found.

## 11.30 paroi\_echange\_externe\_impose

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

See also: [condlim\\_base \(11\)](#) [paroi\\_echange\\_externe\\_impose\\_h \(11.31\)](#)

Usage:

**paroi\_echange\_externe\_impose** **h\_imp** **himpc** **text** **ch**  
where

- **h\_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ\_front\_base (16.1)*: Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ\_front\_base (16.1)*: Boundary field type.

## 11.31 paroi\_echange\_externe\_impose\_h

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

See also: [paroi\\_echange\\_externe\\_impose \(11.30\)](#)

Usage:

**paroi\_echange\_externe\_impose\_h** **h\_imp** **himpc** **text** **ch**  
where

- **h\_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ\_front\_base* (16.1): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ\_front\_base* (16.1): Boundary field type.

### 11.32 paroi\_echange\_global\_impose

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

See also: [condlim\\_base \(11\)](#)

Usage:

**paroi\_echange\_global\_impose h\_imp himpc text ch**

where

- **h\_imp** *str*: Global exchange coefficient value. The global exchange coefficient value is expressed in W.m-2.K-1.
- **himpc** *champ\_front\_base* (16.1): Boundary field type.
- **text** *str*: External temperature value. The external temperature value is expressed in oC or K.
- **ch** *champ\_front\_base* (16.1): Boundary field type.

### 11.33 paroi\_fixe

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

See also: [condlim\\_base \(11\)](#) [paroi\\_fixe\\_iso\\_Genepi2\\_sans\\_contribution\\_aux\\_vitesses\\_sommets \(11.34\)](#)

Usage:

**paroi\_fixe**

### 11.34 paroi\_fixe\_iso\_Genepi2\_sans\_contribution\_aux\_vitesses\_sommets

Description: CL pour obtenir iso Geneppi2, sans interet

See also: [paroi\\_fixe \(11.33\)](#)

Usage:

**paroi\_fixe\_iso\_Genepi2\_sans\_contribution\_aux\_vitesses\_sommets**

### 11.35 paroi\_flux\_impose

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

See also: [condlim\\_base \(11\)](#)

Usage:

**paroi\_flux\_impose ch**

where

- **ch** *champ\_front\_base* (16.1): Boundary field type.

### 11.36 paroi\_knudsen\_non\_negligeable

Description: Boundary condition for number of Knudsen ( $Kn$ ) above 0.001 where slip-flow condition appears: the velocity near the wall depends on the shear stress :  $Kn=l/L$  with  $l$  is the mean-free-path of the molecules and  $L$  a characteristic length scale.

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

Where  $k$  is a coefficient given by several laws:

Mawxell :  $k=(2-s)*l/s$

Bestok&Karniadakis :  $k=(2-s)/s*L*Kn/(1+Kn)$

Xue&Fan :  $k=(2-s)/s*L*tanh(Kn)$

$s$  is a value between 0 and 2 named accomodation coefficient.  $s=1$  seems a good value.

Warning : The keyword is available for VDF calculation only for the moment.

See also: [dirichlet \(11.2\)](#)

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 (16.1)*: Boundary field type.
- **name\_champ\_2** *str into ['vitesse\_paro', 'k']*: Field name.
- **champ\_2** *champ\_front\_base (16.1)*: Boundary field type.

### 11.37 paroi\_rugueuse

Description: Rough wall boundary

See also: [dirichlet \(11.2\)](#)

Usage:

**paroi\_rugueuse** **obj** Lire **obj** {

**erugu** *float*

}

where

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

### 11.38 paroi\_temperature\_imposee

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

See also: [dirichlet \(11.2\)](#) [temperature\\_imposee\\_paro \(11.43\)](#)

Usage:

**paroi\_temperature\_imposee** **ch**

where

- **ch** *champ\_front\_base (16.1)*: Boundary field type.

### 11.39 periodique

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

See also: `condlim_base` ([11](#))

Usage:

**periodique**

### 11.40 scalaire\_impose\_paro

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

See also: `dirichlet` ([11.2](#))

Usage:

**scalaire\_impose\_paro ch**

where

- **ch** *champ\_front\_base* ([16.1](#)): Boundary field type.

### 11.41 sortie\_libre\_temperature\_imposee\_h

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

See also: `neumann` ([11.20](#))

Usage:

**sortie\_libre\_temperature\_imposee\_h ch**

where

- **ch** *champ\_front\_base* ([16.1](#)): Boundary field type.

### 11.42 symetrie

Description: 1). For NAVIER STOKES equations, this keyword is used to designate a symmetry condition concerning the speed at the boundary called bord (edge) (normal speed at the edge equal to zero and tangential speed gradient at the edge equal to zero); 2). For scalar transport equation, this keyword is used to set a symmetry condition on scalar on the boundary named bord (edge).

See also: `condlim_base` ([11](#))

Usage:

**symetrie**

### 11.43 temperature\_imposee\_paro

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

See also: `paroi_temperature_imposee` ([11.38](#))



Usage:

**temperature\_imposee\_paro** **ch**

where

- **ch** *champ\_front\_base* (16.1): Boundary field type.

## 12 discretisation\_base

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

See also: *objet\_u* (33) *vdf* (12.2) *vef* (12.3) *ef* (12.1)

Usage:

### 12.1 ef

Description: Element Finite discretization.

See also: *discretisation\_base* (12)

Usage:

### 12.2 vdf

Description: Finite difference volume discretization.

See also: *discretisation\_base* (12)

Usage:

### 12.3 vef

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

Warning: it becomes an obsolete discretization.

See also: *discretisation\_base* (12) *vefprep1b* (12.4)

Usage:

### 12.4 vefprep1b

Description: Finite element volume discretization (P1NC/P1-bubble element). Since the 1.5.5 version, several new discretizations are available thanks to the optional keyword *Lire*. By default, the *VEFPreP1B* keyword is equivalent to the former *VEFPreP1B* formulation (v1.5.4 and sooner). *P0P1* (if used with the strong formulation for imposed pressure boundary) is equivalent to *VEFPreP1B* but the convergence is slower. *VEFPreP1B dis* is equivalent to *VEFPreP1B dis Lire dis { P0 P1 Changement\_de\_base\_P1Bulle 1 Cl\_preSSION\_sommet\_faible 0 }*

See also: *vef* (12.3)

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.

## 13 domaine

Description: Keyword to create a domain.

See also: [objet\\_u \(33\)](#)

Usage:

## 14 espece

Description: not\_set

See also: [objet\\_u \(33\)](#)

Usage:

```

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

```

- **cp** *champ\_base* [\(15.1\)](#): Specific heat value (J.kg-1.K-1).
- **lambda** *champ\_base* [\(15.1\)](#): Conductivity value (W.m-1.K-1).
- **mu** *champ\_base* [\(15.1\)](#): Dynamic viscosity value (kg.m-1.s-1).
- **masse\_molaire** *float*: Gas molar mass.

## 15 champ\_base

### 15.1 champ\_base

Description: Basic class of fields.

See also: objet\_u ([33](#)) champ\_don\_base ([15.2](#)) champ\_ostwald ([15.15](#)) champ\_input\_base ([15.13](#)) champ\_fonc\_med ([15.6](#)) field\_uniform\_keps\_from\_ud ([15.23](#))

Usage:

### 15.2 champ\_don\_base

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

See also: champ\_base ([15.1](#)) uniform\_field ([15.26](#)) champ\_uniforme\_morceaux ([15.19](#)) champ\_fonc\_xyz ([15.22](#)) champ\_fonc\_txyz ([15.21](#)) champ\_don\_lu ([15.3](#)) init\_par\_partie ([15.24](#)) champ\_tabule\_temps ([15.18](#)) champ\_fonc\_t ([15.9](#)) champ\_fonc\_tabule ([15.10](#)) champ\_init\_canal\_sinal ([15.11](#)) champ\_som\_lu\_vdf ([15.16](#)) champ\_som\_lu\_vef ([15.17](#)) tayl\_green ([15.25](#)) champ\_fonc\_reprise ([15.7](#))

Usage:

### 15.3 champ\_don\_lu

Description: Field to read a data field (values located at the center of the cells) in a file.

See also: champ\_don\_base ([15.2](#))

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

### 15.4 champ\_fonc\_fonction

Description: Field that is a function of another field.

See also: champ\_fonc\_tabule ([15.10](#)) champ\_fonc\_fonction\_txyz ([15.5](#))

Usage:

**champ\_fonc\_fonction dim inco bloc**  
where

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

- **bloc** *bloc\_lecture* (3.38): 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)).

## 15.5 champ\_fonc\_fonction\_txyz

Description: this refers to a field that is a function of another field and time and/or space coordinates

See also: `champ_fonc_fonction` (15.4)

Usage:

**champ\_fonc\_fonction\_txyz dim inco bloc**  
where

- **dim** *int*: Number of field components.
- **inco** *str*: Name of the field (for example: temperature).
- **bloc** *bloc\_lecture* (3.38): 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)).

## 15.6 champ\_fonc\_med

Description: Field to read a data field in a MED-format file .med at a specified time. It is very useful, for example, to restart a calculation with a new or refined geometry. The field post-processed on the new geometry at med format is used as initial condition for restarting.

See also: `champ_base` (15.1)

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.

## 15.7 champ\_fonc\_reprise

Description: This field is used to read a data field in a save file (.xyz or .sauv) at a specified time. It is very useful, for example, to run a thermohydraulic calculation with velocity initial condition read into a save file from a previous hydraulic calculation.

See also: `champ_don_base` (15.2)

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* (15.8): 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.

## 15.8 fonction\_champ\_reprise

Description: not\_set

See also: objet\_lecture (32)

Usage:

**mot fonction**

where

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

## 15.9 champ\_fonc\_t

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

See also: champ\_don\_base (15.2)

Usage:

**champ\_fonc\_t val**

where

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

## 15.10 champ\_fonc\_tabule

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

See also: champ\_don\_base (15.2) champ\_fonc\_fonction (15.4)

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* (3.38): 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)).

## 15.11 champ\_init\_canal\_sinal

Description: For a parabolic profile on U velocity with an unpredictable disturbance on V and W and a sinusoidal disturbance on V velocity.

See also: *champ\_don\_base* (15.2)

Usage:

**champ\_init\_canal\_sinal** *dim* **bloc**

where

- **dim** *int*: Number of field components.
- **bloc** *bloc\_lec\_champ\_init\_canal\_sinal* (15.12): Parameters for the class *champ\_init\_canal\_sinal*.

## 15.12 bloc\_lec\_champ\_init\_canal\_sinal

Description: Parameters for the class *champ\_init\_canal\_sinal*.

in 2D:

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

$V = ampli\_bruit * rand + ampli\_sin * \sin(\omega * x)$

rand: unpredictable value between -1 and 1.

in 3D:

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

$V = ampli\_bruit * rand1 + ampli\_sin * \sin(\omega * x)$

$W = ampli\_bruit * rand2$

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

See also: *objet\_lecture* (32)

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.

### 15.13 champ\_input\_base

Description: not\_set

See also: champ\_base ([15.1](#)) champ\_input\_p0 ([15.14](#))

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*

### 15.14 champ\_input\_p0

Description: not\_set

See also: champ\_input\_base ([15.13](#))

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

### 15.15 champ\_ostwald

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

See also: [champ\\_base \(15.1\)](#)

Usage:  
**champ\_ostwald**

### 15.16 champ\_som\_lu\_vdf

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

See also: [champ\\_don\\_base \(15.2\)](#)

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 Wi+1 -> Next value ...

### 15.17 champ\_som\_lu\_vdf

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

See also: [champ\\_don\\_base \(15.2\)](#)

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 ...

## 15.18 champ\_tabule\_temps

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

See also: champ\_don\_base (15.2)

Usage:

**champ\_tabule\_temps dim bloc**  
where

- **dim** *int*: Number of field components.
- **bloc** *bloc\_lecture* (3.38): Values as a table. The value of the field at any time is calculated by linear interpolation from this table.

## 15.19 champ\_uniforme\_morceaux

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

See also: champ\_don\_base (15.2) champ\_uniforme\_morceaux\_tabule\_temps (15.20) valeur\_totale\_sur\_volume (15.27)

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* (3.38): { Default val\_def sous\_zone\_1 val\_1 ... sous\_zone\_i val\_i } By default, the value val\_def is assigned to the field. It takes the sous\_zone\_i identifier Sous\_Zone (sub\_area) type object value, val\_i. Sous\_Zone (sub\_area) type objects must have been previously defined if the operator wishes to use a Champ\_Uniforme\_Morceaux(partly\_uniform\_field) type object.

## 15.20 champ\_uniforme\_morceaux\_tabule\_temps

Description: this type of field is constant in space on one or several sub\_zones and tabulated as a function of time.

See also: champ\_uniforme\_morceaux (15.19)

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* (3.38): { Defaut val\_def sous\_zone\_1 val\_1 ... sous\_zone\_i val\_i } By default, the value val\_def is assigned to the field. It takes the sous\_zone\_i identifier Sous\_Zone (sub\_area) type object value, val\_i. Sous\_Zone (sub\_area) type objects must have been previously defined if the operator wishes to use a Champ\_Uniforme\_Morceaux(partly\_uniform\_field) type object.

### 15.21 champ\_fonc\_txyz

Description: Field defined by analytical functions. It makes it possible the definition of a field that depends on the time and the space.

See also: champ\_don\_base (15.2)

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).

### 15.22 champ\_fonc\_xyz

Description: Field defined by analytical functions. It makes it possible the definition of a field that depends on (x,y,z).

See also: champ\_don\_base (15.2)

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).

### 15.23 field\_uniform\_keps\_from\_ud

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

See also: champ\_base (15.1)

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

## 15.24 **init\_par\_partie**

Description: ne marche que pour `n_comp=1`

See also: `champ_don_base` ([15.2](#))

Usage:

**init\_par\_partie** **n\_comp** **val1** **val2** **val3**

where

- **n\_comp** *int into [1]*
- **val1** *float*
- **val2** *float*
- **val3** *float*

## 15.25 **tayl\_green**

Description: Class `Tayl_green`.

See also: `champ_don_base` ([15.2](#))

Usage:

**tayl\_green** **dim**

where

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

## 15.26 **uniform\_field**

Synonymous: **champ\_uniforme**

Description: Field that is constant in space and stationary.

See also: `champ_don_base` ([15.2](#))

Usage:

**uniform\_field** **val**

where

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

## 15.27 **valeur\_totale\_sur\_volume**

Description: Similar as `Champ_Uniforme_Morceaux` with the same syntax. Used for source terms when we want to specify a source term with a value given for the volume (eg: heat in Watts) and not a value per volume unit (eg: heat in Watts/m3).

See also: `champ_uniforme_morceaux` ([15.19](#))

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* (3.38): { 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 champ\_front\_base

### 16.1 champ\_front\_base

Description: Basic class for fields at domain boundaries.

See also: objet\_u (33) champ\_front\_uniforme (16.21) champ\_front\_fonc\_xyz (16.13) champ\_front\_fonc\_txyz (16.12) champ\_front\_fonc\_pois\_ipsn (16.10) champ\_front\_fonc\_pois\_tube (16.11) champ\_front\_tabule (16.19) champ\_front\_fonction (16.14) champ\_front\_bruite (16.6) champ\_front\_tangentiel\_vef (16.20) champ\_front\_lu (16.15) boundary\_field\_inward (16.2) champ\_front\_pression\_from\_u (16.17) champ\_front\_debit (16.9) champ\_front\_contact\_vef (16.8) champ\_front\_calc (16.7) champ\_front\_recyclage (16.18) ch\_front\_input (16.4) boundary\_field\_uniform\_keps\_from\_ud (16.3) champ\_front\_normal\_vef (16.16)

Usage:

### 16.2 boundary\_field\_inward

Description: this field is used to define the normal vector field standard at the boundary in VDF or VEF discretization.

See also: champ\_front\_base (16.1)

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.

### 16.3 boundary\_field\_uniform\_keps\_from\_ud

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

See also: champ\_front\_base (16.1)

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

## 16.4 ch\_front\_input

Description: not\_set

See also: champ\_front\_base ([16.1](#)) ch\_front\_input\_uniforme ([16.5](#))

Usage:

**ch\_front\_input** obj Lire obj {

```

    nb_comp  int
    nom      str
    [ initial_value  n x1 x2 ... xn]
    probleme  str
    [ sous_zone      str]

```

}

where

- **nb\_comp** *int*
- **nom** *str*
- **initial\_value** *n x1 x2 ... xn*
- **probleme** *str*
- **sous\_zone** *str*

## 16.5 ch\_front\_input\_uniforme

Description: for coupling, you can use ch\_front\_input\_uniforme which is a champ\_front\_uniforme, which use an external value. It must be used with Problem.setInputField.

See also: ch\_front\_input ([16.4](#))

Usage:

**ch\_front\_input\_uniforme** obj Lire obj {

```

    nb_comp  int
    nom      str
    [ initial_value  n x1 x2 ... xn]
    probleme  str
    [ sous_zone      str]

```

}

where

- **nb\_comp** *int* for inheritance
- **nom** *str* for inheritance
- **initial\_value** *n x1 x2 ... xn* for inheritance
- **probleme** *str* for inheritance
- **sous\_zone** *str* for inheritance

## 16.6 champ\_front\_bruit

Description: Field which is variable in time and space in a random manner.

See also: `champ_front_base` ([16.1](#))

Usage:

**champ\_front\_bruit nb\_comp bloc**

where

- **nb\_comp** *int*: Number of field components.
- **bloc** *bloc\_lecture* ([3.38](#)): { [N val L val ] Moyenne  $m_1, \dots, [m_i]$  Amplitude  $A_1, \dots, [A_i]$  }:  
Random noise: If N and L are not defined, the *i*th component of the field varies randomly around an average value  $m_i$  with a maximum amplitude  $A_i$ .  
White noise: If N and L are defined, these two additional parameters correspond to L, the domain length and N, the number of nodes in the domain. Noise frequency will be between  $2\pi/L$  and  $2\pi N/(4L)$ .  
For example, formula for speed:  $u=U_0(t)$   $v=U_1(t)U_j(t)=M_j+2A_j\text{bruit\_blanc}$  where `bruit_blanc` (white\_noise) is the formula given in the `mettre_a_jour` (update) method of the `Champ_front_bruit` (noise\_boundary\_field) (Refer to the `Ch_fr_bruit.cpp` file).

## 16.7 champ\_front\_calc

Description: This keyword is used on a boundary to get a field from another boundary. The local and remote boundaries should have the same mesh. If not, the `Champ_front_recyclage` keyword could be used instead. It is used in the condition block at the limits of equation which itself refers to a problem called `pb1`. We are working under the supposition that `pb1` is coupled to another problem.

See also: `champ_front_base` ([16.1](#))

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 recognised by the `problem_name` object.

## 16.8 champ\_front\_contact\_vef

Description: This field is used on a boundary between a solid and fluid domain to exchange a calculated temperature at the contact face of the two domains according to the flux of the two problems.

See also: `champ_front_base` ([16.1](#))

Usage:

**champ\_front\_contact\_vef local\_pb local\_boundary remote\_pb remote\_boundary**

where

- **local\_pb** *str*: Name of the problem.
- **local\_boundary** *str*: Name of the boundary.
- **remote\_pb** *str*: Name of the second problem.
- **remote\_boundary** *str*: Name of the boundary in the second problem.

## 16.9 champ\_front\_debit

Description: This field is used to define a flow rate field instead of a velocity field for a Dirichlet boundary condition on Navier Stokes equation.

See also: `champ_front_base` ([16.1](#))

Usage:

**champ\_front\_debit ch**

where

- **ch** *champ\_front\_base* ([16.1](#)): field (`champ_front_uniforme`) to define the flow rate.

## 16.10 champ\_front\_fonc\_pois\_ipsn

Description: Boundary field `champ_front_fonc_pois_ipsn`.

See also: `champ_front_base` ([16.1](#))

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)*

## 16.11 champ\_front\_fonc\_pois\_tube

Description: Boundary field `champ_front_fonc_pois_tube`.

See also: `champ_front_base` ([16.1](#))

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)*

## 16.12 champ\_front\_fonc\_txyz

Description: Boundary field which is not constant in space and in time.

See also: `champ_front_base` ([16.1](#))

Usage:

**champ\_front\_fonc\_txyz val**

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

### 16.13 champ\_front\_fonc\_xyz

Description: Boundary field which is not constant in space.

See also: champ\_front\_base ([16.1](#))

Usage:

**champ\_front\_fonc\_xyz val**

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

### 16.14 champ\_front\_fonction

Description: boundary field that is function of another field

See also: champ\_front\_base ([16.1](#))

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.

### 16.15 champ\_front\_lu

Description: boundary field which is given from data issued from a read file. The format of this file has to be the same that the one generated by Ecrire\_fichier\_xyz\_valeur

Example for K and epsilon quantities to be defined for inlet condition in a boundary named 'entree':

entree frontiere\_ouverte\_K\_Eps\_impose Champ\_Front\_lu dom 2pb\_K\_EPS\_PERIO\_1006.306198.dat

See also: champ\_front\_base ([16.1](#))

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

### 16.16 champ\_front\_normal\_vef

Description: Field to define the normal vector field standard at the boundary in VEF discretization.

See also: champ\_front\_base ([16.1](#))

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).

## 16.17 champ\_front\_pression\_from\_u

Description: this field is used to define a pressure field depending of a velocity field.

See also: champ\_front\_base (16.1)

Usage:

**champ\_front\_pression\_from\_u** *expression*

where

- **expression** *str*: value depending of a velocity (like  $2 * u_{moy}^2$ ).

## 16.18 champ\_front\_recyclage

Description: This keyword is used on a boundary to get a field from another boundary. New keyword in the 1.6.1 version which replaces and generalizes several obsolete ones:

```

Champ_front_calc_intern
Champ_front_calc_recycl_fluct_pbperio
Champ_front_calc_recycl_champ
Champ_front_calc_intern_2pbs
Champ_front_calc_recycl_fluct
Champ_front_recyclage {
pb_champ_evaluateur pb field nb_comp
[ distance_plan dist0 dist1 [dist2] ]
[ moyenne_imposee methode_moy [fichier file [second_file] ]
[ moyenne_recyclee methode_recyc [fichier file [second_file] ]
[ direction_anisotrope 1|2|3 ]
[ ampli_moyenne_imposee 2|3 alpha(0) alpha(1) [alpha(2)] ]
[ ampli_moyenne_recyclee 2|3 beta(0) beta(1) [beta(2)] ]
[ ampli_fluctuation 2|3 gamma(0) gamma(1) [gamma(2)] ]
}

```

This keyword is to use, in a general way, on a boundary of a local\_pb problem, a field calculated from a linear combination of an imposed field  $g(x,y,z,t)$  with an instantaneous  $f(x,y,z,t)$  and a spatial mean field  $\langle f \rangle(t)$  or a temporal mean field  $\langle f \rangle(x,y,z)$  field extracted from a plane of a problem named pb (pb may be local\_pb itself) :

For each component  $i$ , the field  $F$  applied on the boundary will be:

$$F_i(x,y,z,t) = \alpha_i g_i(x,y,z,t) + xsi_i [f_i(x,y,z,t) - \beta_i \langle f_i \rangle]$$

The different options are:

**pb\_champ\_evaluateur** pb field nb\_comp : To give the name of the pb problem, the name of the field of the problem and its number of components nb\_comp.

**distance\_plan** dist0 dist1 [dist2] : Vector which gives the distance between the boundary and the plane from where the field  $F$  will be extracted. By default, the vector is zero, that should imply the two domains have coincident boundaries.

**ampli\_moyenne\_imposee** 2|3 alpha(0) alpha(1) [alpha(2)] :  $\alpha_i$  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** direction : If an integer is given for direction (X:1, Y:2, Z:3, by default, direction is negative), the imposed field  $g$  will be 0 for the 2 other directions.

**moyenne\_imposee** methode\_moy : Value of the imposed  $g$  field. The methode\_moy option can be :

profil [2l3] valx(x,y,z,t) valy(x,y,z,t) [valz(x,y,z,t)] : to specify analytic profile for the imposed g field.  
 interpolation fichier file : to create a imposed field built by interpolation of values read into a file. The imposed field is applied on the direction given by the keyword direction\_anisotrope (the field is zero for the other directions). The format of the file is:

```
pos(1) val(1)
pos(2) val(2)
```

```
pos(N) val(N)
```

If direction given by direction\_anisotrope is 1 (or 2 or 3), then pos will be X (or Y or Z) coordinate and val will be X value (or Y value, or Z value) of the imposed field.

connexion\_approchee fichier file : to read the imposed field into a file where positions and values are given (it is not necessary that the coordinates of the points match the coordinates of the faces of the boundary, indeed, the nearest point of each face of the boundary will be used). The format of the file is:

```
N
x(1) y(1) [z(1)] valx(1) valy(1) [valz(1)]
x(2) y(2) [z(2)] valx(2) valy(2) [valz(2)]
```

```
x(N) y(N) [z(N)] valx(N) valy(N) [valz(N)]
```

connection\_exacte fichier file second\_file : to read the imposed field into two files. The first file contains the points coordinates (which should be the same than the coordinates of each faces of the boundary) and the second\_file contains the mean values. The format of the first file is:

```
N
1 x(1) y(1) [z(1)]
2 x(2) y(2) [z(2)]
```

```
N x(N) y(N) [z(N)]
```

The format of the second\_file is:

```
N
1 valx(1) valy(1) [valz(1)]
2 valx(2) valy(2) [valz(2)]
```

...

```
N valx(N) valy(N) [valz(N)]
```

logarithmique diametre double u\_tau double visco\_cin double direction integer : to specify the imposed field (in this case, velocity) by an analytical logarithmic law of the wall :

$$g(x,y,z) = u\_tau * ( \log(0.5*diametre*u\_tau/visco\_cin)/Kappa + 5.1 )$$

With  $g(x,y,z)=u(x,y,z)$  if direction is set to 1 ( $g=v(x,y,z)$  if direction is set to 2, and  $g=w(x,y,z)$  if set to 3)  
 moyenne\_recyclee methode\_recyc : Method used to do a spatial or a temporal averaging of f field to specify <f>. <f> can be the surface mean of f on the plane (surface option, see below) or it can be read from several files (for example generated by the chmoy\_faceperio option of the Traitement\_particulier keyword to obtain a temporal mean field). The option methode\_recyc can be :

surfacique : surface mean for <f> from f values on the plane

Same options of methode\_moy options but applied to read a temporal mean field <f>(x,y,z):

interpolation

connexion\_approchee fichier file

connexion\_exacte fichier file second\_file

See also: champ\_front\_base ([16.1](#))

Usage:

**champ\_front\_recyclage bloc**

where

- **bloc** *str*

## 16.19 champ\_front\_tabule

Description: Constant field on the boundary, tabulated as a function of time.

See also: `champ_front_base` ([16.1](#))

Usage:

**champ\_front\_tabule nb\_comp bloc**

where

- **nb\_comp** *int*: Number of field components.
  - **bloc** *bloc\_lecture* ([3.38](#)): {nt1 t2 t3 ...tn u1 [v1 w1 ...] u2 [v2 w2 ...] u3 [v3 w3 ...] ... un [vn wn ...]}
- Values are entered into a table based on n couples (ti, ui) if nb\_comp value is 1. The value of a field at a given time is calculated by linear interpolation from this table.

## 16.20 champ\_front\_tangentiel\_vef

Description: Field to define the tangential speed vector field standard at the boundary in VEF discretisation.

See also: `champ_front_base` ([16.1](#))

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].

## 16.21 champ\_front\_uniforme

Description: Boundary field which is constant in space and stationary.

See also: `champ_front_base` ([16.1](#))

Usage:

**champ\_front\_uniforme val**

where

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

## 17 loi\_etat\_base

Description: Basic class for state laws.

See also: `objet_u` ([33](#)) `gaz_parfait` ([17.3](#)) `melange_gaz_parfait` ([17.2](#)) `gaz_reel_rhot` ([17.1](#))

Usage:

## 17.1 gaz\_reel\_rhot

Description: Real gas.

See also: `loi_etat_base` ([17](#))

Usage:

**gaz\_reel\_rhot** **bloc**

where

- **bloc** *bloc\_lecture* ([3.38](#)): Description.

## 17.2 melange\_gaz\_parfait

Description: Mixing of perfect gas.

See also: `loi_etat_base` ([17](#))

Usage:

**melange\_gaz\_parfait** obj Lire obj {

    [ **Sc** *float*]

**Prandtl** *float*

}

where

- **Sc** *float*: Schmidt number of the gas  $Sc = \nu/D$  (D: diffusion coefficient of the mixing).
- **Prandtl** *float*: Prandtl number of the gas  $Pr = \mu * Cp / \lambda$

## 17.3 gaz\_parfait

Description: Perfect gas.

See also: `loi_etat_base` ([17](#))

Usage:

**gaz\_parfait** obj Lire obj {

**Cp** *float*

    [ **Cv** *float*]

    [ **gamma** *float*]

**Prandtl** *float*

    [ **rho\_constant\_pour\_debug** *champ\_base*]

}

where

- **Cp** *float*: Specific heat at constant pressure (J/kg/K).
- **Cv** *float*: Specific heat at constant volume (J/kg/K).
- **gamma** *float*:  $Cp/Cv$
- **Prandtl** *float*: Prandtl number of the gas  $Pr = \mu * Cp / \lambda$
- **rho\_constant\_pour\_debug** *champ\_base* ([15.1](#))

## 18 loi\_fermeture\_base

Description: Class for appends fermeture to problem

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

See also: objet\_u (33) loi\_fermeture\_test (18.1)

Usage:

### 18.1 loi\_fermeture\_test

Description: Loi for test only

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

See also: loi\_fermeture\_base (18)

Usage:

**loi\_fermeture\_test** obj Lire obj {

    [ **coef** *float*]

}

where

- **coef** *float*: coefficient

## 19 loi\_horaire

Description: to define the movement with a time-dependant law for the solid interface.

See also: objet\_u (33)

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*

## 20 milieu\_base

Description: Basic class for medium (physics properties of medium).

See also: objet\_u (33) solide (20.6) constituant (20.1) fluide\_incompressible (20.2)

Usage:

**milieu\_base** obj Lire obj {

```

    [ rho  champ_base]
    [ cp  champ_base]
    [ lambda  champ_base]
}
where

```

- **rho** *champ\_base* (15.1): Density (kg.m-3).
- **cp** *champ\_base* (15.1): Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* (15.1): Conductivity (W.m-1.K-1).

## 20.1 constituent

Description: Constituent.

See also: milieu\_base (20)

Usage:

```

constituant obj Lire obj {
    [ coefficient_diffusion  champ_base]
    [ rho  champ_base]
    [ cp  champ_base]
    [ lambda  champ_base]
}
where

```

- **coefficient\_diffusion** *champ\_base* (15.1): Constituent diffusion coefficient value (m2.s-1). If a multi-constituent problem is being processed, the diffusivity will be a vectorial and each components will be the diffusion of the constituent.
- **rho** *champ\_base* (15.1) for inheritance: Density (kg.m-3).
- **cp** *champ\_base* (15.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* (15.1) for inheritance: Conductivity (W.m-1.K-1).

## 20.2 fluide\_incompressible

Description: This is a uncompressible fluid.

See also: milieu\_base (20) fluide\_quasi\_compressible (20.4) fluide\_ostwald (20.3)

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* (15.1): Thermal expansion (K-1).
- **mu** *champ\_base* (15.1): Dynamic viscosity (kg.m-1.s-1).
- **beta\_co** *champ\_base* (15.1): Volume expansion coefficient values in concentration.
- **indice** *champ\_base* (15.1): Refractivity of fluid.
- **kappa** *champ\_base* (15.1): Absorptivity of fluid (m-1).
- **rho** *champ\_base* (15.1) for inheritance: Density (kg.m-3).
- **cp** *champ\_base* (15.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* (15.1) for inheritance: Conductivity (W.m-1.K-1).

### 20.3 fluide\_ostwald

Description: Non-Newtonian fluids governed by Ostwald's law. The law applicable to stress tensor is:

$\tau = K(T) \cdot (D:D)^{1/n} \cdot D$  Where:

D refers to the deformation speed tensor

K refers to fluid consistency (may be a function of the temperature T)

n refers to the fluid structure index  $n=1$  for a Newtonian fluid,  $n<1$  for a rheofluidifier fluid,  $n>1$  for a rheothickening fluid.

See also: [fluide\\_incompressible \(20.2\)](#)

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* (15.1): Fluid consistency.
- **n** *champ\_base* (15.1): Fluid structure index.
- **beta\_th** *champ\_base* (15.1) for inheritance: Thermal expansion (K-1).
- **mu** *champ\_base* (15.1) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **beta\_co** *champ\_base* (15.1) for inheritance: Volume expansion coefficient values in concentration.
- **indice** *champ\_base* (15.1) for inheritance: Refractivity of fluid.
- **kappa** *champ\_base* (15.1) for inheritance: Absorptivity of fluid (m-1).
- **rho** *champ\_base* (15.1) for inheritance: Density (kg.m-3).
- **cp** *champ\_base* (15.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* (15.1) for inheritance: Conductivity (W.m-1.K-1).

### 20.4 fluide\_quasi\_compressible

Description: Compressible flow at low mach number.

See also: [fluide\\_incompressible \(20.2\)](#)

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* (20.5): Sutherland law for viscosity and for conductivity.
- **pression** *float*: Initial pression.
- **loi\_etat** *loi\_etat\_base* (17): State law.
- **traitement\_pth** *str into ['edo', 'constant', 'conservation\_masse']*: Particular treatment for the thermodynamic pressure Pth ; there are three possibilities:
  - 1) with the keyword 'edo' the code computes Pth solving an O.D.E. ; in this case, the mass is not strictly conserved (it is the default case for quasi compressible computation);
  - 2) the keyword 'conservation\_masse' forces the conservation of the mass (closed geometry or with periodic boundaries condition)
  - 3) the keyword 'constant' makes it possible to have a constant Pth ; it's the good choice when the flow is open (e.g. with pressure boundary conditions).
- **traitement\_rho\_gravite** *str into ['standard', 'moins\_rho\_moyen']*: It may be :1) 'standard': the gravity term is evaluated with  $\rho \cdot g$  (It is the default). 2) 'moins\_rho\_moyen': the gravity term is evaluated with  $(\rho - \rho_{\text{moy}}) \cdot g$ .
- **temps\_debut\_prise\_en\_compte\_drho\_dt** *float*: While time < value, dRho/dt is set to zero (Rho, volumic mass). Useful for some calculation during the first time steps with big variation of temperature and volumic mass.
- **omega\_relaxation\_drho\_dt** *float*: Optional option to have a relaxed algorithm to solve the mass equation. value is used (1 per default) to specify omega.
- **mu** *champ\_base* (15.1) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **indice** *champ\_base* (15.1) for inheritance: Refractivity of fluid.
- **kappa** *champ\_base* (15.1) for inheritance: Absorptivity of fluid (m-1).
- **rho** *champ\_base* (15.1) for inheritance: Density (kg.m-3).
- **cp** *champ\_base* (15.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* (15.1) for inheritance: Conductivity (W.m-1.K-1).

## 20.5 bloc\_sutherland

Description: Sutherland law for viscosity  $\mu(T) = \mu_0 \cdot ((T_0 + C)/(T + C)) \cdot (T/T_0)^{1.5}$  and (optional) for conductivity  $\lambda(T) = \mu_0 \cdot C_p / \text{Prandtl} \cdot ((T_0 + S\lambda)/(T + S\lambda)) \cdot (T/T_0)^{1.5}$

See also: objet\_lecture (32)

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*

## 20.6 solide

Description: Solid.

See also: milieu\_base (20)

Usage:

```
solide obj Lire obj {  
    [ rho champ_base]  
    [ cp champ_base]  
    [ lambda champ_base]  
}
```

where

- **rho** *champ\_base* (15.1) for inheritance: Density (kg.m-3).
- **cp** *champ\_base* (15.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ\_base* (15.1) for inheritance: Conductivity (W.m-1.K-1).

## 21 modele\_turbulence\_scal\_base

Description: Basic class for turbulence model for energy equation.

See also: objet\_u (33) prandtl (21.1) schmidt (21.2)

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* (30): 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».

## 21.1 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 (21)

Usage:

```
prandtl obj Lire obj {  
    [ prdt str]  
    [ prandt_turbulent_fonction_nu_t_alpha str]  
    [ turbulence_paroi turbulence_paroi_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,  $\alpha_t = \nu_t / \text{Prt}$ ) with another formulae, for example:  $\alpha_t = \nu_t^2 / (0.7 * \alpha + 0.85 * \nu_{tt})$  with the string  $\nu_t * \nu_t / (0.7 * \alpha + 0.85 * \nu_t)$  where  $\alpha$  is the thermal diffusivity.
- **turbulence\_paro***i* *turbulence\_paro**i\_scalaire\_base* (30) 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 :  $\text{Nu} = ((\lambda + \lambda_{dt}) / \lambda) * d_{\text{wall}} / d_{\text{eq}}$  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  $h = (\lambda + \lambda_{dt}) / d_{\text{eq}}$  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».

## 21.2 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 (21)

Usage:

```
schmidt obj Lire obj {  
    [ seturb float]  
    [ turbulence_paroi turbulence_paroi_scalaire_base]  
    [ dt_impr_nusselt float]
```

```
}
```

where

- **seturb** *float*: Keyword to modify the constant (Sct) of Schmlidt model :  $Dt=Nut/Sct$  Default value is 0.7.
- **turbulence\_paro** *turbulence\_paro\_scalaire\_base* (30) 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».

## 22 nom

Description: Class to name the TRUST objects.

See also: `objet_u` (33) `nom_anonyme` (22.1)

Usage:

**nom** [ **mot** ]

where

- **mot** *str*: Chain of characters.

### 22.1 nom\_anonyme

Description: `not_set`

See also: `nom` (22)

Usage:

[ **mot** ]

where

- **mot** *str*: Chain of characters.

## 23 partitionneur\_deriv

Description: `not_set`

See also: `objet_u` (33) `metis` (23.2) `sous_zones` (23.4) `tranche` (23.5) `partition` (23.3) `fichier_decoupage` (23.1)

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).

## 23.1 fichier\_decoupage

Description: This algorithm reads an array of integer values on the disc, one value for each mesh element. Each value is interpreted as the target part number  $n \geq 0$  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` ([23](#))

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).

## 23.2 metis

Description: Metis is an external partitionning library. It is a general algorithm that will generate a partition of the domain.

See also: `partitionneur_deriv` ([23](#))

Usage:

**metis** 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).

### 23.3 partition

Synonymous: **decouper**

Description: This algorithm re-use the partition of the domain named DOMAINE\_NAME. It is useful to partition for example a post processing domain. The partition should match with the calculation domain.

See also: `partitionneur_deriv` (23)

Usage:

**partition** 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).

### 23.4 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` (23)

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).

## 23.5 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` ([23](#))

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).

## 24 precondition\_base

Description: Basic class for preconditioning.

See also: `objet_u` ([33](#)) `ssor` ([24.3](#)) `ssor_bloc` ([24.4](#)) `precondsolv` ([24.2](#)) `precond_local` ([24.1](#))

Usage:

### 24.1 precondition\_local

Description: This keyword can be used with the conjugate gradient (GCP) to choose a local preconditionment for parallel calculation (ie: Cholesky, SSOR,...).

See also: `precond_base` ([24](#))

Usage:

```
precond_local solveur  
where
```

- **solveur** *solveur\_sys\_base* ([9.12](#)): Solver type.

### 24.2 precondsolv

Description: `not_set`

See also: `precond_base` ([24](#))

Usage:

**precondsolv** **solveur**

where

- **solveur** *solveur\_sys\_base* (9.12): Solver type.

### 24.3 ssor

Description: Symmetric successive over-relaxation algorithm.

See also: *precond\_base* (24)

Usage:

**ssor** obj Lire obj {

**omega** *float*

}

where

- **omega** *float*: Over-relaxation facteur (between 1 and 2, optimal value around 1.5-1.6).

### 24.4 ssor\_bloc

Description: not\_set

See also: *precond\_base* (24)

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* (24)
- **alpha\_1** *float*
- **precond1** *precond\_base* (24)
- **alpha\_a** *float*
- **preconda** *precond\_base* (24)

## 25 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 \(33\)](#) [scheme\\_euler\\_explicit \(25.3\)](#) [schema\\_predictor\\_corrector \(25.16\)](#) [Sch\\_CN\\_iteratif \(25.2\)](#) [runge\\_kutta\\_ordre\\_3 \(25.5\)](#) [runge\\_kutta\\_ordre\\_4\\_d3p \(25.6\)](#) [leap\\_frog \(25.4\)](#) [runge\\_kutta\\_rationnel\\_ordre\\_2 \(25.7\)](#) [schema\\_implicite\\_base \(25.15\)](#) [schema\\_adams\\_bashforth\\_order\\_2 \(25.8\)](#) [schema\\_adams\\_bashforth\\_order\\_3 \(25.9\)](#)

Usage:

**schema\_temps\_base** obj Lire obj {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicite int into [0, 1]]
[ niter_max_diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
[ no_conv_subiteration_diffusion_implicite int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

where

- **tinit** *float*: 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).
- **facsec** *float*: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int*: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** *float*: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.



- **dt\_start** *dt\_start* (9.5): **dt\_min** : the first iteration is based on **dt\_min**  
**dt\_start dt\_calc** : the time step at first iteration is calculated in agreement with CFL condition.  
**dt\_start dt\_fixe** value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on **dt\_calc**.
- **seuil\_statio** *float*: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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 into [0, 1]*
- **diffusion\_implicit** *int into [0, 1]*: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, **vrel** should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt\_convection$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a **facsec** that is too large. Start with a **facsec** of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt\_max$ .
- **niter\_max\_diffusion\_implicit** *int*: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float*: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]*: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int*: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]*
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]*
- **periode\_sauvegarde\_securite\_en\_heures** *int*: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** : To disable the check of the available amount of disk space during the calculation.

## 25.1 Sch\_CN\_EX\_iteratif

Description: This keyword also describes a Crank-Nicholson method of second order accuracy but here, for scalars, because of instabilities encountered when  $dt > dt\_CFL$ , the Crank Nicholson scheme is not applied to scalar quantities. Scalars are treated according to Euler-Explicite scheme at the end of the CN treatment for velocity flow fields (by doing  $p$  Euler explicite under-iterations at  $dt \leq dt\_CFL$ ). Parameters are the same (but default values may change) compare to the **Sch\_CN\_iterative** scheme plus a relaxation keyword: **niter\_min** (2 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** (25.2)

Usage:

**Sch\_CN\_EX\_iteratif** obj Lire obj {

```
[ omega float]
[ niter_min int]
[ niter_max int]
[ niter_avg int]
```

```

[ facsec_max float]
[ seuil float]
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicite int into [0, 1]]
[ niter_max_diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
[ no_conv_subiteration_diffusion_implicite int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
where

```

- **omega** *float*: relaxation factor (0.1 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).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.

- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: **dt\_min** : the first iteration is based on dt\_min  
**dt\_start dt\_calc** : the time step at first iteration is calculated in agreement with CFL condition.  
**dt\_start dt\_fixe** value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/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 into [0, 1]* for inheritance
- **diffusion\_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec\*dt\_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec\*dt\_max.
- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.2 Sch\_CN\_iteratif

Description: The Crank-Nicholson method of second order accuracy. A mid-point rule formulation is used (Euler-centered scheme). The basic scheme is:

$$u(t + 1) = u(t) + du/dt(t + 1/2) * dt$$

The estimation of the time derivative du/dt at the level (t+1/2) is obtained either by iterative process. The time derivative du/dt at the level (t+1/2) is calculated iteratively with a simple under-relaxations method. Since the method is implicit, neither the cfl nor the fourier stability criteria must be respected. The time step is calculated in a way that the iterative procedure converges with the less iterations as possible.

Remark : for stationary or RANS calculations, no limitation can be given for time step through high value of facsec\_max parameter (for instance : facsec\_max 1000). In counterpart, for LES calculations, high values of facsec\_max may engender numerical instabilities.

See also: schema\_temps\_base (25) Sch\_CN\_EX\_iteratif (25.1)

Usage:

**Sch\_CN\_iteratif** obj Lire obj {

```

    [ niter_min  int]
    [ niter_max  int]
    [ niter_avg  int]
    [ facsec_max float]
    [ seuil      float]
    [ tinit      float]
    [ tmax       float]
    [ tcpumax   float]
    [ dt_min    float]
    [ dt_max    float]
    [ facsec    float]
    [ nb_pas_dt_max  int]
    [ dt_sauv   float]
    [ dt_impr   float]
    [ dt_start  dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille  int into [0, 1]]
    [ diffusion_implicit  int into [0, 1]]
    [ niter_max_diffusion_implicit  int]
    [ seuil_diffusion_implicit  float]
    [ impr_diffusion_implicit  int into [0, 1]]
    [ precision_impr  int]
    [ no_error_if_not_converged_diffusion_implicit  int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicit  int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures  int]
    [ no_check_disk_space ]

```

}

where

- **niter\_min** *int*: minimal number of p-iterations to satisfy convergence criteria (2 by default)
- **niter\_max** *int*: number of maximum p-iterations allowed to satisfy convergence criteria (6 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).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.

Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth\_order\_3

- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: dt\_min : the first iteration is based on dt\_min  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/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 into [0, 1]* for inheritance
- **diffusion\_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec\*dt\_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec\*dt\_max.
- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

### 25.3 scheme\_euler\_explicit

Synonymous: **schema\_euler\_explicite**

Description: This is the Euler explicite scheme.

See also: **schema\_temps\_base** (25)

Usage:

```
scheme_euler_explicit obj Lire obj {  
    [ tinit float]
```

```

[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
where

```

- **tinit** float for inheritance: Value of initial calculation time (0 by default).
- **tmax** float for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** float for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** float for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** float for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** float for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3
- **nb\_pas\_dt\_max** int for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** float for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** dt\_start (9.5) for inheritance: dt\_min : the first iteration is based on dt\_min  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **seuil\_statio** float for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/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 into [0, 1] for inheritance
- **diffusion\_implicit** int into [0, 1] for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec\*dt\_convection). Thus, in some circumstances, an

important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt\_max$ .

- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.4 leap\_frog

Description: This is the leap-frog scheme.

See also: [schema\\_temps\\_base \(25\)](#)

Usage:

```
leap_frog obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicit int into [0, 1]]
    [ niter_max_diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicit int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: dt\_min : the first iteration is based on dt\_min  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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 into [0, 1]* for inheritance
- **diffusion\_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.



## 25.5 runge\_kutta\_ordre\_3

Description: This is the Runge-Kutta scheme of third order.

See also: `schema_temps_base` (25)

Usage:

```
runge_kutta_ordre_3 obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max float]  
    [ facsec float]  
    [ nb_pas_dt_max int]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ dt_start dt_start]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int into [0, 1]]  
    [ diffusion_implicite int into [0, 1]]  
    [ niter_max_diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int into [0, 1]]  
    [ precision_impr int]  
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]  
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
}
```

where

- **tinit** float for inheritance: Value of initial calculation time (0 by default).
- **tmax** float for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** float for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** float for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** float for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** float for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example `Schema_Adams_Bashforth_order_3`
- **nb\_pas\_dt\_max** int for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** float for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the `.out` file.
- **dt\_start** dt\_start (9.5) for inheritance: **dt\_min** : the first iteration is based on **dt\_min**  
**dt\_start dt\_calc** : the time step at first iteration is calculated in agreement with CFL condition.  
**dt\_start dt\_fixe** value : the first time step is fixed by the user (recommended when restarting calcula-

tion with Crank Nicholson temporal scheme to ensure continuity).

By default, the first iteration is based on `dt_calc`.

- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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 into [0, 1]* for inheritance
- **diffusion\_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, `vrel` should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt\_convection$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a `facsec` that is too large. Start with a `facsec` of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt\_max$ .
- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into `.out` files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in `.sauv` file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.6 runge\_kutta\_ordre\_4\_d3p

Description: `not_set`

See also: `schema_temps_base` (25)

Usage:

**runge\_kutta\_ordre\_4\_d3p** obj Lire obj {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
```

```

[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]

```

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: dt\_min : the first iteration is based on dt\_min  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* into [0, 1] for inheritance
- **diffusion\_implicit** *int* into [0, 1] for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.

- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.7 runge\_kutta\_rationnel\_ordre\_2

Description: This is the Runge-Kutta rational scheme of second order.

See also: [schema\\_temps\\_base \(25\)](#)

Usage:

```
runge_kutta_rationnel_ordre_2 obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicit int into [0, 1]]
    [ niter_max_diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicit int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.

Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3

- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: dt\_min : the first iteration is based on dt\_min  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/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 into [0, 1]* for inheritance
- **diffusion\_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec\*dt\_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec\*dt\_max.
- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.8 schema\_adams\_bashforth\_order\_2

Description: not\_set

See also: schema\_temps\_base (25)

Usage:

```
schema_adams_bashforth_order_2 obj Lire obj {
    [ tinit float]
```

```

[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
where

```

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: dt\_min : the first iteration is based on dt\_min  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/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* into [0, 1] for inheritance
- **diffusion\_implicit** *int* into [0, 1] for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec\*dt\_convection). Thus, in some circumstances, an

important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt = facsec * dt\_max$ .

- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.9 schema\_adams\_bashforth\_order\_3

Description: not\_set

See also: schema\_temps\_base (25)

Usage:

```
schema_adams_bashforth_order_3 obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicit int into [0, 1]]
    [ niter_max_diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicit int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
```



where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: dt\_min : the first iteration is based on dt\_min  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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 into [0, 1]* for inheritance
- **diffusion\_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.



## 25.10 schema\_adams\_moulton\_order\_2

Description: not\_set

See also: schema\_implicite\_base ([25.15](#))

Usage:

```
schema_adams_moulton_order_2 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]  
    [ facsec float]  
    [ nb_pas_dt_max int]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ dt_start dt_start]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int into [0, 1]]  
    [ diffusion_implicite int into [0, 1]]  
    [ niter_max_diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int into [0, 1]]  
    [ precision_impr int]  
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]  
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
}
```

where

- **facsec\_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec\_max value.

Warning: Some implicit schemes do not permit high facsec\_max, example Schema\_Adams\_Moulton\_order\_3 needs facsec=facsec\_max=1.

Advice:

The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec\_max limit. But the user can also choose to specify a constant facsec (facsec\_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:

-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30

-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100

-Thermohydraulic with natural convection, facsec around 300

-Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial facsec with a facsec\_max limit higher.

- **max\_iter\_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).

- **solueur** *solueur\_implicite\_base* (26) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solueur* is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, PISO (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.  
Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then PISO, and at least Simpler. Because the two first give a fastest convergence (several times) than PISO and the Simpler has not been validated. It seems also than Implicite and PISO schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to PISO or Implicite scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: *dt\_min* : the first iteration is based on *dt\_min*  
*dt\_start dt\_calc* : the time step at first iteration is calculated in agreement with CFL condition.  
*dt\_start dt\_fixe* value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on *dt\_calc*.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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 into [0, 1]* for inheritance
- **diffusion\_implicite** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, *vrel* should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **niter\_max\_diffusion\_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.

- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.11 schema\_adams\_moulton\_order\_3

Description: not\_set

See also: [schema\\_implicit\\_base \(25.15\)](#)

Usage:

```

schema_adams_moulton_order_3 obj Lire obj {
    [ facsec_max float]
    [ max_iter_implicit int]
    solveur solveur_implicit_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicit int into [0, 1]]
    [ niter_max_diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicit int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
where

```

- **facsec\_max** *float*: 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* (26) 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).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the *facsec* to 0.5.  
Warning: Some schemes needs a *facsec* lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: *dt\_min* : the first iteration is based on *dt\_min*  
*dt\_start dt\_calc* : the time step at first iteration is calculated in agreement with CFL condition.  
*dt\_start dt\_fixe* value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on *dt\_calc*.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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* into [0, 1] for inheritance

- **diffusion\_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, *vrel* should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt\_convection$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a *facsec* that is too large. Start with a *facsec* of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt\_max$ .
- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.12 schema\_backward\_differentiation\_order\_2

Description: not\_set

See also: `schema_implicit_base` ([25.15](#))

Usage:

`schema_backward_differentiation_order_2` obj Lire obj {

```
[ facsec_max float]
[ max_iter_implicit int]
solveur solveur_implicit_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
```

```

[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
where

```

- **facsec\_max** *float*: 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* (26) 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).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.

- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: **dt\_min** : the first iteration is based on **dt\_min**  
**dt\_start dt\_calc** : the time step at first iteration is calculated in agreement with CFL condition.  
**dt\_start dt\_fixe** value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on **dt\_calc**.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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 into [0, 1]* for inheritance
- **diffusion\_implicite** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, **vrel** should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt\_convection$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a **facsec** that is too large. Start with a **facsec** of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt\_max$ .
- **niter\_max\_diffusion\_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicite** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicite** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicite** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.13 schema\_backward\_differentiation\_order\_3

Description: not\_set

See also: [schema\\_implicite\\_base](#) (25.15)

Usage:

**schema\_backward\_differentiation\_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]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicite int into [0, 1]]
[ niter_max_diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
[ no_conv_subiteration_diffusion_implicite int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
where

```

- **facsec\_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by `facsec` keyword is changed during the calculation with the implicit scheme but it couldn't be higher than `facsec_max` value.

Warning: Some implicit schemes do not permit high `facsec_max`, example `Schema_Adams_Moulton_order_3` needs `facsec=facsec_max=1`.

Advice:

The calculation may start with a `facsec` specified by the user and increased by the algorithm up to the `facsec_max` limit. But the user can also choose to specify a constant `facsec` (`facsec_max` will be set to `facsec` value then). Faster convergence has been seen and depends on the kind of calculation:

- Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value `beta` low), `facsec` between 20-30
- Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value `beta` high), `facsec` between 90-100
- Thermohydraulic with natural convection, `facsec` around 300
- Conduction only, `facsec` can be set to a very high value (1e8) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial `facsec` with a `facsec_max` limit higher.

- **max\_iter\_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur\_implicite\_base* (26) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. `solveur` is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are `Simple` (SIMPLE type algorithm), `Simpler` (SIMPLER type algorithm) for incompressible systems, `Piso` (Pressure Implicit with Split Operator), and `Implicite` (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.  
Advice: Since the 1.6.0 version, we recommend to use first the `Implicite` or `Simple`, then `Piso`, and at least `Simpler`. Because the two first give a fastest convergence (several times) than `Piso` and the `Simpler` has not been validated. It seems also than `Implicite` and `Piso` schemes give better results than the `Simple` scheme when the flow is not fully stationary. Thus, if the solution obtained with `Simple` is not stationary, it is recommended to switch to `Piso` or `Implicite` scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).



- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: dt\_min : the first iteration is based on dt\_min  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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 into [0, 1]* for inheritance
- **diffusion\_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergence criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.14 `scheme_euler_implicit`

Synonymous: `schema_euler_implicite`

Description: This is the Euler implicite scheme.

See also: `schema_implicite_base` ([25.15](#))

Usage:

```
scheme_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]  
    [ facsec float]  
    [ nb_pas_dt_max int]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ dt_start dt_start]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int into [0, 1]]  
    [ diffusion_implicite int into [0, 1]]  
    [ niter_max_diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int into [0, 1]]  
    [ precision_impr int]  
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]  
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
}
```

where

- **facsec\_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by `facsec` keyword is changed during the calculation with the implicit scheme but it couldn't be higher than `facsec_max` value.

Warning: Some implicit schemes do not permit high `facsec_max`, example `Schema_Adams_Moulton_order_3` needs `facsec=facsec_max=1`.

Advice:

The calculation may start with a `facsec` specified by the user and increased by the algorithm up to the `facsec_max` limit. But the user can also choose to specify a constant `facsec` (`facsec_max` will be set to `facsec` value then). Faster convergence has been seen and depends on the kind of calculation:

-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value `beta` low), `facsec` between 20-30

-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value `beta` high), `facsec` between 90-100

-Thermohydraulic with natural convection, `facsec` around 300

-Conduction only, `facsec` can be set to a very high value ( $1e8$ ) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial `facsec` with a `facsec_max` limit higher.

- **max\_iter\_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur\_implicite\_base* (26) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solveur* is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, PISO (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.  
Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then PISO, and at least Simpler. Because the two first give a fastest convergence (several times) than PISO and the Simpler has not been validated. It seems also than Implicite and PISO schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to PISO or Implicite scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: *dt\_min* : the first iteration is based on *dt\_min*  
*dt\_start dt\_calc* : the time step at first iteration is calculated in agreement with CFL condition.  
*dt\_start dt\_fixe* value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on *dt\_calc*.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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 into [0, 1]* for inheritance
- **diffusion\_implicite** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, *vrel* should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **niter\_max\_diffusion\_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.

- **seuil\_diffusion\_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicite** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicite** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicite** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.15 schema\_implicite\_base

Description: Basic class for implicite time scheme.

See also: [schema\\_temps\\_base \(25\)](#) [scheme\\_euler\\_implicit \(25.14\)](#) [schema\\_adams\\_moulton\\_order\\_2 \(25.10\)](#) [schema\\_adams\\_moulton\\_order\\_3 \(25.11\)](#) [schema\\_backward\\_differentiation\\_order\\_2 \(25.12\)](#) [schema\\_backward\\_differentiation\\_order\\_3 \(25.13\)](#)

Usage:

```

schema_implicite_base obj Lire obj {
    [ max_iter_implicite int ]
    solveur solveur_implicite_base
    [ tinit float ]
    [ tmax float ]
    [ tcpumax float ]
    [ dt_min float ]
    [ dt_max float ]
    [ facsec float ]
    [ nb_pas_dt_max int ]
    [ dt_sauv float ]
    [ dt_impr float ]
    [ dt_start dt_start ]
    [ seuil_statio float ]
    [ seuil_statio_relatif_deconseille int into [0, 1] ]
    [ diffusion_implicite int into [0, 1] ]
    [ niter_max_diffusion_implicite int ]
    [ seuil_diffusion_implicite float ]
    [ impr_diffusion_implicite int into [0, 1] ]
    [ precision_impr int ]
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1] ]
    [ no_conv_subiteration_diffusion_implicite int into [0, 1] ]
    [ periode_sauvegarde_securite_en_heures int ]
    [ no_check_disk_space ]
}
where
```

- **max\_iter\_implicite** *int*: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur\_implicite\_base* (26): 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).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema\_Adams\_Bashforth\_order\_3
- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: dt\_min : the first iteration is based on dt\_min  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives  $dG_i/dt$  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 into [0, 1]* for inheritance
- **diffusion\_implicite** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step ( $dt=facsec*dt_{convection}$ ). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore  $dt=facsec*dt_{max}$ .
- **niter\_max\_diffusion\_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicite** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.

- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicite** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicite** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 25.16 schema\_predictor\_corrector

Description: This is the predictor-corrector scheme (second order). It is more accurate and economic than MacCormack scheme. It gives best results with a second ordre convective scheme like quick, centre (VDF).

See also: [schema\\_temps\\_base \(25\)](#)

Usage:

```
schema_predictor_corrector obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicite int into [0, 1]]
    [ niter_max_diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt\_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt\_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.  
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth\_order\_3



- **nb\_pas\_dt\_max** *int* for inheritance: Maximum number of calculation time steps (1e9 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.
- **dt\_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt\_start** *dt\_start* (9.5) for inheritance: dt\_min : the first iteration is based on dt\_min  
dt\_start dt\_calc : the time step at first iteration is calculated in agreement with CFL condition.  
dt\_start dt\_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).  
By default, the first iteration is based on dt\_calc.
- **seuil\_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/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 into [0, 1]* for inheritance
- **diffusion\_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec\*dt\_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec\*dt\_max.
- **niter\_max\_diffusion\_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil\_diffusion\_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr\_diffusion\_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision\_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no\_error\_if\_not\_converged\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **no\_conv\_subiteration\_diffusion\_implicit** *int into [0, 1]* for inheritance
- **periode\_sauvegarde\_securite\_en\_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no\_check\_disk\_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

## 26 solveur\_implicit\_base

Description: Class for solver in the situation where the time scheme is the implicit scheme. Solver allows equation diffusion and convection operators to be set as implicit terms.

See also: objet\_u (33) solveur\_lineaire\_std (26.5) simplifier (26.4)

Usage:

## 26.1 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 ([26.2](#))

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* ([9.12](#)) 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.

## 26.2 piso

Description: Piso (Pressure Implicit with Split Operator) - method to solve N\_S.

See also: simplr ([26.4](#)) implicite ([26.1](#)) simple ([26.3](#))

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* (9.12) 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.

## 26.3 simple

Description: SIMPLE type algorithm

See also: `piso` (26.2)

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\_implicit** *float* for inheritance: Convergence criteria.
- **nb\_corrections\_max** *int* for inheritance: Maximum number of corrections performed by the PISO  
algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections  
then nb\_corrections\_max if the accuracy of the projection is sufficient. (By default nb\_corrections-  
\_max is set to 21).
- **seuil\_convergence\_solveur** *float* for inheritance: value of the convergence criteria for the resolution  
of the implicit system build by solving several times per time step the Navier-Stokes equation and the  
scalar equations if any. This value MUST be used when a coupling between problems is considered  
(should be set to a value typically of 0.1 or 0.01).
- **seuil\_generation\_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as  
the convergence threshold (implicit linear system  $Ax=B$  will be solved if residual error  $\|Ax-B\|$  is  
lesser than vrel).
- **seuil\_verification\_solveur** *float* for inheritance: Option to check if residual error  $\|Ax-B\|$  is lesser  
than vrel after the implicit linear system  $Ax=B$  has been solved.
- **seuil\_test\_preliminaire\_solveur** *float* for inheritance: Option to decide if the implicit linear system  
 $Ax=B$  should be solved by checking if the residual error  $\|Ax-B\|$  is bigger than vrel.
- **solveur** *solveur\_sys\_base* (9.12) 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.

## 26.4 simplr

Description: Simpler method for incompressible systems.

See also: solveur\_implicit\_base (26) piso (26.2)

Usage:

**simplr** obj Lire obj {

```

seuil_convergence_implicit 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* (9.12): 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.

## 26.5 solveur\_lineaire\_std

Description: not\_set

See also: solveur\_implicit\_base (26)

Usage:

**solveur\_lineaire\_std** obj Lire obj {

    [ **solveur** *solveur\_sys\_base*]

}

where

- **solveur** *solveur\_sys\_base* (9.12)

## 27 source\_base

Description: Basic class of source terms introduced in the equation.

See also: objet\_u (33) source\_generique (27.19) boussinesq\_temperature (27.4) boussinesq\_concentration (27.3) dirac (27.8) puissance\_thermique (27.17) source\_qdm\_lambdaup (27.21) source\_th\_tdivu (27.25) source\_robin (27.22) source\_robin\_scalaire (27.23) canal\_perio (27.5) source\_constituant (27.18) source\_transport\_k\_eps (27.26) acceleration (27.2) coriolis (27.6) source\_qdm (27.20) perte\_charge\_singuliere (27.16) perte\_charge\_directionnelle (27.12) perte\_charge\_isotrope (27.13) perte\_charge\_anisotrope (27.10) perte\_charge\_circulaire (27.11) darcy (27.7) forchheimer (27.9) perte\_charge\_reguliere (27.14)

Usage:

## 27.1 Source\_Transport\_K\_Eps\_anisotherme

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transportation equation. By default, these constants are set to: C1\_eps=1.44 C2\_eps=1.92 C3\_eps=1.0

See also: `source_transport_k_eps` ([27.26](#))

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.

## 27.2 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` ([27](#))

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 ([15.1](#)): 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 ([15.1](#)): 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 ([15.1](#)): 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 ([15.1](#)): 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 ([15.1](#)): 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  $\mathbf{0}'=(0,0,0)$ ). 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.

### 27.3 boussinesq\_concentration

Description: Class to describe a source term that couples the movement quantity equation and constituent transportation equation with the Boussinesq hypothesis.

See also: [source\\_base \(27\)](#)

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.

### 27.4 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 \(27\)](#)

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.

### 27.5 canal\_perio

Description: Momentum source term to maintain flow rate. The expression of the source term is:

$$S(t) = (2*(Q(0) - Q(t)) - (Q(0) - Q(t-dt)))/(coeff*dt*area)$$

Where:

coeff=damping coefficient  
area=area of the periodic boundary  
 $Q(t)$ =flow rate at time  $t$   
dt=time step

Three files will be created during calculation on a datafile named DataFile.data. The first file contains the flow rate evolution. The second file is useful for restarting a calculation with the flow rate of the previous stopped calculation, and the last one contains the pressure gradient evolution:

-DataFile\_Channel\_Flow\_Rate\_ProblemName\_BoundaryName  
-DataFile\_Channel\_Flow\_Rate\_repr\_ProblemName\_BoundaryName  
-DataFile\_Pressure\_Gradient\_ProblemName\_BoundaryName

See also: [source\\_base \(27\)](#)

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.

## 27.6 coriolis

Description: Keyword for a Coriolis term in hydraulic equation. Warning: Only available in VDF.

See also: [source\\_base \(27\)](#)

Usage:

**coriolis** **omega**

where

- **omega** *str*: Value of omega.

## 27.7 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 \(27\)](#)

Usage:

**darcy bloc**

where

- **bloc** *bloc\_lecture* (3.38): Description.

## 27.8 dirac

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: *source\_base* (27)

Usage:

**dirac position ch**

where

- **position** *n x1 x2 ... xn*
- **ch** *champ\_base* (15.1): 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.

## 27.9 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* (27)

Usage:

**forchheimer bloc**

where

- **bloc** *bloc\_lecture* (3.38): Description.

## 27.10 perte\_charge\_anisotrope

Description: Anisotropic pressure loss.

See also: *source\_base* (27)

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* (15.2): Hydraulic diameter value.
- **direction** *champ\_don\_base* (15.2): Field which indicates the direction of the pressure loss.
- **sous\_zone** *str*: Optional sub-area where pressure loss applies.

## 27.11 perte\_charge\_circulaire

Description: New pressure loss.

See also: [source\\_base](#) (27)

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* (15.2): Hydraulic diameter value.
- **diam\_hydr\_ortho** *champ\_don\_base* (15.2): Transverse hydraulic diameter value.
- **direction** *champ\_don\_base* (15.2): Field which indicates the direction of the pressure loss.
- **sous\_zone** *str*: Optional sub-area where pressure loss applies.

## 27.12 perte\_charge\_directionnelle

Description: Directional pressure loss.

See also: [source\\_base](#) (27)

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* (15.2): Hydraulic diameter value.
- **direction** *champ\_don\_base* (15.2): Field which indicates the direction of the pressure loss.
- **sous\_zone** *str*: Optional sub-area where pressure loss applies.



### 27.13 perte\_charge\_isotrope

Description: Isotropic pressure loss.

See also: [source\\_base \(27\)](#)

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* ([15.2](#)): Hydraulic diameter value.
- **sous\_zone** *str*: Optional sub-area where pressure loss applies.

### 27.14 perte\_charge\_reguliere

Description: Source term modelling the presence of a bundle of tubes in a flow.

See also: [source\\_base \(27\)](#)

Usage:

**perte\_charge\_reguliere** **spec** **zone\_name**

where

- **spec** *spec\_pdc\_base* ([27.15](#)): 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.

### 27.15 spec\_pdc\_base

Description: Class to read the source term modelling the presence of a bundle of tubes in a flow.  $C_f = A / Re \cdot B$ .

See also: [objet\\_lecture \(32\)](#) longitudinale ([27.15.1](#)) transversale ([27.15.2](#))

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.

### 27.15.1 longitudinale

Description: Class to define the pressure loss in the direction of the tube bundle.

See also: `spec_pdcr_base` ([27.15](#))

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.

### 27.15.2 transversale

Description: Class to define the pressure loss in the direction perpendicular to the tube bundle.

See also: `spec_pdcr_base` ([27.15](#))

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.

## 27.16 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` ([27](#))

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* (3.38): Surface definition block : In VDF, the surface area definition syntax is identical to that used to define sides (edges) in the Block, for example { X = x0 y0 <= Y <= y1 } for a line perpendicular to the Ox axis in a two-dimensional domain, or { Y = y0 x0 <= X <= x1 z0 <= Z <= z1 } for a surface perpendicular to the Oy axis in a 3D domain.  
example : sources { Perte\_Charge\_Singuliere KX 0.5 { X = 1. 0. <= Y <= 1. } }

VEF : the surface area definition syntax relies on sub-areas definition (see 4.3.22). First value (X=0.35 in the example below, in regard to KX keyword) allows to determine the faces of elements in sub-area for which the pressure loss is applied.

example : sources { Perte\_Charge\_Singuliere KX 0.5 { 0.35 sous\_zone\_toto } }

Observations :

- If the surface area is not included in the calculation domain or if (in VDF) it is not perpendicular to the space direction in accordance with which the pressure loss is being calculated, Trio-U exists in error.
- The surface area may be diminished at only one side if a sudden shrinking or widening occurs.

## 27.17 puissance\_thermique

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: source\_base (27)

Usage:

**puissance\_thermique ch**

where

- **ch** *champ\_base* (15.1): 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> in 3D. It is a power per volume unit (in a porous media, it is a power per fluid volume unit).

## 27.18 source\_constituant

Description: Keyword to specify source rates, in [[C]/s], for each one of the nb constituents. [C] is the concentration unit.

See also: source\_base (27)

Usage:

**source\_constituant ch**

where

- **ch** *champ\_base* (15.1): Field type.

## 27.19 source\_generique

Description: to define a source term depending on some discrete fields of the problem and (or) analytic expression. It is expressed by the way of a generic field usually used for post-processing.

See also: source\_base (27)

Usage:

**source\_generique champ**

where

- **champ** *champ\_generique\_base* (7): the source field

## 27.20 source\_qdm

Description: Momentum source term in the Navier Stokes equation.

See also: *source\_base* (27)

Usage:

**source\_qdm ch**

where

- **ch** *champ\_base* (15.1): Field type.

## 27.21 source\_qdm\_lambdaup

Description: This source term is a dissipative term which is intended to minimise the energy associated to non-conformscales  $u'$  (responsible for spurious oscillations in some cases). The equation for these scales can be seen as:  $du'/dt = -\lambda u' + \text{grad } P'$  where  $-\lambda u'$  represents the dissipative term, with  $\lambda = a/\Delta t$ . For Crank-Nicholson temporal scheme, recommended value for  $a$  is 2.

Remark : This method requires to define a filtering operator.

See also: *source\_base* (27)

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  $\bar{u}_{\text{umprim\_cible}}$

## 27.22 source\_robin

Description: This source term should be used when a *Paroi\_decalee\_Robin* boundary condition is set in a hydraulic equation. The source term will be applied on the  $N$  specified boundaries. To post-process the values of  $\tau_w$ ,  $u_{\tau}$  and  $\text{Reynolds}_{\tau}$  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* (27)

Usage:

**source\_robin bords**

where

- **bords** *vect\_nom* (3.101)

### 27.23 source\_robin\_scalaire

Description: This source term should be used when a Paroi\_decalee\_Robin boundary condition is set in an energy equation. The source term will be applied on the N specified boundaries. The values temp\_wall\_valueI are the temperature specified on the 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 (27)

Usage:

**source\_robin\_scalaire bords**

where

- **bords** *listdeuxmots\_sacc* (27.24)

### 27.24 listdeuxmots\_sacc

Description: List of groups of two words (without accodances).

See also: listobj (31.3)

Usage:

n object1 object2 ....

list of *deuxmots* (5.20)

### 27.25 source\_th\_tdivu

Description: This term source is dedicated for any scalar (called T) transportation. Coupled with upwind (amont) or muscl scheme, this term gives for final expression of convection :  $\text{div}(\mathbf{U} \cdot \mathbf{T}) - \mathbf{T} \cdot \text{div}(\mathbf{U}) = \mathbf{U} \cdot \text{grad}(\mathbf{T})$ . This ensures, in incompressible flow when divergence free is badly resolved, to stay in a better way in the physical boundaries.

Warning: Only available in VEF discretization.

See also: source\_base (27)

Usage:

**source\_th\_tdivu**

### 27.26 source\_transport\_k\_eps

Description: Keyword to alter the source term constants in the standard k-eps model epsilon transportation equation. By default, these constants are set to: C1\_eps=1.44 C2\_eps=1.92

See also: source\_base (27) Source\_Transport\_K\_Eps\_anisotherme (27.1) source\_transport\_k\_eps\_aniso\_concen (27.27) source\_transport\_k\_eps\_aniso\_therm\_concen (27.28)

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.

## 27.27 source\_transport\_k\_eps\_aniso\_concen

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transportation equation. By default, these constants are set to: C1\_eps=1.44 C2\_eps=1.92 C3\_eps=1.0

See also: [source\\_transport\\_k\\_eps \(27.26\)](#)

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.

## 27.28 source\_transport\_k\_eps\_aniso\_therm\_concen

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transportation equation. By default, these constants are set to: C1\_eps=1.44 C2\_eps=1.92 C3\_eps=1.0

See also: [source\\_transport\\_k\\_eps \(27.26\)](#)

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.

## 28 sous\_zone

Description: It is an object type describing a domain sub-set.

A Sous\_Zone (Sub-area) type object must be associated with a Domaine type object. The Lire (Read) interpreter is used to define the items comprising the sub-area.

Caution: The Domain type object `nom_domaine` must have been meshed (and triangulated or tetrahedralised in VEF) prior to carrying out the Associer (Associate) `nom_sous_zone nom_domaine` instruction; this instruction must always be preceded by the read instruction.

See also: `objet_u` (33)

Usage:

```
sous_zone obj Lire obj {
    [ restriction str]
    [ rectangle bloc_origine_cotes]
    [ segment bloc_origine_cotes]
    [ boite bloc_origine_cotes]
    [ liste n n1 n2 ... nn]
    [ fichier str]
    [ intervalle deuxentiers]
    [ polynomes bloc_lecture]
    [ couronne bloc_couronne]
    [ tube bloc_tube]
    [ fonction_sous_zone str]
    [ union str]
}
```

where

- **restriction** *str*: The elements of the sub-area `nom_sous_zone` must be included into the other sub-area named `nom_sous_zone2`. This keyword should be used first in the Lire keyword.
- **rectangle** *bloc\_origine\_cotes* (28.1): The sub-area will include all the domain elements whose centre of gravity is within the Rectangle (in dimension 2).
- **segment** *bloc\_origine\_cotes* (28.1)
- **boite** *bloc\_origine\_cotes* (28.1): 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* (28.2): The sub-area will include domain items whose number is between *n1* and *n2* (where  $n1 \leq n2$ ).
- **polynomes** *bloc\_lecture* (3.38): A REPENDRE
- **couronne** *bloc\_couronne* (28.3): In 2D case, to create a couronne.
- **tube** *bloc\_tube* (28.4): In 3D case, to create a tube.
- **fonction\_sous\_zone** *str*: Keyword to build a sub-area with the the elements included into the area defined by `fonction>0`.
- **union** *str*: The elements of the sub-area `nom_sous_zone3` will be added to the sub-area `nom_sous_zone`. This keyword should be used last in the Lire keyword.

## 28.1 bloc\_origine\_cotes

Description: Class to create a rectangle (or a box).

See also: `objet_lecture` (32)

Usage:

```
name origin name2 cotes
where
```

- **name** *str into ['Origine']*: Keyword to define the origin of the rectangle (or the box).

- **origin** *x1 x2 (x3)*: Co-ordinates of the origin of the rectangle (or the box).
- **name2** *str into ['Cotes']*: Keyword to define the length along the axes.
- **cotes** *x1 x2 (x3)*: Length along the axes.

## 28.2 deuxentiers

Description: Two integers.

See also: [objet\\_lecture \(32\)](#)

Usage:

**int1 int2**

where

- **int1** *int*: First integer.
- **int2** *int*: Second integer.

## 28.3 bloc\_couronne

Description: Class to create a couronne (2D).

See also: [objet\\_lecture \(32\)](#)

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.

## 28.4 bloc\_tube

Description: Class to create a tube (3D).

See also: [objet\\_lecture \(32\)](#)

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.



## 29 turbulence\_paro\_base

Description: Basic class for wall laws for NAVIER STOKES equations.

See also: objet\_u (33) loi\_standard\_hydr\_old (29.3) loi\_standard\_hydr (29.2) paroi\_tble (29.5) negligible (29.4) utau\_imp (29.9)

Usage:

### 29.1 loi\_expert\_hydr

Description: This keyword is similar to the previous keyword Loi\_standard\_hydr but has several additional options into brackets.

See also: loi\_standard\_hydr (29.2)

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)

### 29.2 loi\_standard\_hydr

Description: Keyword for the logarithmic wall law for a hydraulic problem. Loi\_standard\_hydr refers to first cell rank eddy-viscosity defined from continuous analytical functions, whereas Loi\_standard\_hydr\_3couches from functions separately defined for each sub-layer

See also: turbulence\_paro\_base (29) loi\_expert\_hydr (29.1)

Usage:

**loi\_standard\_hydr**

### 29.3 loi\_standard\_hydr\_old

Description: not\_set

See also: turbulence\_paro\_base (29)

Usage:

**loi\_standard\_hydr\_old**

### 29.4 negligible

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 (29)

Usage:

**negligeable**

### 29.5 paroi\_tble

Description: Keyword for the Thin Boundary Layer Equation wall-model (a more complete description of the model can be found into this PDF file). The wall shear stress is evaluated thanks to boundary layer equations applied in a one-dimensional fine grid in the near-wall region.

See also: turbulence\_paro\_base (29)

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** (29.6): 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** (29.7)

- **restart**
- **stationnaire** *entierfloat* (29.8)
- **lambda** *str*
- **mu** *str*
- **sans\_source\_boussinesq**
- **alpha** *float*
- **kappa** *float*

## 29.6 twofloat

Description: two reals.

See also: [objet\\_lecture \(32\)](#)

Usage:

**a b**

where

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

## 29.7 liste\_sonde\_tble

Description: not\_set

See also: [listobj \(31.3\)](#)

Usage:

n object1 object2 ....

list of *sonde\_tble* (29.7.1)

### 29.7.1 sonde\_tble

Description: not\_set

See also: [objet\\_lecture \(32\)](#)

Usage:

**name point**

where

- **name** *str*
- **point** *un\_point* (3.7.3)

## 29.8 entierfloat

Description: An integer and a real.

See also: [objet\\_lecture \(32\)](#)

Usage:

**the\_int the\_float**

where

- **the\_int** *int*: Integer.

- **the\_float** *float*: Real.

## 29.9 utau\_imp

Description: Keyword to impose the friction velocity on the wall with a turbulence model for thermohydraulic problems. There are two possibilities to use this keyword :

- 1 - we can impose directly the value of the friction velocity  $u_{star}$ .
- 2 - we can also give the friction coefficient and hydraulic diameter. So, TRUST determines the friction velocity by :  $u_{star} = U \cdot \sqrt{\lambda_c / 8}$ .

See also: [turbulence\\_paro\\_base \(29\)](#)

Usage:

```
utau_imp obj Lire obj {
    [ u_tau  champ_base]
    [ lambda_c  str]
    [ diam_hydr  champ_base]
}
```

where

- **u\_tau** *champ\_base* (15.1): 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* (15.1): The hydraulic diameter.

## 30 turbulence\_paro\_scalaire\_base

Description: Basic class for wall laws for energy equation.

See also: [objet\\_u \(33\)](#) [loi\\_standard\\_hydr\\_scalaire \(30.4\)](#) [loi\\_analytique\\_scalaire \(30.1\)](#) [paroi\\_tble\\_scal \(30.6\)](#) [loi\\_paro\\_nu\\_impose \(30.3\)](#) [negligeable\\_scalaire \(30.5\)](#)

Usage:

### 30.1 loi\_analytique\_scalaire

Description: not\_set

See also: [turbulence\\_paro\\_scalaire\\_base \(30\)](#)

Usage:

```
loi_analytique_scalaire
```

### 30.2 loi\_expert\_scalaire

Description: Keyword similar to keyword `Loi_standard_hydr_scalaire` but with additional option.

See also: [loi\\_standard\\_hydr\\_scalaire \(30.4\)](#)

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.

### 30.3 loi\_paroι\_nu\_impose

Description: Keyword to impose Nusselt numbers on the wall for the thermohydraulic problems. To use this option, it is necessary to give in the data file the value of the hydraulic diameter and the expression of the Nusselt number.

See also: turbulence\_paroι\_scalaire\_base (30)

Usage:

```
loi_paroι_nu_impose obj Lire obj {
    nusselt str
    diam_hydr champ_base
}
where
```

- **nusselt** *str*: The Nusselt number. This expression can be a function of x, y, z, Re (Reynolds number), Pr (Prandtl number).
- **diam\_hydr** *champ\_base* (15.1): The hydraulic diameter.

### 30.4 loi\_standard\_hydr\_scalaire

Description: Keyword for the law of the wall.

See also: turbulence\_paroι\_scalaire\_base (30) loi\_expert\_scalaire (30.2)

Usage:

```
loi_standard_hydr_scalaire
```

### 30.5 negligeable\_scalaire

Description: Keyword to suppress the calculation of a law of the wall with a turbulence model for thermohydraulic problems. The wall stress is directly calculated with the derivative of the velocity, in the direction perpendicular to the wall.

See also: turbulence\_paroι\_scalaire\_base (30)

Usage:

```
negligeable_scalaire
```

### 30.6 paroi\_tble\_scal

Description: Keyword for the Thin Boundary Layer Equation thermal wall-model.

See also: turbulence\_paro\_scalaire\_base (30)

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* (30.7): 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* (29.7)
- **prandtl** *float*

### 30.7 fourfloat

Description: Four reals.

See also: objet\_lecture (32)

Usage:

```
a b c d  
where
```

- **a** *float*: First real.
- **b** *float*: Second real.
- **c** *float*: Third real.
- **d** *float*: Fourth real.

## 31 listobj\_impl

Description: not\_set

See also: objet\_u (33) listobj (31.3)

Usage:

### 31.1 list\_un\_pb

Description: pour les groupes

See also: listobj (31.3)

Usage:

{ object1 , object2 .... }  
list of *un\_pb* (31.2) separated with ,

### 31.2 un\_pb

Description: pour les groupes

See also: objet\_lecture (32)

Usage:

**mot**

where

- **mot** *str*: la chaîne

### 31.3 listobj

Description: List of objects.

See also: listobj\_impl (31) champs\_a\_post (4.2.18) list\_stat\_post (4.2.21) listpoints (4.2.7) sondes (4.2.3) listchamp\_generique (7.3) list\_nom\_virgule (7.2) definition\_champs (4.2.1) post\_processings (4.3) liste\_post (4.5) liste\_post\_ok (4.4) condlims (5.4) sources (5.5) vect\_nom (3.101) list\_nom (3.86) list\_bord (3.46.4) list\_bloc\_mailler (3.46) list\_un\_pb (31.1) list\_list\_nom (4.8) ecrire\_fichier\_xyz\_valeur\_param (5.6) pp (5.16) listdeuxmots\_sacc (27.24) liste\_sonde\_tble (29.7) listeqn (4.10) list\_info\_med (4.32) listsous\_zone\_valeur (5.9.12) reactions (8.1)

Usage:

## 32 objet\_lecture

Description: Auxiliary class for reading.

See also: objet\_u (33) bloc\_lecture (3.38) deuxmots (5.20) format\_file (4.6) deuxentiers (28.2) floatfloat (5.21) entierfloat (29.8) 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 (3.7.3) sonde (4.2.4) definition\_champ (4.2.2) postraitemement\_base (4.4.2) un\_postraitemement (4.3.1) type\_un\_post (4.5.2) type\_postraitemement\_ft\_lata (4.5.3) un\_postraitemement\_spec (4.5.1) nom\_postraitemement (4.4.1) condinit (5.3.1) condinits (5.3) condlimlu (5.4.1) mailler\_base (3.46.1) bloc\_pave (3.46.3) defbord (3.46.7) bord\_base (3.46.5) parametre\_equation\_base (5.7) un\_pb (31.2) bords\_ecrire (5.6.2) ecrire\_fichier\_xyz\_valeur\_item (5.6.1) convection\_deriv (5.9.1) bloc\_convection (5.9) diffusion\_deriv (5.2.1) op\_implicite (5.2.9) bloc\_diffusion (5.2) traitement\_particulier\_base (5.22.1) traitement\_particulier (5.22) penalisation\_l2\_ftd\_lec (5.16.1) dt\_impr\_ustar\_mean\_only (5.25.1) modele\_turbulence\_hyd\_deriv (5.25) paroi\_ft\_disc\_deriv (32.1) bloc\_sutherland (20.5) form\_a\_nb\_points (5.25.4) modele\_fonction\_bas\_reynolds\_base (5.25.11) fourfloat (30.7) twofloat (29.6) sonde\_tble (29.7.1) remove\_elem\_bloc (3.74) lecture\_bloc\_moment\_base (3.7) bloc\_origine\_cotes (28.1) bloc\_couronne (28.3) bloc\_tube (28.4) bloc\_lecture\_poro (3.57) bloc\_lec\_champ\_init\_canal\_sinal (15.12) fonction\_champ\_reprise (15.8) bloc\_decouper (3.54) troisi (3.32) spec\_pdc\_base (27.15) format\_lata\_to\_med (3.42) info\_med

([4.32.1](#)) `methode_transport_deriv` ([32.2](#)) `bloc_ef` ([5.9.9](#)) `sous_zone_valeur` ([5.9.13](#)) `bloc_diffusion_standard` ([5.2.7](#)) `reaction` ([8.1.1](#))

Usage:

## 32.1 `paroi_ft_disc_deriv`

Description: `not_set`

See also: `objet_lecture` ([32](#)) `symetrie` ([32.1.1](#))

Usage:

**`paroi_ft_disc_deriv`**

### 32.1.1 `symetrie`

Description: Symetrie condition in the case of two-phase flows

See also: `paroi_ft_disc_deriv` ([32.1](#))

Usage:

**`symetrie`**

## 32.2 `methode_transport_deriv`

Description: Basic class for method of transport of interface.

See also: `objet_lecture` ([32](#)) `loi_horaire` ([32.2.1](#))

Usage:

**`methode_transport_deriv`**

### 32.2.1 `loi_horaire`

Description: `not_set`

See also: `methode_transport_deriv` ([32.2](#))

Usage:

**`loi_horaire` `nom_loi`**

where

- **`nom_loi`** *str*

## 33 index



## Index

/\*, 132  
#, 152  
  
, 88, 91, 95, 112  
associer, 5  
champ\_post\_statistiques\_correlation, 58, 135  
champ\_post\_statistiques\_ecart\_type, 57, 136  
champ\_post\_statistiques\_moyenne, 57, 139  
champ\_uniforme, 176  
decouper, 29, 194  
discretiser, 11  
divergence, 136  
ecrire\_fichier, 48  
extraction, 137  
fin, 18  
gradient, 138  
interpolation, 138  
lire, 33  
lire\_fichier, 34  
lire\_fichier\_bin, 34  
lire\_med, 35  
morceau\_equation, 139  
operateur\_eqn, 134  
postraitement, 60  
postraitements, 59  
raffiner\_simplexes, 33  
rectify\_mesh, 36  
reduction\_0d, 140  
refchamp, 141  
resoudre, 40  
schema\_euler\_explicite, 202  
schema\_euler\_implicite, 223  
tparoi\_vef, 142  
transformation, 142  
<=, 24  
=, 24  
a, 238, 239  
amont, 99  
ancien, 94, 101  
antisym, 97  
arrete, 120–128  
avec\_les\_cl, 110, 115, 117, 130  
avec\_sources, 110, 115, 117, 130  
avec\_sources\_et\_operateurs, 110, 115, 117, 130  
b, 238, 239  
binaire, 11, 55, 61, 170  
bords, 92  
C, 190  
C\_ext, 154  
centre, 99  
cf, 238, 239  
  
chakravarthy, 99  
champ\_frontiere, 137  
chsom, 51  
composante, 142, 143  
conservation\_masse, 189  
constant, 189  
coriolis\_seul, 233, 234  
Cotes, 245  
d, 239  
debit\_total, 19, 20  
default, 138, 139  
defaut\_bar, 89, 97  
dir, 245  
distant, 24  
divrhout\_moins\_Tdivrhout, 94, 101  
divuT\_moins\_Tdivu, 94, 101  
dt\_integr, 58  
dt\_post, 55, 56  
edo, 189  
elem, 27, 55, 57, 58, 169  
entrainement\_seul, 233, 234  
faces, 55, 57, 58  
family\_names\_from\_group\_names, 35  
filtrer\_resu, 90, 97  
Fluctu\_Temperature\_ext, 154  
flux\_bords, 139  
Flux\_Chaleur\_Turb\_ext, 154  
fonction, 170  
format\_post\_sup, 20  
formatte, 11, 55, 61, 170  
formule, 142, 143  
grad\_Ubar, 90  
grav, 51  
hauteur, 245  
homogene, 24  
implicite, 90  
integrale\_en\_z, 19, 20  
k, 164  
K\_Eps\_ext, 154  
kx, 239  
ky, 239  
kz, 239  
last\_time, 169  
lata, 20, 21, 31, 32, 50, 60  
lata\_v1, 20, 21, 31, 32, 50, 60  
lata\_v2, 20, 21, 31, 32, 50, 60  
lml, 20, 21, 31, 32, 50, 60  
local, 24  
max, 141  
med, 20, 21, 31, 32, 50, 60

meshtv , 20, 21, 31, 32, 50, 60  
 min , 141  
 minmod , 99  
 moins\_rho\_moyen , 189  
 moyenne , 141  
 moyenne\_ponderee , 141  
 mu0 , 190  
 muscl , 99  
 nb\_pas\_dt\_post , 55, 56  
 no , 132, 138, 139  
 nodes , 51  
 non , 28  
 normalized\_norm\_l2 , 141  
 norme , 142, 143  
 norme\_l2 , 141  
 nu , 90  
 nu\_transp , 90  
 nut , 90  
 nut\_transp , 90  
 Origine , 244, 245  
 oui , 28  
 periode , 51  
 post\_processing , 61  
 postraitement , 61  
 postraitement\_ft\_lata , 61  
 postraitement\_lata , 61  
 produit\_scalaire , 142, 143  
 que\_les\_faces\_des\_elts\_dirichlet , 246  
 re , 245  
 ri , 245  
 sans\_rien , 110, 115, 117, 130  
 scotti , 120–128  
 short\_family\_names , 35  
 Slambda , 190  
 solveur , 90  
 som , 27, 51, 55, 57, 58, 169  
 somme , 141  
 somme\_ponderee , 141  
 stabilite , 139  
 standard , 189  
 superbee , 99  
 T0 , 190  
 T\_ext , 154  
 terme\_complet , 233, 234  
 toutes\_les\_faces\_accrochees , 246  
 trace , 137  
 transportant\_bar , 97  
 transporte\_bar , 97  
 use\_existing\_domain , 169  
 V2\_ext , 154  
 valeur\_normale , 182  
 vanalbada , 99  
 vanleer , 99  
 vecteur , 142, 143

vef , 35  
 vitesse\_paroie , 164  
 vitesse\_tangentielle , 184  
 volume , 120–128  
 volume\_sans\_lissage , 120–128  
 X , 24, 39, 245  
 x , 239  
 xyz , 61, 170  
 Y , 24, 39, 245  
 y , 239  
 yes , 132, 138, 139  
 Z , 24, 39, 245  
 z , 239  
 , 88, 91, 95, 112  
**champs** , 50, 60  
**conditions\_initiales** , 87, 94, 101, 103–109, 111, 116, 118, 131, 132  
**conditions\_limites** , 87, 94, 101, 103–109, 111, 116, 118, 131, 132  
**fichier** , 32  
**nom\_zones** , 29  
**partitionneur** , 29  
**postraitement** , 49, 63–75, 77–84, 86  
**postraitements** , 49, 63–75, 77–84, 86  
**save\_matrice** , 146, 147, 152  
**sondes** , 50, 60  
**1D** , 114  
**3D** , 114  
**A\_plus** , 246  
**acceleration** , 233  
**alias** , 102, 104  
**alpha** , 97, 98, 248  
**alpha\_0** , 196  
**alpha\_1** , 196  
**alpha\_a** , 196  
**alpha\_sous\_zone** , 98  
**amont\_sous\_zone** , 98  
**ampli\_bruit** , 171  
**ampli\_sin** , 171  
**ascii** , 5, 42  
**avec\_certains\_bords** , 15  
**avec\_certains\_bords\_pour\_extraire\_surface** , 14  
**avec\_les\_bords** , 15  
**beta\_co** , 188  
**beta\_th** , 187, 188  
**binaire** , 9, 32  
**boite** , 244  
**bord** , 7, 113, 235  
**bords\_a\_decouper** , 9  
**boundaries** , 119  
**boundary\_conditions** , 87, 94, 101, 103–109, 111, 116, 118, 131, 132  
**boundary\_xmax** , 26  
**boundary\_xmin** , 26

boundary\_ymax , 26  
 boundary\_ymin , 26  
 boundary\_zmax , 26  
 boundary\_zmin , 26  
 btd , 100  
 c0 , 234  
 c1\_eps , 233, 243  
 c2\_eps , 233, 243  
 c3\_eps , 233, 243  
 calc\_spectre , 114  
 calcul\_ldp\_en\_flux\_impose , 250  
 canalx , 126  
 centre\_rotation , 233  
 champ\_med , 20  
 changement\_de\_base\_p1bulle , 167  
 cl\_presson\_sommet\_faible , 167  
 cmu , 129  
 coef , 186  
 coeff , 235  
 coefficient\_diffusion , 187  
 coefficients\_activites , 144  
 compo , 139  
 condition\_elements , 14, 15  
 condition\_faces , 15  
 condition\_geometrique , 9  
 conduction , 64  
 conservation\_Ec , 114  
 constante\_modele\_micro\_melange , 143  
 constante\_taux\_reaction , 144  
 contre\_energie\_activation , 144  
 contre\_reaction , 144  
 controle\_residu , 147, 229–232  
 convection , 94, 101, 102, 104–109, 111, 116, 118, 131, 132  
 convection\_diffusion\_chaleur\_qc , 78, 79  
 convection\_diffusion\_chaleur\_turbulent\_qc , 82, 83  
 convection\_diffusion\_concentration , 66, 67, 73, 74  
 convection\_diffusion\_concentration\_turbulent , 68, 69, 75, 76  
 convection\_diffusion\_temperature , 72–74, 80  
 convection\_diffusion\_temperature\_turbulent , 75, 76, 81, 84  
 correction\_visco\_turb\_pour\_controle\_pas\_de\_temps , 119, 121, 122, 124, 125, 127–129  
 correction\_visco\_turb\_pour\_controle\_pas\_de\_temps\_parametre , 119, 121, 123–125, 127–129  
 corriger\_partition , 193  
 couronne , 244  
 Cp , 185  
 cp , 160, 161, 167, 187–190  
 crank , 93  
 critere\_absolu , 16  
 cs , 123  
 Cv , 185  
 cw , 122  
 d , 175, 178  
 debit , 160, 161  
 debit\_impose , 235  
 debut\_stat , 113  
 definition\_champs , 50, 60  
 delta , 159  
 derivee\_rotation , 186  
 dh , 160, 161  
 diag , 147  
 diam\_hydr , 237, 238, 249, 250  
 diam\_hydr\_ortho , 237  
 diffusion , 87, 94, 101, 103–109, 111, 116, 118, 131, 132  
 diffusion\_implicite , 198, 200, 202, 203, 205, 207, 208, 210, 211, 213, 215, 217, 220, 222, 224, 226, 228  
 dim\_espace\_krilov , 147  
 dir , 160, 161  
 dir\_flow , 171  
 dir\_wall , 172  
 direction , 7, 16–18, 113, 237  
 dmax , 126  
 domain , 26  
 domaine , 7, 9, 14–18, 32, 50, 60, 137, 138, 194  
 domaine\_final , 8, 16  
 domaine\_grossier , 9  
 domaine\_init , 8, 16  
 domaines , 32  
 domegadt , 233  
 dt\_impr , 119, 160, 161, 197, 199, 202, 203, 205, 206, 208, 210, 211, 213, 215, 217, 219, 222, 224, 226, 228  
 dt\_impr\_moy\_spat , 113  
 dt\_impr\_moy\_temp , 113  
 dt\_impr\_nusselt , 190–192  
 dt\_impr\_ustar , 119, 121, 123–125, 127–129  
 dt\_impr\_ustar\_mean\_only , 119, 121, 123–125, 127–129  
 dt\_max , 197, 199, 201, 203, 205, 206, 208, 209, 211, 213, 215, 217, 219, 222, 224, 226, 227  
 dt\_min , 197, 199, 201, 203, 205, 206, 208, 209, 211, 213, 215, 217, 219, 222, 224, 226, 227  
 dt\_projection , 111, 116, 117, 130  
 dt\_sauv , 197, 199, 202, 203, 205, 206, 208, 210, 211, 213, 215, 217, 219, 222, 224, 226, 228  
 dt\_start , 197, 200, 202, 203, 205, 206, 208, 210, 211, 213, 215, 217, 220, 222, 224, 226,

228  
 Ec , 114  
 Ec\_dans\_repere\_fixe , 114  
 ecrire\_decoupage , 29  
 ecrire\_fichier\_xyz\_valeur , 87, 94, 101, 103–109, 111, 116, 118, 131, 132  
 ecrire\_fichier\_xyz\_valeur\_bin , 87, 94, 102–109, 111, 116, 118, 131, 132  
 ecrire\_frontiere , 32  
 ecrire\_lata , 30  
 emissivite\_pour\_rayonnement\_entre\_deux\_plaques , 171, 235  
   \_quasi\_infinies , 162  
 energie\_activation , 144  
 enthalpie\_reaction , 144  
 epaisseur , 14, 16  
 eps\_min , 119, 121, 123–125, 127–129  
 equation\_frequence\_resolue , 94  
 equation\_non\_resolue , 87, 94, 95, 102–107, 109–111, 116, 118, 131, 132  
 equations\_scalaires\_passifs , 63, 67, 69, 74, 76, 79, 80, 83, 84  
 Erugu , 246  
 erugu , 164  
 espece , 105, 106  
 espece\_en\_competition\_micro\_melange , 143  
 exposant\_beta , 144  
 expression , 143  
 facon\_init , 114  
 facsec , 197, 199, 201, 203, 205, 206, 208, 209, 211, 213, 215, 217, 219, 222, 224, 226, 227  
 facsec\_max , 199, 201, 214, 216, 219, 221, 223  
 facteur , 100, 101, 247, 251  
 facteurs , 22  
 fichier , 50, 60, 126, 193, 244  
 fichier\_ecriture\_K\_Eps , 126  
 fichier\_matrice , 41  
 fichier\_post , 8  
 fichier\_secmem , 41  
 fichier\_solution , 41  
 fichier\_solveur , 42  
 fichier\_solveur\_non\_recree , 148  
 fichier\_sortie , 20  
 fields , 50, 60  
 file , 32  
 file\_coord\_x , 26  
 file\_coord\_y , 26  
 file\_coord\_z , 26  
 fin\_stat , 113  
 fonction , 37, 125  
 fonction\_filtre , 27  
 fonction\_sous\_zone , 244  
 format , 32, 50, 60  
 format\_post , 27  
 formatte , 30  
 formulation\_a\_nb\_points , 121–123, 125, 126, 128  
 frequence\_recalc , 148  
 function\_coord\_x , 26  
 function\_coord\_y , 26  
 function\_coord\_z , 26  
 gamma , 185  
 genere\_fichier\_solveur , 42  
 ghost\_thickness , 26  
 groupes , 62  
 h , 171, 235  
 hexa\_old , 16  
 impr , 41, 145–147, 152  
 impr\_diffusion\_implicit , 198, 200, 202, 204, 205, 207, 208, 210, 212, 213, 215, 218, 220, 222, 225, 226, 228  
 indice , 188, 189  
 info , 89  
 init\_Ec , 114  
 initial\_conditions , 87, 94, 101, 103–109, 111, 116, 118, 131, 132  
 initial\_value , 172, 173, 178  
 interfaces , 50, 60  
 intervalle , 244  
 inverse\_condition\_element , 14  
 joints\_non\_postraites , 32  
 k , 188  
 k\_min , 119, 121, 123–125, 127–129  
 kappa , 188, 189, 246, 248  
 kmetis , 193  
 lambda , 160, 161, 167, 187–190, 236–238, 241, 248  
 lambda\_c , 249  
 lambda\_max , 241  
 lambda\_min , 241  
 lambda\_ortho , 237  
 larg\_joint , 29  
 liste , 37, 244  
 liste\_cas , 12  
 liste\_de\_postraitements , 49, 63–75, 77–84, 86  
 liste\_postraitements , 49, 63–75, 77–84, 86  
 localisation , 27, 138, 143  
 loi\_etat , 189  
 longueur\_boite , 114  
 longueur\_maille , 121, 122, 124–126, 128  
 longueurs , 22  
 main , 30  
 masse\_molaire , 102, 104, 167  
 max\_iter\_implicit , 214, 217, 219, 221, 223, 225  
 methode , 20, 137, 138, 141, 142  
 methode\_calcul\_face\_keps\_impose , 246  
 methode\_calcul\_pression\_initiale , 110, 115, 117, 130  
 min\_dir\_flow , 172

min\_dir\_wall , 172  
 mode\_calcul\_convection , 94, 101  
 modele\_fonc\_bas\_reynolds , 129  
 modele\_micro\_melange , 143  
 modele\_turbulence , 101, 104, 106, 108, 117, 130  
 modele\_visco , 247, 251  
 modif\_div\_face\_dirichlet , 167  
 moyenne\_convergee , 140  
 mu , 160, 161, 167, 188, 189, 248  
 n , 161, 188, 247, 251  
 name\_of\_initial\_zones , 5  
 name\_of\_new\_zones , 5  
 navier\_stokes\_qc , 78, 79  
 navier\_stokes\_standard , 65–67, 72–74, 80  
 navier\_stokes\_turbulent , 68–70, 75, 76, 81, 84  
 navier\_stokes\_turbulent\_qc , 82, 83  
 nb\_comp , 172, 173, 178, 251  
 nb\_corrections\_max , 229–231  
 nb\_it\_max , 146, 147, 152, 229–232  
 nb\_nodes , 26  
 nb\_parts , 192–195  
 nb\_parts\_geom , 9  
 nb\_parts\_naif , 9  
 nb\_parts\_tot , 30  
 nb\_pas\_dt\_max , 197, 199, 201, 203, 205, 206, 208, 210, 211, 213, 215, 217, 219, 222, 224, 226, 227  
 nb\_points\_par\_phase , 113  
 nb\_procs , 12  
 nb\_test , 41  
 nb\_tranche , 20  
 nb\_tranches , 16–18  
 nb\_var , 125  
 new\_jacobian , 89  
 niter\_avg , 199, 201  
 niter\_max , 199, 201  
 niter\_max\_diffusion\_implicite , 93, 198, 200, 202, 204, 205, 207, 208, 210, 212, 213, 215, 218, 220, 222, 224, 226, 228  
 niter\_min , 199, 201  
 no\_check\_disk\_space , 198, 200, 202, 204, 205, 207, 209, 210, 212, 213, 216, 218, 220, 222, 225, 227, 228  
 no\_conv\_subiteration\_diffusion\_implicite , 198, 200, 202, 204, 205, 207, 209, 210, 212, 213, 216, 218, 220, 222, 225, 227, 228  
 no\_error\_if\_not\_converged\_diffusion\_implicite , 198, 200, 202, 204, 205, 207, 209, 210, 212, 213, 216, 218, 220, 222, 225, 227, 228  
 no\_qdm , 229–232  
 nom , 172, 173, 178  
 nom\_bord , 16  
 nom\_cl\_derriere , 18  
 nom\_cl\_devant , 18  
 nom\_domaine , 27  
 nom\_fichier\_post , 27  
 nom\_fichier\_solveur , 148  
 nom\_fichier\_sortie , 9  
 nom\_frontiere , 137  
 nom\_inconnue , 102, 104  
 nom\_pb , 27  
 nom\_source , 133–143  
 nombre\_de\_noeuds , 22  
 noms\_champs , 27  
 non\_perio , 16  
 normal\_value , 177  
 nu , 89, 160, 161  
 nu\_transp , 89  
 numero , 139, 143  
 numero\_op , 135  
 numero\_source , 135  
 nusselt , 250  
 nut , 89  
 nut\_max , 119, 121, 123–125, 127–129  
 nut\_transp , 89  
 old , 98  
 omega , 171, 196, 199, 233  
 omega\_relaxation\_drho\_dt , 189  
 optimisation\_sous\_maillage , 138  
 optimized , 146, 152  
 option , 139, 233  
 Origine , 22  
 origine , 14  
 p0 , 167  
 p1 , 167  
 p\_imposee\_aux\_faces , 28  
 pa , 167  
 par\_sous\_zone , 8  
 parametre\_equation , 87, 95, 102–107, 109–111, 116, 118, 131, 132  
 Partition\_tool , 29  
 pas\_de\_solution\_initiale , 42  
 pb\_champ , 140, 141  
 pb\_name , 30  
 penalisation\_l2\_ftd , 107  
 perio\_x , 26  
 perio\_y , 26  
 perio\_z , 26  
 periode , 114  
 periode\_calc\_spectre , 114  
 periode\_sauvegarde\_securite\_en\_heures , 198, 200, 202, 204, 205, 207, 209, 210, 212, 213, 216, 218, 220, 222, 225, 227, 228  
 periodique , 30  
 point1 , 14  
 point2 , 14  
 point3 , 14

polynomes , 244  
 position , 186  
 Post\_processing , 49, 63–75, 77–84, 86  
 Post\_processings , 49, 63–75, 77–84, 86  
 prandtl\_turbulent\_fonction\_nu\_t\_alpha , 191  
 Prandtl , 185  
 prandtl , 251  
 prandtl\_eps , 119, 121, 123–125, 127–129  
 prandtl\_k , 119, 121, 123–125, 127–129  
 prdt , 191  
 prdt\_sur\_kappa , 250  
 precision\_impr , 198, 200, 202, 204, 205, 207, 209, 210, 212, 213, 216, 218, 220, 222, 225, 226, 228  
 precond , 146, 152  
 precond0 , 196  
 precond1 , 196  
 precond\_nul , 146, 152  
 preconda , 196  
 preconditionnement\_diag , 93  
 pression , 189  
 Probes , 50, 60  
 probleme , 14, 15, 172, 173, 178  
 produits , 144  
 projection\_initiale , 110, 115, 117, 130  
 projection\_normale\_bord , 16  
 pulsation\_w , 113  
 quiet , 145–147, 152  
 reactifs , 144  
 reactions , 143  
 rectangle , 244  
 relax\_pression , 231  
 reorder , 30  
 reprise , 49, 63–74, 76–83, 85, 86, 113  
 reprise\_correlation , 161, 162  
 resolution\_explicite , 94  
 restart , 247  
 restriction , 244  
 resume\_last\_time , 50, 63–68, 70–73, 75–82, 84–86  
 rho , 160, 161, 187–190  
 rho\_constant\_pour\_debug , 185  
 rotation , 186  
 sans\_passer\_par\_le2D , 16  
 sans\_solveur\_masse , 135  
 sans\_source\_boussinesq , 248  
 sauvegarde , 49, 63–75, 77–84, 86  
 sauvegarde\_simple , 49, 63–74, 76–83, 85, 86  
 save\_matrix , 146, 147, 152  
 Sc , 185  
 scturb , 191  
 segment , 244  
 seuil , 146, 147, 152, 199, 201  
 seuil\_convergence\_implicit , 93, 229–232  
 seuil\_convergence\_solveur , 94, 229–232  
 seuil\_diffusion\_implicit , 93, 198, 200, 202, 204, 205, 207, 208, 210, 212, 213, 215, 218, 220, 222, 224, 226, 228  
 seuil\_divU , 111, 116, 117, 130  
 seuil\_generation\_solveur , 229–232  
 seuil\_statio , 198, 200, 202, 203, 205, 207, 208, 210, 211, 213, 215, 217, 220, 222, 224, 226, 228  
 seuil\_statio\_relatif\_deconseille , 198, 200, 202, 203, 205, 207, 208, 210, 211, 213, 215, 217, 220, 222, 224, 226, 228  
 seuil\_test\_preliminaire\_solveur , 229–232  
 seuil\_verification , 42  
 seuil\_verification\_solveur , 229–232  
 solveur , 42, 94, 214, 217, 219, 221, 224, 225, 229–232  
 solveur0 , 146  
 solveur1 , 146  
 solveur\_bar , 111, 115, 117, 130  
 solveur\_pression , 111, 115, 117, 130  
 sonde\_tble , 247, 251  
 source , 133–143  
 source\_reference , 133–143  
 sources , 87, 94, 101, 103–109, 111, 116, 118, 131–143  
 sources\_reference , 133–143  
 sous\_zone , 14, 172, 173, 178, 237, 238  
 sous\_zones , 194  
 splitting , 26  
 standard , 89  
 stationnaire , 248  
 statistiques , 50, 60  
 statistiques\_en\_serie , 50, 60  
 stats , 247, 251  
 surface , 161  
 surfacique , 31  
 sutherland , 189  
 symx , 22  
 symy , 22  
 symz , 22  
 t0 , 234  
 t\_deb , 135–137, 140  
 t\_fin , 135–137, 140  
 tanh , 22  
 tanh\_dilatation , 22  
 tanh\_taille\_premiere\_maille , 23  
 tcpumax , 197, 199, 201, 203, 205, 206, 208, 209, 211, 213, 215, 217, 219, 221, 224, 226, 227  
 tdivu , 98  
 temps\_debut\_prise\_en\_compte\_drho\_dt , 189  
 test , 98  
 tinf , 160, 161

**tinit** , 197, 199, 201, 203, 205, 206, 208, 209, 211, 213, 215, 217, 219, 221, 224, 226, 227  
**tmax** , 197, 199, 201, 203, 205, 206, 208, 209, 211, 213, 215, 217, 219, 221, 224, 226, 227  
**traitement\_coins** , 28  
**traitement\_particulier** , 111, 116, 118, 131  
**traitement\_pth** , 189  
**traitement\_rho\_gravite** , 189  
**tranches** , 195  
**transport\_k\_epsilon** , 129  
**triangle** , 14  
**trois\_tetra** , 16  
**tsup** , 160, 161  
**tube** , 244  
**turbulence\_paro** , 119, 121, 123–125, 127–129, 190–192  
**tuyauz** , 126  
**type** , 139  
**u** , 175, 177  
**u\_star\_impose** , 246  
**u\_tau** , 249  
**ubar\_umprim\_cible** , 241  
**ucent** , 171  
**union** , 244  
**use\_weights** , 193  
**val\_Ec** , 114  
**verif\_boussinesq** , 234  
**verif\_dparoi** , 126  
**via\_extraire\_surface** , 14  
**vingt\_tetra** , 16  
**vitesse** , 186, 233  
**volume** , 160  
**volumes\_etendus** , 98  
**volumes\_non\_etendus** , 98  
**volumique** , 31  
**with\_nu** , 132  
**xinf** , 161  
**xsup** , 161  
**zmax** , 20  
**zmin** , 20  
**zones\_name** , 29  
  
**acceleration** , 233  
**amont** , 95  
**amont\_old** , 95  
**analyse\_angle** , 5  
**associate** , 5  
**axi** , 6  
  
**bidim\_axi** , 6  
**bord** , 23  
**bord\_base** , 23  
**boundary\_field\_inward** , 177  
**boundary\_field\_uniform\_keps\_from\_ud** , 177  
  
**boussinesq\_concentration** , 234  
**boussinesq\_temperature** , 234  
**btd** , 100  
  
**calcul** , 6  
**calculer\_moments** , 6  
**canal** , 113  
**canal\_perio** , 234  
**centre** , 96  
**centre4** , 96  
**centre\_de\_gravite** , 7  
**centre\_old** , 96  
**ch\_front\_input** , 178  
**ch\_front\_input\_uniforme** , 178  
**champ\_base** , 168  
**champ\_don\_base** , 168  
**champ\_don\_lu** , 168  
**champ\_fonc\_fonction** , 168  
**champ\_fonc\_fonction\_txyz** , 169  
**champ\_fonc\_med** , 169  
**champ\_fonc\_reprise** , 169  
**champ\_fonc\_t** , 170  
**champ\_fonc\_tabule** , 170  
**champ\_fonc\_txyz** , 175  
**champ\_fonc\_xyz** , 175  
**champ\_front\_base** , 177  
**champ\_front\_bruite** , 178  
**champ\_front\_calc** , 179  
**champ\_front\_contact\_vef** , 179  
**champ\_front\_debit** , 179  
**champ\_front\_fonc\_pois\_ipsn** , 180  
**champ\_front\_fonc\_pois\_tube** , 180  
**champ\_front\_fonc\_txyz** , 180  
**champ\_front\_fonc\_xyz** , 180  
**champ\_front\_fonction** , 181  
**champ\_front\_lu** , 181  
**champ\_front\_normal\_vef** , 181  
**champ\_front\_pression\_from\_u** , 182  
**champ\_front\_recyclage** , 182  
**champ\_front\_tabule** , 183  
**champ\_front\_tangentiel\_vef** , 184  
**champ\_front\_uniforme** , 184  
**champ\_generique\_base** , 133  
**champ\_init\_canal\_sinal** , 171  
**champ\_input\_base** , 172  
**champ\_input\_p0** , 172  
**champ\_ostwald** , 173  
**champ\_post\_de\_champs\_post** , 133  
**champ\_post\_extraction** , 137  
**champ\_post\_interpolation** , 138  
**champ\_post\_morceau\_equation** , 139  
**champ\_post\_operateur\_base** , 134  
**champ\_post\_operateur\_divergence** , 136  
**champ\_post\_operateur\_eqn** , 134

champ\_post\_operateur\_gradient, 137  
 champ\_post\_reduction\_0d, 140  
 champ\_post\_refchamp, 141  
 champ\_post\_statistiques\_base, 135  
 champ\_post\_tparoi\_vef, 141  
 champ\_post\_transformation, 142  
 champ\_som\_lu\_vdf, 173  
 champ\_som\_lu\_vef, 173  
 champ\_tabule\_temps, 174  
 champ\_uniforme\_morceaux, 174  
 champ\_uniforme\_morceaux\_tabule\_temps, 174  
 Champ\_front\_fonc\_txyz, 3  
 chimie, 143  
 chmoy\_faceperio, 114  
 Cholesky, 148–150  
 cholesky, 144  
 circle, 54  
 circle\_3, 54  
 class\_generic, 144  
 combinaison, 124  
 Concentration, 56, 58  
 condlim\_base, 153  
 condlims, 91  
 conduction, 87  
 constituant, 187  
 convection\_deriv, 95  
 convection\_diffusion\_chaleur\_qc, 94  
 convection\_diffusion\_chaleur\_turbulent\_qc, 101  
 convection\_diffusion\_concentration, 102  
 convection\_diffusion\_concentration\_turbulent, 103  
 convection\_diffusion\_fraction\_massique\_qc, 104  
 convection\_diffusion\_fraction\_massique\_turbulent\_qc, 105  
 convection\_diffusion\_temperature, 106  
 convection\_diffusion\_temperature\_turbulent, 108  
 coriolis, 235  
 Correlation, 55, 56  
 correlation, 58, 135  
 corriger\_frontiere\_periodique, 7  
 create\_domain\_from\_sous\_zone, 8  
  
 darcy, 235  
 debug, 8  
 decoupebord\_pour\_rayonnement, 9  
 decouper\_bord\_coincident, 9  
 di\_l2, 96  
 diffusion\_deriv, 88  
 dilate, 10  
 dimension, 10  
 dirac, 236  
 dirichlet, 153  
 discretisation\_base, 166  
 discretiser\_domaine, 10  
 discretize, 10  
  
 distance\_paro, 11  
 domain, 25  
 domaine, 167  
 dt\_calc, 145  
 dt\_fixe, 145  
 dt\_min, 145  
 dt\_start, 145  
 Dt\_post, 55, 56  
  
 ec, 113  
 ecart\_type, 57, 136  
 Ecart\_type, 55, 56, 58  
 ecrire, 48  
 ecrire\_champ\_med, 11  
 ecrire\_fichier\_bin, 48  
 ecrire\_fichier\_formatte, 11  
 ecrire\_med, 48  
 ecriturelecturespecial, 12  
 ef, 96, 166  
 ef\_stab, 97  
 end, 18  
 entree\_temperature\_imposee\_h, 153  
 epsilon, 25  
 eqn\_base, 109  
 execute\_parallel, 12  
 export, 12  
 extract\_2d\_from\_3d, 13  
 extract\_2daxi\_from\_3d, 13  
 extraire\_domaine, 13  
 extraire\_plan, 14  
 extraire\_surface, 14  
 extrudebord, 15  
 extrudeparoi, 16  
 extruder, 16  
 extruder\_en20, 17  
 extruder\_en3, 17  
  
 fichier\_decoupage, 192  
 field\_uniform\_keps\_from\_ud, 175  
 fluide\_incompressible, 187  
 fluide\_ostwald, 188  
 fluide\_quasi\_compressible, 188  
 forchheimer, 236  
 frontiere\_ouverte, 153  
 frontiere\_ouverte\_concentration\_imposee, 154  
 frontiere\_ouverte\_fraction\_massique\_imposee, 154  
 frontiere\_ouverte\_gradient\_pression\_impose, 154  
 frontiere\_ouverte\_gradient\_pression\_impose\_vef, 154  
 frontiere\_ouverte\_gradient\_pression\_impose\_vefprep1b, 155  
 frontiere\_ouverte\_gradient\_pression\_libre\_vef, 155  
 frontiere\_ouverte\_gradient\_pression\_libre\_vefprep1b, 155  
 frontiere\_ouverte\_k\_eps\_impose, 155



frontiere\_ouverte\_pression\_imposee, 156  
 frontiere\_ouverte\_pression\_imposee\_orlansky, 156  
 frontiere\_ouverte\_pression\_moyenne\_imposee, 156  
 frontiere\_ouverte\_rho\_u\_impose, 156  
 frontiere\_ouverte\_temperature\_imposee, 157  
 frontiere\_ouverte\_vitesse\_imposee, 157  
 frontiere\_ouverte\_vitesse\_imposee\_sortie, 157  
  
 gaz\_parfait, 185  
 gaz\_reel\_rhot, 184  
 GCP, 148, 151  
 gcp, 151  
 gcp\_ns, 145  
 gen, 146  
 generic, 99  
 gmres, 146  
 Gradient, 148  
  
 IBICGSTAB, 148  
 implicite, 228  
 imprimer\_flux, 18  
 imprimer\_flux\_sum, 19  
 init\_par\_partie, 175  
 integrer\_champ\_med, 19  
 Interface, 149  
 internes, 24  
 interpreter, 4  
  
 k\_epsilon, 128  
 kquick, 99  
  
 lata\_to\_med, 20  
 lata\_to\_other, 20  
 leap\_frog, 204  
 lire\_ideas, 21  
 lire\_tgrid, 34  
 list\_bloc\_mailler, 21  
 list\_bord, 23  
 list\_nom, 41  
 list\_nom\_virgule, 133  
 liste\_post, 60  
 liste\_post\_ok, 59  
 listobj, 252  
 listobj\_impl, 251  
 local, 150  
 loi\_analytique\_scalaire, 249  
 loi\_etat\_base, 184  
 loi\_expert\_hydr, 246  
 loi\_expert\_scalaire, 249  
 loi\_fermeture\_base, 186  
 loi\_fermeture\_test, 186  
 loi\_horaire, 186, 253  
 loi\_paroι\_nu\_impose, 250  
 loi\_standard\_hydr, 246  
 loi\_standard\_hydr\_old, 246  
 loi\_standard\_hydr\_scalaire, 250  
 longitudinale, 238  
 longueur\_melange, 126  
  
 mailler, 21  
 mailler\_base, 21  
 maillerparallel, 25  
 melange\_gaz\_parfait, 185  
 methode\_transport\_deriv, 253  
 metis, 193  
 milieu\_base, 186  
 mod\_turb\_hyd\_ss\_maille, 120  
 modele\_fonction\_bas\_reynolds\_base, 129  
 modele\_turbulence\_hyd\_deriv, 118  
 modele\_turbulence\_scal\_base, 190  
 modif\_bord\_to\_raccord, 26  
 mor\_eqn, 87  
 Moyenne, 55, 56, 58  
 moyenne, 57, 139  
 moyenne\_volumique, 27  
 muscl, 99  
 muscl3, 97  
 muscl\_new, 100  
 muscl\_old, 99  
  
 N, 149  
 navier\_stokes\_qc, 110  
 navier\_stokes\_standard, 115  
 navier\_stokes\_turbulent, 116  
 navier\_stokes\_turbulent\_qc, 129  
 negligeable, 88, 100, 247  
 negligeable\_scalaire, 250  
 nettoiepasnoeuds, 28  
 neumann, 157  
 nom, 192  
 NUL, 119  
 NULL, 150  
 numero\_elem\_sur\_maitre, 52  
  
 objet\_lecture, 252  
 optimal, 147  
 option, 90  
 option\_vdf, 28  
 orientefacesbord, 28  
 orienter\_simplexes, 35  
  
 p1b, 88  
 p1ncp1b, 88  
 parametre\_diffusion\_implicite, 93  
 parametre\_equation\_base, 92  
 parametre\_implicite, 93  
 Paroi, 153  
 paroi\_adiabatique, 158

paroi\_contact, 158  
 paroi\_contact\_fictif, 158  
 paroi\_couple, 159  
 paroi\_decalee\_robin, 159  
 paroi\_defilante, 159  
 paroi\_echange\_contact\_correlation\_vdf, 160  
 paroi\_echange\_contact\_correlation\_vdf, 161  
 paroi\_echange\_contact\_vdf, 162  
 paroi\_echange\_externer\_impose, 162  
 paroi\_echange\_externer\_impose\_h, 162  
 paroi\_echange\_global\_impose, 163  
 paroi\_fixe, 163  
 paroi\_fixe\_iso\_Genepi2\_sans\_contribution\_aux\_vitesse\_sommets, 163  
 paroi\_flux\_impose, 163  
 paroi\_ft\_disc\_deriv, 252  
 paroi\_knudsen\_non\_negligeable, 163  
 paroi\_rugueuse, 164  
 paroi\_tble, 247  
 paroi\_tble\_scal, 250  
 paroi\_temperature\_imposee, 164  
 partition, 28, 194  
 partitionneur\_deriv, 192  
 pave, 21  
 pb\_avec\_passif, 62  
 Pb\_base, 49  
 pb\_conduction, 63  
 pb\_gen\_base, 49  
 pb\_hydraulique, 64  
 pb\_hydraulique\_concentration, 65  
 pb\_hydraulique\_concentration\_scalaires\_passifs, 66  
 pb\_hydraulique\_concentration\_turbulent, 67  
 pb\_hydraulique\_concentration\_turbulent\_scalaires\_passifs, 68  
 pb\_hydraulique\_turbulent, 70  
 pb\_thermohydraulique, 71  
 pb\_thermohydraulique\_concentration, 72  
 pb\_thermohydraulique\_concentration\_scalaires\_passifs, 74  
 pb\_thermohydraulique\_concentration\_turbulent, 75  
 pb\_thermohydraulique\_concentration\_turbulent\_scalaires\_passifs, 76  
 pb\_thermohydraulique\_qc, 77  
 pb\_thermohydraulique\_qc\_fraction\_massique, 78  
 pb\_thermohydraulique\_scalaires\_passifs, 79  
 pb\_thermohydraulique\_turbulent, 80  
 pb\_thermohydraulique\_turbulent\_qc, 81  
 pb\_thermohydraulique\_turbulent\_qc\_fraction\_massique, 82  
 pb\_thermohydraulique\_turbulent\_scalaires\_passifs, 84  
 pbc\_med, 85  
 periodique, 164  
 perte\_charge\_anisotrope, 236  
 perte\_charge\_circulaire, 237  
 perte\_charge\_directionnelle, 237  
 perte\_charge\_isotrope, 237  
 perte\_charge\_reguliere, 238  
 perte\_charge\_singuliere, 239  
 Petsc, 148, 150  
 petsc, 148  
 pilote\_icoco, 30  
 piso, 229  
 plan, 53  
 point, 52  
 points, 52  
 porosites, 30  
 porosites\_champ, 31  
 position\_like, 53  
 post\_processing, 59  
 post\_processings, 58  
 postraitement\_base, 59  
 postraiter\_domaine, 31  
 pp, 107  
 prandtl, 190  
 precisiongeom, 32  
 Precond, 148, 150  
 precondition\_base, 195  
 precondition\_local, 195  
 precondition\_solver, 195  
 predefini, 140  
 Pression, 56, 58  
 Print, 149  
 problem\_read\_generic, 85  
 probleme\_couple, 61  
 puissance\_thermique, 240  
 quick, 100  
 raccord, 24  
 raffiner\_anisotrope, 32  
 raffiner\_isotrope, 33  
 Raffiner\_isotrope\_parallele, 5  
 read, 33  
 read\_file, 34  
 read\_file\_binary, 34  
 read\_med, 35  
 read\_unsupported\_ascii\_file\_from\_icem, 34  
 redresser\_hexaedres\_vdf, 36  
 refine\_mesh, 36  
 regroupebord, 36  
 remove\_elem, 36  
 remove\_invalid\_internal\_boundaries, 37  
 reordonner, 38  
 reordonner\_faces\_periodiques, 38  
 reorienter\_tetraedres, 38  
 reorienter\_triangles, 38  
 rotation, 39  
 runge\_kutta\_ordre\_3, 206

runge\_kutta\_ordre\_4\_d3p, 207  
 runge\_kutta\_rationnel\_ordre\_2, 209  
  
 scalaire\_impose\_pari, 165  
 scatter, 39  
 scatterformatte, 39  
 scattermed, 40  
 Sch\_CN\_EX\_iteratif, 198  
 Sch\_CN\_iteratif, 200  
 schema\_adams\_bashforth\_order\_2, 210  
 schema\_adams\_bashforth\_order\_3, 212  
 schema\_adams\_moulton\_order\_2, 214  
 schema\_adams\_moulton\_order\_3, 216  
 schema\_backward\_differentiation\_order\_2, 218  
 schema\_backward\_differentiation\_order\_3, 220  
 schema\_implicite\_base, 225  
 schema\_predictor\_corrector, 227  
 schema\_temps\_base, 197  
 scheme\_euler\_explicit, 202  
 scheme\_euler\_implicit, 222  
 schmidt, 191  
 segment, 53  
 segmentpoints, 52  
 simple, 230  
 simplifier, 231  
 solide, 190  
 solve, 40  
 Solver, 148, 151  
 Solveur, 148, 150  
 solveur\_implicite\_base, 228  
 solveur\_lineaire\_std, 232  
 solveur\_sys\_base, 152  
 Solveur\_pression, 148, 150  
 sonde\_base, 51  
 sortie\_libre\_temperature\_imposee\_h, 165  
 source\_base, 232  
 source\_constituant, 240  
 source\_generique, 240  
 source\_qdm, 241  
 source\_qdm\_lambdaup, 241  
 source\_robin, 241  
 source\_robin\_scalaire, 242  
 source\_th\_tdivu, 242  
 source\_transport\_k\_eps, 242  
 source\_transport\_k\_eps\_aniso\_concen, 243  
 source\_transport\_k\_eps\_aniso\_therm\_concen, 243  
 Source\_Transport\_K\_Eps\_anisotherme, 232  
 sources, 91  
 sous\_maille, 127  
 sous\_maille\_smago, 123  
 sous\_maille\_wale, 122  
 sous\_zone, 243  
 sous\_zones, 194  
 Spai, 150  
  
 spec\_pdc\_base, 238  
 SSOR, 150, 151  
 ssor, 196  
 ssor\_bloc, 196  
 stab, 88  
 standard, 89  
 stat\_post\_deriv, 56  
 Statistiques, 56, 58  
 Statistiques\_en\_serie, 58  
 sug, 100  
 supprime\_bord, 40  
 symetrie, 165, 253  
 system, 41  
  
 t\_deb, 57  
 t\_fin, 57  
 tayl\_green, 176  
 Temperature, 56, 58  
 temperature, 112  
 temperature\_imposee\_pari, 165  
 test\_solveur, 41  
 testeur, 42  
 testeur\_medcoupling, 42  
 tetraedriser, 42  
 tetraedriser\_homogene, 43  
 tetraedriser\_homogene\_compact, 43  
 tetraedriser\_homogene\_fin, 43  
 tetraedriser\_par\_prisme, 44  
 thi, 114  
 traitement\_particulier\_base, 112  
 tranche, 194  
 transformer, 45  
 transport\_k\_epsilon, 131  
 transversale, 239  
 trianguler, 45  
 trianguler\_fin, 45  
 trianguler\_h, 46  
 turbulence\_pari\_base, 246  
 turbulence\_pari\_scalaire\_base, 249  
 type, 55, 56, 58, 149, 150  
  
 uniform\_field, 176  
 utau\_imp, 249  
  
 valeur\_totale\_sur\_volume, 176  
 vdf, 166  
 vect\_nom, 47  
 vef, 166  
 vefprep1b, 166  
 verifier\_qualite\_raffinements, 46  
 verifier\_simplexes, 47  
 verifiercoin, 47  
 Vitesse, 56, 58  
 volume, 53

xyz, 3