

TrioCFD Reference Manual V1.8.0

Support team: trust@cea.fr

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

November 28, 2019

Contents

1	Syntax to define a mathematical function	15
2	Existing & predefined fields names	16
3	interprete	18
3.1	Op_Conv_EF_Stab_PolyMAC_Face	18
3.2	Raffiner_isotrope_parallele	19
3.3	read_med	19
3.4	lire_medfile	20
3.5	Solver_moving_mesh_ALE	20
3.6	bloc_lecture	20
3.7	analyse_angle	21
3.8	associate	21
3.9	associer_algo	21
3.10	associer_pbmng_pbin	22
3.11	associer_pbmng_pbgglobal	22
3.12	axi	22
3.13	bidim_axi	22
3.14	calculer_moments	23
3.15	lecture_bloc_moment_base	23
3.15.1	calcul	23
3.15.2	centre_de_gravite	23
3.15.3	un_point	23
3.16	corriger_frontiere_periodique	24
3.17	create_domain_from_sous_zone	24
3.18	debog	25
3.19	{	25
3.20	decoupebord_pour_rayonnement	25
3.21	decouper_bord_coincident	26
3.22	dilate	26
3.23	dimension	27
3.24	disable_TU	27
3.25	discretiser_domaine	27
3.26	discretize	27
3.27	distance_paro	28
3.28	ecrire_champ_med	28
3.29	ecrire_fichier_formatte	28
3.30	ecriturelecturespecial	28
3.31	execute_parallel	29
3.32	export	29
3.33	extract_2d_from_3d	29
3.34	extract_2daxi_from_3d	30
3.35	extraire_domaine	30
3.36	extraire_plan	30
3.37	extraire_surface	31
3.38	extrudebord	32
3.39	extrudeparoi	33
3.40	extruder	33
3.41	troisf	33
3.42	extruder_en20	34
3.43	extruder_en3	34
3.44	end	35

3.45	}	35
3.46	imposer_vit_bords_ale	35
3.47	imprimer_flux	35
3.48	imprimer_flux_sum	36
3.49	integrer_champ_med	36
3.50	interprete_geometrique_base	36
3.51	lata_to_med	37
3.52	format_lata_to_med	37
3.53	lata_to_other	37
3.54	lire_ideas	37
3.55	mailler	38
3.56	list_bloc_mailler	38
3.56.1	mailler_base	38
3.56.2	pave	38
3.56.3	bloc_pave	39
3.56.4	list_bord	40
3.56.5	bord_base	40
3.56.6	bord	40
3.56.7	defbord	40
3.56.8	defbord_2	41
3.56.9	defbord_3	41
3.56.10	raccord	41
3.56.11	internes	42
3.56.12	epsilon	42
3.56.13	domain	42
3.57	maillerparallel	43
3.58	modif_bord_to_raccord	44
3.59	moyenne_volumique	44
3.60	nettoiepasnoeuds	45
3.61	option_vdf	45
3.62	orientefacesbord	46
3.63	partition	46
3.64	bloc_decouper	46
3.65	pilote_icoco	47
3.66	porosites	48
3.67	bloc_lecture_poro	48
3.68	porosites_champ	48
3.69	postraiter_domaine	49
3.70	precisiongeom	49
3.71	raffiner_anisotrope	50
3.72	raffiner_isotrope	50
3.73	read	51
3.74	read_file	52
3.75	read_file_binary	52
3.76	lire_tgrid	52
3.77	read_unsupported_ascii_file_from_icem	52
3.78	orienter_simplexes	53
3.79	redresser_hexaedres_vdf	53
3.80	refine_mesh	53
3.81	regroupebord	54
3.82	remove_elem	54
3.83	remove_elem_bloc	54
3.84	remove_invalid_internal_boundaries	55
3.85	reordonner_faces_periodiques	55

3.86	reorienter_tetraedres	55
3.87	reorienter_triangles	56
3.88	reordonner	56
3.89	rotation	56
3.90	scatter	56
3.91	scatterformatte	57
3.92	scattermed	57
3.93	solve	57
3.94	supprime_bord	58
3.95	list_nom	58
3.96	system	58
3.97	test_solveur	58
3.98	testeur	59
3.99	testeur_medcoupling	59
3.100	tetraedriser	59
3.101	tetraedriser_homogene	60
3.102	tetraedriser_homogene_compact	60
3.103	tetraedriser_homogene_fin	61
3.104	tetraedriser_par_prisme	62
3.105	transformer	62
3.106	triangler	63
3.107	triangler_fin	63
3.108	triangler_h	64
3.109	verifier_qualite_raffinements	64
3.110	vect_nom	64
3.111	verifier_simplexes	64
3.112	verifiercoin	65
3.113	verifiercoin_bloc	65
3.114	ecrire	65
3.115	ecrire_fichier_bin	66
3.116	ecrire_med	66
3.117	ecrire_medfile	66
4	pb_gen_base	66
4.1	Pb_base	67
4.2	corps_postraitement	68
4.2.1	definition_champs	68
4.2.2	definition_champ	68
4.2.3	sondes	69
4.2.4	sonde	69
4.2.5	sonde_base	69
4.2.6	points	70
4.2.7	listpoints	70
4.2.8	point	70
4.2.9	segmentpoints	70
4.2.10	numero_elem_sur_maitre	71
4.2.11	position_like	71
4.2.12	segment	71
4.2.13	plan	71
4.2.14	volume	72
4.2.15	circle	72
4.2.16	circle_3	72
4.2.17	segmentfacesx	73
4.2.18	segmentfacesy	73

4.2.19	segmentfacesz	73
4.2.20	champs_posts	73
4.2.21	champs_a_post	74
4.2.22	champ_a_post	74
4.2.23	stats_posts	74
4.2.24	list_stat_post	75
4.2.25	stat_post_deriv	75
4.2.26	t_deb	75
4.2.27	t_fin	76
4.2.28	moyenne	76
4.2.29	ecart_type	76
4.2.30	correlation	77
4.2.31	stats_serie_posts	77
4.3	post_processings	78
4.3.1	un_postraitement	78
4.4	liste_post_ok	78
4.4.1	nom_postraitement	78
4.4.2	postraitement_base	78
4.4.3	post_processing	79
4.4.4	postraitement_ft_lata	79
4.5	liste_post	80
4.5.1	un_postraitement_spec	80
4.5.2	type_un_post	80
4.5.3	type_postraitement_ft_lata	80
4.6	format_file	80
4.7	probleme_couple	81
4.8	list_list_nom	81
4.9	modele_rayo_semi_transp	81
4.10	eq_rayo_semi_transp	82
4.10.1	condlims	83
4.10.2	condlimlu	83
4.11	pb_avec_passif	83
4.12	listeqn	84
4.13	pb_conduction	84
4.14	pb_conduction_milieu_variable	85
4.15	pb_couple_rayo_semi_transp	86
4.16	pb_hydraulique	87
4.17	pb_hydraulique_ALE	88
4.18	pb_hydraulique_concentration	89
4.19	pb_hydraulique_concentration_scalaires_passifs	90
4.20	pb_hydraulique_concentration_turbulent	91
4.21	pb_hydraulique_concentration_turbulent_scalaires_passifs	92
4.22	pb_hydraulique_turbulent	93
4.23	pb_mg	94
4.24	pb_phase_field	94
4.25	pb_post	95
4.26	pb_thermohydraulique	96
4.27	pb_thermohydraulique_concentration	97
4.28	pb_thermohydraulique_concentration_scalaires_passifs	98
4.29	pb_thermohydraulique_concentration_turbulent	99
4.30	pb_thermohydraulique_concentration_turbulent_scalaires_passifs	100
4.31	pb_thermohydraulique_qc	102
4.32	pb_thermohydraulique_qc_fraction_massique	103
4.33	pb_thermohydraulique_scalaires_passifs	104

4.34	pb_thermohydraulique_turbulent	105
4.35	pb_thermohydraulique_turbulent_qc	106
4.36	pb_thermohydraulique_turbulent_qc_fraction_massique	107
4.37	pb_thermohydraulique_turbulent_scalaires_passifs	108
4.38	pb_med	110
4.39	list_info_med	110
4.39.1	info_med	110
4.40	problem_read_generic	110
4.41	pb_couple_rayonnement	111
4.42	probleme_ft_disc_gen	112
5	mor_eqn	113
5.1	Transport_K_Eps_Realisable	113
5.2	bloc_convection	114
5.2.1	convection_deriv	114
5.2.2	amont	114
5.2.3	amont_old	114
5.2.4	centre	115
5.2.5	centre4	115
5.2.6	centre_old	115
5.2.7	di_l2	115
5.2.8	ef	115
5.2.9	bloc_ef	116
5.2.10	muscl3	116
5.2.11	ef_stab	116
5.2.12	listsous_zone_valeur	117
5.2.13	sous_zone_valeur	117
5.2.14	generic	117
5.2.15	kquick	118
5.2.16	muscl	118
5.2.17	muscl_old	118
5.2.18	muscl_new	118
5.2.19	negligeable	119
5.2.20	quick	119
5.2.21	ale	119
5.2.22	btd	119
5.2.23	supg	120
5.2.24	RT	120
5.3	bloc_diffusion	120
5.3.1	diffusion_deriv	120
5.3.2	negligeable	121
5.3.3	p1b	121
5.3.4	p1ncp1b	121
5.3.5	stab	121
5.3.6	standard	122
5.3.7	bloc_diffusion_standard	122
5.3.8	option	123
5.3.9	op_implicite	123
5.4	condinit	123
5.4.1	condinit	123
5.5	sources	124
5.6	ecrire_fichier_xyz_valeur_param	124
5.6.1	ecrire_fichier_xyz_valeur_item	124
5.6.2	bords_ecrire	124

5.7	parametre_equation_base	125
5.7.1	parametre_diffusion_implicite	125
5.7.2	parametre_implicite	125
5.8	conduction	126
5.9	conduction_milieu_variable	127
5.10	convection_diffusion_chaleur_qc	128
5.11	convection_diffusion_chaleur_turbulent_qc	129
5.12	convection_diffusion_concentration	130
5.13	convection_diffusion_concentration_ft_disc	131
5.14	convection_diffusion_concentration_turbulent	133
5.15	convection_diffusion_fraction_massique_qc	134
5.16	convection_diffusion_fraction_massique_turbulent_qc	135
5.17	convection_diffusion_phase_field	136
5.18	convection_diffusion_temperature	137
5.19	pp	138
5.19.1	penalisation_l2_ftd_lec	138
5.20	convection_diffusion_temperature_ft_disc	139
5.21	objet_lecture_maintien_temperature	140
5.22	convection_diffusion_temperature_turbulent	140
5.23	eqn_base	141
5.24	navier_stokes_ft_disc	142
5.25	penalisation_forage	145
5.26	modele_turbulence_hyd_deriv	146
5.26.1	dt_impr_ustar_mean_only	146
5.26.2	NUL	147
5.26.3	mod_turb_hyd_ss_maille	147
5.26.4	form_a_nb_points	148
5.26.5	sous_maille_wale	149
5.26.6	sous_maille_smago	150
5.26.7	combinaison	151
5.26.8	longueur_melange	152
5.26.9	sous_maille	154
5.26.10	sous_maille_selectif_mod	155
5.26.11	deuxentiers	156
5.26.12	floatentier	156
5.26.13	sous_maille_selectif	156
5.26.14	sous_maille_1elt	157
5.26.15	sous_maille_1elt_selectif_mod	159
5.26.16	sous_maille_axi	160
5.26.17	sous_maille_smago_filtre	161
5.26.18	sous_maille_smago_dyn	162
5.26.19	mod_turb_hyd_rans	163
5.26.20	k_epsilon	164
5.26.21	modele_fonction_bas_reynolds_base	165
5.26.22	Launder_Sharma	165
5.26.23	Lam_Bremhorst	165
5.26.24	EASM_Baglietto	166
5.26.25	standard_KEps	166
5.26.26	Jones_Launder	166
5.26.27	K_Epsilon_Realisable	166
5.27	deuxmots	167
5.28	floatfloat	168
5.29	traitement_particulier	168
5.29.1	traitement_particulier_base	168

5.29.2	temperature	168
5.29.3	canal	169
5.29.4	ec	169
5.29.5	thi	170
5.29.6	thi_thermo	171
5.29.7	chmoy_faceperio	171
5.29.8	profils_thermo	172
5.29.9	brech	172
5.29.10	ceg	172
5.29.11	ceg_areva	173
5.29.12	ceg_cea_jaea	173
5.30	navier_stokes_phase_field	173
5.31	navier_stokes_qc	175
5.32	navier_stokes_standard	177
5.33	navier_stokes_turbulent	179
5.34	navier_stokes_turbulent_qc	180
5.35	transport_interfaces_ft_disc	182
5.36	methode_transport_deriv	186
5.36.1	loi_horaire	186
5.36.2	vitesse_imposee	186
5.36.3	vitesse_interpolee	187
5.37	bloc_lecture_remaillage	187
5.38	parcours_interface	188
5.39	interpolation_champ_face_deriv	189
5.39.1	base	189
5.39.2	lineaire	189
5.40	transport_k_epsilon	189
5.41	transport_marqueur_ft	190
5.42	injection_marqueur	192
6	algo_base	192
6.1	algo_couple_1	193
7	/*	193
7.1	/*	193
8	champ_generique_base	193
8.1	champ_post_de_champs_post	193
8.2	list_nom_virgule	194
8.3	listchamp_generique	194
8.4	champ_post_operateur_base	194
8.5	champ_post_operateur_eqn	195
8.6	champ_post_statistiques_base	195
8.7	correlation	196
8.8	champ_post_operateur_divergence	196
8.9	ecart_type	197
8.10	champ_post_extraction	197
8.11	champ_post_operateur_gradient	198
8.12	champ_post_interpolation	199
8.13	champ_post_morceau_equation	199
8.14	moyenne	200
8.15	predefini	201
8.16	champ_post_reduction_0d	201
8.17	champ_post_refchamp	202

8.18	champ_post_tparoi_vef	202
8.19	champ_post_transformation	203
9	chimie	204
9.1	reactions	204
9.1.1	reaction	204
10	class_generic	205
10.1	Modele_Fonc_Realisable	205
10.2	Modele_Fonc_Realisable_base	205
10.3	Modele_Shih_Zhu_Lumley_VDF	205
10.4	Shih_Zhu_Lumley	206
10.5	cholesky	206
10.6	dt_calc	206
10.7	dt_fixe	206
10.8	dt_min	207
10.9	dt_start	207
10.10	gcp_ns	207
10.11	gen	208
10.12	gmres	208
10.13	optimal	209
10.14	petsc	210
10.15	gcp	213
10.16	solveur_sys_base	214
11	#	214
11.1	#	214
12	condlim_base	215
12.1	Neumann_homogene	215
12.2	Neumann_paro_adiabatique	215
12.3	Paroi	215
12.4	contact_vdf_vef	215
12.5	contact_vef_vdf	216
12.6	dirichlet	216
12.7	echange_contact_rayo_transp_vdf	216
12.8	echange_contact_vdf_ft_disc	217
12.9	echange_contact_vdf_ft_disc_solid	217
12.10	entree_temperature_imposee_h	217
12.11	flux_radiatif	218
12.12	flux_radiatif_vdf	218
12.13	flux_radiatif_vef	218
12.14	frontiere_ouverte	219
12.15	frontiere_ouverte_concentration_imposee	219
12.16	frontiere_ouverte_fraction_massique_imposee	219
12.17	frontiere_ouverte_gradient_pression_imposee	219
12.18	frontiere_ouverte_gradient_pression_imposee_vefprep1b	220
12.19	frontiere_ouverte_gradient_pression_libre_vef	220
12.20	frontiere_ouverte_gradient_pression_libre_vefprep1b	220
12.21	frontiere_ouverte_k_eps_imposee	220
12.22	frontiere_ouverte_pression_imposee	220
12.23	frontiere_ouverte_pression_imposee_orlansky	221
12.24	frontiere_ouverte_pression_moyenne_imposee	221
12.25	frontiere_ouverte_rayo_semi_transp	221

12.26	frontiere_ouverte_rayo_transp	221
12.27	frontiere_ouverte_rayo_transp_vdf	222
12.28	frontiere_ouverte_rayo_transp_vef	222
12.29	frontiere_ouverte_rho_u_impose	222
12.30	frontiere_ouverte_temperature_imposee	223
12.31	frontiere_ouverte_temperature_imposee_rayo_semi_transp	223
12.32	frontiere_ouverte_temperature_imposee_rayo_transp	223
12.33	frontiere_ouverte_vitesse_imposee	223
12.34	frontiere_ouverte_vitesse_imposee_sortie	224
12.35	neumann	224
12.36	paroi_adiabatique	224
12.37	paroi_contact	224
12.38	paroi_contact_fictif	225
12.39	paroi_decalee_robin	225
12.40	paroi_defilante	226
12.41	paroi_echange_contact_correlation_vdf	226
12.42	paroi_echange_contact_correlation_vef	227
12.43	paroi_echange_contact_odvm_vdf	228
12.44	paroi_echange_contact_rayo_semi_transp_vdf	228
12.45	paroi_echange_contact_vdf	228
12.46	paroi_echange_contact_vdf_ft	229
12.47	paroi_echange_contact_vdf_zoom_fin	229
12.48	paroi_echange_contact_vdf_zoom_grossier	230
12.49	paroi_echange_externe_impose	230
12.50	paroi_echange_externe_impose_h	230
12.51	paroi_echange_externe_impose_rayo_semi_transp	231
12.52	paroi_echange_externe_impose_rayo_transp	231
12.53	paroi_echange_global_impose	231
12.54	paroi_fixe	231
12.55	paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets	232
12.56	paroi_flux_impose	232
12.57	paroi_flux_impose_rayo_semi_transp_vdf	232
12.58	paroi_flux_impose_rayo_semi_transp_vef	232
12.59	paroi_flux_impose_rayo_transp	233
12.60	paroi_ft_disc	233
12.61	paroi_ft_disc_deriv	233
12.61.1	symetrie	233
12.61.2	constant	233
12.62	paroi_knudsen_non_negligeable	234
12.63	paroi_rugueuse	234
12.64	paroi_temperature_imposee	234
12.65	paroi_temperature_imposee_rayo_semi_transp	235
12.66	paroi_temperature_imposee_rayo_transp	235
12.67	periodique	235
12.68	scalaire_impose_paro	235
12.69	sortie_libre_rho_variable	236
12.70	sortie_libre_temperature_imposee_h	236
12.71	symetrie	236
12.72	temperature_imposee_paro	236

13 discretisation_base	237
13.1 ef	237
13.2 polymac	237
13.3 vdf	237
13.4 vef	237
13.5 vefprep1b	237
14 domaine	238
14.1 domaine_ale	238
15 espece	238
16 champ_base	239
16.1 champ_base	239
16.2 Champ_Fonc_MED_Tabule	239
16.3 Champ_Fonc_MEDfile	239
16.4 Champ_Tabule_Morceaux	239
16.5 champ_don_base	240
16.6 champ_don_lu	240
16.7 champ_fonc_fonction	240
16.8 champ_fonc_fonction_txyz	241
16.9 champ_fonc_med	241
16.10champ_fonc_reprise	241
16.11fonction_champ_reprise	242
16.12champ_fonc_t	242
16.13champ_fonc_tabule	242
16.14champ_init_canal_sinal	243
16.15bloc_lec_champ_init_canal_sinal	243
16.16champ_input_base	244
16.17champ_input_p0	244
16.18champ_ostwald	245
16.19champ_som_lu_vdf	245
16.20champ_som_lu_vef	245
16.21champ_tabule_temps	246
16.22champ_uniforme_morceaux	246
16.23champ_uniforme_morceaux_tabule_temps	246
16.24champ_fonc_txyz	247
16.25champ_fonc_xyz	247
16.26field_uniform_keps_from_ud	247
16.27init_par_partie	248
16.28tayl_green	248
16.29uniform_field	248
16.30valeur_totale_sur_volume	248
17 champ_front_base	249
17.1 champ_front_base	249
17.2 Ch_front_input_ALE	249
17.3 Champ_front_ale	249
17.4 Champ_front_debit_QC_VDF	250
17.5 boundary_field_inward	250
17.6 boundary_field_uniform_keps_from_ud	250
17.7 ch_front_input	251
17.8 ch_front_input_uniforme	251
17.9 champ_front_MED	251

17.10	champ_front_bruite	252
17.11	champ_front_calc	252
17.12	champ_front_contact_rayo_semi_transp_vef	253
17.13	champ_front_contact_rayo_transp_vef	253
17.14	champ_front_contact_vef	253
17.15	champ_front_debit	254
17.16	champ_front_debit_massique	254
17.17	champ_front_fonc_pois_ipsn	254
17.18	champ_front_fonc_pois_tube	254
17.19	champ_front_fonc_t	255
17.20	champ_front_fonc_txyz	255
17.21	champ_front_fonc_xyz	255
17.22	champ_front_fonction	255
17.23	champ_front_lu	256
17.24	champ_front_normal_vef	256
17.25	champ_front_pression_from_u	256
17.26	champ_front_recyclage	257
17.27	champ_front_tabule	259
17.28	champ_front_tangentiel_vef	259
17.29	champ_front_uniforme	259
17.30	champ_front_vortex	259
17.31	champ_front_xyz_debit	260
17.32	champ_front_zoom	260
18	loi_etat_base	260
18.1	gaz_reel_rhot	261
18.2	melange_gaz_parfait	261
18.3	gaz_parfait	261
19	loi_fermeture_base	262
19.1	loi_fermeture_test	262
20	loi_horaire	262
21	milieu_base	263
21.1	constituant	263
21.2	fluide_diphasique	263
21.3	fluide_incompressible	264
21.4	fluide_ostwald	265
21.5	fluide_quasi_compressible	265
21.6	bloc_sutherland	266
21.7	solide	267
22	milieu_v2_base	267
23	modele_rayonnement_base	267
23.1	modele_rayonnement_milieu_transparent	267
24	modele_turbulence_scal_base	269
24.1	prandtl	269
24.2	schmidt	270
24.3	sous_maille_dyn	271
25	nom	271
25.1	nom_anonyme	272

26	partitionneur_deriv	272
26.1	fichier_decoupage	272
26.2	metis	273
26.3	partition	273
26.4	sous_domaine	274
26.5	sous_zones	274
26.6	tranche	275
26.7	union	275
27	precond_base	275
27.1	ilu	276
27.2	precondsolv	276
27.3	ssor	276
27.4	ssor_bloc	276
28	schema_temps_base	277
28.1	implicit_euler_steady_scheme	279
28.2	Sch_CN_EX_iteratif	281
28.3	Sch_CN_iteratif	283
28.4	scheme_euler_explicit	285
28.5	leap_frog	287
28.6	rk3_ft	288
28.7	runge_kutta_ordre_3	290
28.8	runge_kutta_ordre_4_d3p	292
28.9	runge_kutta_rationnel_ordre_2	294
28.10	schema_adams_bashforth_order_2	295
28.11	schema_adams_bashforth_order_3	297
28.12	schema_adams_moulton_order_2	299
28.13	schema_adams_moulton_order_3	301
28.14	schema_backward_differentiation_order_2	303
28.15	schema_backward_differentiation_order_3	306
28.16	scheme_euler_implicit	308
28.17	schema_implicite_base	310
28.18	schema_phase_field	312
28.19	schema_predictor_corrector	314
28.20	schema_euler_explicite_ALE	316
29	solveur_implicite_base	318
29.1	implicit_steady	318
29.2	implicite	319
29.3	implicite_ALE	320
29.4	piso	320
29.5	simple	321
29.6	simpler	322
29.7	solveur_lineaire_std	323
29.8	solveur_u_p	323
30	source_base	324
30.1	Source_Transport_K_Eps_anisotherme	324
30.2	acceleration	325
30.3	boussinesq_concentration	326
30.4	boussinesq_temperature	326
30.5	canal_perio	326
30.6	coriolis	327

30.7	darcy	327
30.8	dirac	328
30.9	forchheimer	328
30.10	perte_charge_anisotrope	328
30.11	perte_charge_circulaire	329
30.12	perte_charge_directionnelle	329
30.13	perte_charge_isotrope	329
30.14	perte_charge_reguliere	330
30.15	spec_pdc_base	330
30.15.1	longitudinale	330
30.15.2	transversale	331
30.16	perte_charge_singuliere	331
30.17	puissance_thermique	331
30.18	source_con_phase_field	332
30.19	source_constituant	333
30.20	flottabilite	333
30.21	source_generique	333
30.22	masse_ajoutee	334
30.23	source_qdm	334
30.24	source_qdm_lambdaup	334
30.25	source_qdm_phase_field	334
30.26	source_rayo_semi_transp	335
30.27	source_robin	335
30.28	source_robin_scalaire	335
30.29	listdeuxmots_sacc	336
30.30	source_th_tdivu	336
30.31	trainee	336
30.32	source_transport_k_eps	336
30.33	source_transport_k_eps_aniso_concen	337
30.34	source_transport_k_eps_aniso_therm_concen	337
31	sous_zone	337
31.1	bloc_origine_cotes	338
31.2	bloc_couronne	338
31.3	bloc_tube	339
32	turbulence_paro_base	339
32.1	loi_ciofalo_hydr	339
32.2	loi_expert_hydr	340
32.3	loi_puissance_hydr	340
32.4	loi_standard_hydr	340
32.5	loi_standard_hydr_old	341
32.6	loi_ww_hydr	341
32.7	negligeable	341
32.8	paroi_tble	341
32.9	twofloat	342
32.10	liste_sonde_tble	342
32.10.1	sonde_tble	342
32.11	entierfloat	342
32.12	utau_imp	343

33	turbulence_paroil_scalaire_base	343
33.1	loi_WW_scalaire	343
33.2	loi_analytique_scalaire	344
33.3	loi_expert_scalaire	344
33.4	loi_odvm	344
33.5	loi_paroil_nu_impose	345
33.6	loi_standard_hydr_scalaire	345
33.7	negligeable_scalaire	345
33.8	paroi_tble_scal	345
33.9	fourfloat	346
34	listobj_impl	346
34.1	list_un_pb	346
34.2	un_pb	347
34.3	listobj	347
35	objet_lecture	347
36	index	348

1 Syntax to define a mathematical function

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

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

You can also use the following operations:

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

You can also use the following constants:

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

The variables which can be used are:

x,y,z : coordinates
 t : time

Examples:

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

Possible errors:

Error 1:

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

Error 2:

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

2 Existing & predefined fields names

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

Physical values	Keyword for field_name	Unit
Velocity	Vitesse or Velocity	$m.s^{-1}$
Kinetic energy per elements ($0.5\rho u_i ^2$)	Energie_cinetique_elem	$kg.m^{-1}.s^{-2}$
Total kinetic energy ($\frac{\sum_{i=1}^{nb_elem} 0.5\rho u_i ^2 vol_i}{\sum_{i=1}^{nb_elem} vol_i}$)	Energie_cinetique_totale	$kg.m^{-1}.s^{-2}$
Vorticity	Vorticite	s^{-1}
... continued on next page ...		

Physical values	Keyword for field_name	Unit
Pressure in incompressible flow ($P/\rho + gz$) For Front Tracking probleme ($P + \rho gz$)	Pression ¹	$Pa.m^3.kg^{-1}$ or Pa
Pressure in incompressible flow ($P + \rho gz$)	Pression_pa or Pressure	Pa
Pressure in compressible flow	Pression	Pa
Hydrostatic pressure (ρgz)	Pression_hydrostatique	Pa
Totale pressure (when quasi compressible model is used)=Pth+P	Pression_tot	Pa
Pressure gradient ($\nabla(P/\rho + gz)$)	Gradient_pression	$m.s^{-2}$
Velocity gradient	gradient_vitesse	s^{-1}
Temperature	Temperature	$^{\circ}C$ 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 ²	$W.m^{-2}.K^{-1}$
Turbulent heat flux	Flux_Chaleur_Turbulente	$m.K.s^{-1}$
Turbulent viscosity	Viscosite_turbulente	$m^2.s^{-1}$
Turbulent dynamic viscosity (when quasi compressible model is used)	Viscosite_dynamique_turbulente	$kg.m.s^{-1}$
Turbulent kinetic energy	K	$m^2.s^{-2}$
Turbulent dissipation rate	Eps	$m^3.s^{-1}$
Turbulent quantities K and Epsilon	K_Eps	$(m^2.s^{-2}, m^3.s^{-1})$
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	$m.s^{-1}$
Cell volumes	Volume_maille	m^3
Chemical potential	Potentiel_Chimique_Generalise	
Source term in non		
... continued on next page ...		

¹The 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.

²Tref 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
Galilean referential	Acceleration_terme_source	$m.s^{-2}$
Stability time steps	Pas_de_temps	S
Listing of boundary fluxes	Flux_bords	cf each *.out file
Volumetric porosity	Porosite_volumique	dimensionless
Distance to the wall	Distance_Paroi ³	m
Volumic thermal power	Puissance_volumique	$W.m^{-3}$
Local shear strain rate defined as $\sqrt{(2S_{ij}S_{ij})}$	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 (36) read (3.73) associate (3.8) discretize (3.26) mailler (3.55) maillerparallel (3.57) ecrire_fichier_bin (3.115) ecrire (3.114) read_file (3.74) lire_tgrid (3.76) solve (3.93) execute_parallel (3.31) end (3.44) dimension (3.23) bidim_axi (3.13) axi (3.12) transformer (3.105) rotation (3.89) dilate (3.22) testeur (3.98) test_solveur (3.97) postraiter_domaine (3.69) modif_bord_to_raccord (3.58) remove_elem (3.82) regroupebord (3.81) supprime_bord (3.94) calculer_moments (3.14) imprimer_flux (3.47) decouper_bord_coincident (3.21) raffiner_anisotrope (3.71) raffiner_isotrope (3.72) trianguler (3.106) tetraedriser (3.100) orientefacesbord (3.62) reorienter_tetraedres (3.86) reorienter_triangles (3.87) verifiercoin (3.112) porosites (3.66) porosites_champ (3.68) discretiser_domaine (3.25) { (3.19) } (3.45) export (3.32) debog (3.18) pilote_icoco (3.65) moyenne_volumique (3.59) ecrire_champ_med (3.28) read_med (3.3) lire_ideas (3.54) ecrire_med (3.116) system (3.96) redresser_hexaedres_vdf (3.79) analyse_angle (3.7) remove_invalid_internal_boundaries (3.84) reordonner (3.88) precisiongeom (3.70) nettoiepasnoeuds (3.60) scatter (3.90) partition (3.63) reordonner_faces_periodiques (3.85) corriger_frontiere_periodique (3.16) distance_parois (3.27) extruder (3.40) extract_2d_from_3d (3.33) extruder_en20 (3.42) extrudeparois (3.39) ecriturelecturespecial (3.30) lata_to_med (3.51) lata_to_other (3.53) decoupebord_pour_rayonnement (3.20) extraire_plan (3.36) extraire_domaine (3.35) extraire_surface (3.37) integrer_champ_med (3.49) orienter_simplexes (3.78) verifier_simplexes (3.111) verifier_qualite_raffinements (3.109) testeur_medcoupling (3.99) Raffiner_isotrope_parallele (3.2) option_vdf (3.61) interpreter_geometrique_base (3.50) extrudebord (3.38) disable_TU (3.24) refine_mesh (3.80) Op_Conv_EF_Stab_PolyMAC_Face (3.1) Solver_moving_mesh_ALE (3.5) imposer_vit_bords_ale (3.46)

Usage:

interpret

3.1 Op_Conv_EF_Stab_PolyMAC_Face

Description: Class Op_Conv_EF_Stab_PolyMAC_Face_PolyMAC

See also: interpret (3)

Usage:

Op_Conv_EF_Stab_PolyMAC_Face {

[**alpha** float]

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

}
where

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

3.2 Raffiner_isotrope_parallelele

Description: Refine parallel mesh in parallel

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

Usage:

```
Raffiner_isotrope_parallelele {  
    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

3.3 read_med

Synonymous: **lire_med**

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

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

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

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

Read_Med

Read_file domain_name_ssz.geo ;

During the parallel calculation, you will include something:

Scatter { ... }

Read_file domain_name_ssz_par.geo ;

See also: [interpret \(3\)](#) [lire_medfile \(3.4\)](#)

Usage:

```
read_med [ vef ] [ family_names_from_group_names ] [ short_family_names ] nom_dom nom-  
_dom_med file  
where
```

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

3.4 lire_medfile

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

See also: read_med (3.3)

Usage:

lire_medfile [**vef**] [**family_names_from_group_names**] [**short_family_names**] **nom_dom** **nom_dom_med** **file**
 where

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

3.5 Solver_moving_mesh_ALE

Description: Solver used to solve the system giving the mesh velocity for the ALE (Arbitrary Lagrangian-Eulerian) framework.

See also: interpret (3)

Usage:

Solver_moving_mesh_ALE **dom** **bloc**
 where

- **dom** *str*: Name of domain.
- **bloc** *bloc_lecture* (3.6): Example: { PETSC GCP { precondition ssor { omega 1.5 } seuil 1e-7 impr } }

3.6 bloc_lecture

Description: to read between two braces

See also: objet_lecture (35)

Usage:

bloc_lecture

where

- **bloc_lecture** *str*

3.7 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*

3.8 associate

Synonymous: **associer**

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

See also: interpret (3) associer_pbm_glb (3.11) associer_pbm_glbfin (3.10) associer_algo (3.9)

Usage:

associate objet_1 objet_2

where

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

3.9 associer_algo

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

See also: associate (3.8)

Usage:

associer_algo objet_1 objet_2

where

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

3.10 associer_pbmng_pbfin

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

See also: [associate \(3.8\)](#)

Usage:

associer_pbmng_pbfin **objet_1** **objet_2**

where

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

3.11 associer_pbmng_pbgglobal

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

See also: [associate \(3.8\)](#)

Usage:

associer_pbmng_pbgglobal **objet_1** **objet_2**

where

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

3.12 axi

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

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

Usage:

axi

3.13 bidim_axi

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

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

Usage:

bidim_axi

3.14 calculer_moments

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

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

Usage:

calculer_moments nom_dom mot

where

- **nom_dom** *str*: Name of domain.
- **mot** *lecture_bloc_moment_base (3.15)*: Keyword.

3.15 lecture_bloc_moment_base

Description: Auxiliary class to compute and print the moments.

See also: [objet_lecture \(35\)](#) [calcul \(3.15.1\)](#) [centre_de_gravite \(3.15.2\)](#)

Usage:

3.15.1 calcul

Description: The centre of gravity will be calculated.

See also: [\(3.15\)](#)

Usage:

calcul

3.15.2 centre_de_gravite

Description: To specify the centre of gravity.

See also: [\(3.15\)](#)

Usage:

centre_de_gravite point

where

- **point** *un_point (3.15.3)*: A centre of gravity.

3.15.3 un_point

Description: A point.

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

Usage:

pos

where

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

3.16 corriger_frontiere_periodique

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

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

Usage:

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

where

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

3.17 create_domain_from_sous_zone

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

See also: [interpret_geometrique_base \(3.50\)](#)

Usage:

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

where

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

3.18 debug

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

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

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

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

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

Usage:

debug pb fichier1 fichier2 seuil mode

where

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

3.19 {

Description: Block's beginning.

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

Usage:

{

3.20 decoupebord_pour_rayonnement

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

A mesh file (ASCII format, except if `binaire` option is specified) named by default `newgeom` (or specified by the `nom_fichier_sortie` keyword) will be created and will contain the fine_domain_name domain with the splitted boundaries named `boundary_name`

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

Usage:

decoupebord_pour_rayonnement {

domaine *str*

```

[ domaine_grossier str]
[ nb_parts_naif n n1 n2 ... nn]
[ nb_parts_geom n n1 n2 ... nn]
bords_a_decouper n word1 word2 ... wordn
[ nom_fichier_sortie str]
[ condition_geometrique n word1 word2 ... wordn]
[ binaire int]
}
where

```

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

3.21 decouper_bord_coincident

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

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

Usage:

```
decouper_bord_coincident domain_name bord
where
```

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

3.22 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.

3.23 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.

3.24 disable_TU

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

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

Usage:

disable_TU

3.25 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.

3.26 discretize

Synonymous: **discretiser**

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

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

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

Usage:

discretize problem_name dis

where

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

3.27 distance_pari

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

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

Usage:

distance_pari dom bords format

where

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

3.28 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

3.29 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 \(3.115\)](#)

Usage:

ecrire_fichier_formatte name_obj filename

where

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

3.30 ecriturelecturespecial

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

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

Usage:

ecriturelecturespecial type

where

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

3.31 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.

3.32 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

3.33 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](#) (3.34)

Usage:

```
extract_2d_from_3d dom3D bord dom2D
```

where

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

3.34 extract_2daxi_from_3d

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

See also: `extract_2d_from_3d` ([3.33](#))

Usage:

extract_2daxi_from_3d **dom3D** **bord** **dom2D**

where

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

3.35 extraire_domaine

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

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

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

Usage:

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

where

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

3.36 extraire_plan

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

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

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

Usage:

```
extraire_plan {  
    domaine str  
    probleme str  
    epaisseur float  
    origine n x1 x2 ... xn  
    point1 n x1 x2 ... xn  
    point2 n x1 x2 ... xn  
    [ point3 n x1 x2 ... xn ]  
    [ triangle ]  
    [ via_extraire_surface ]  
    [ inverse_condition_element ]  
    [ avec_certains_bords_pour_extraire_surface n word1 word2 ... wordn ]  
}
```

where

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

3.37 **extraire_surface**

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

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

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

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

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

3.38 extrudebord

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

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

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

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

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

Usage:

```

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

```

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

3.39 extrudeparoi

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

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

Usage:

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

where

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

3.40 extruder

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

See also: [interprete \(3\)](#) [extruder_en3 \(3.43\)](#)

Usage:

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

where

- **domaine** *str*: Name of the domain.
- **direction** *troisf* [\(3.41\)](#): Direction of the extrude operation.
- **nb_tranches** *int*: Number of elements in the extrusion direction.

3.41 troisf

Description: Auxiliary class to extrude.

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

Usage:

```
lx ly lz  
where
```

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

3.42 extruder_en20

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

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

Usage:

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

where

- **domaine** *str*: Name of the domain.
- **direction** *troisf* [\(3.41\)](#): 0 Direction of the extrude operation.
- **nb_tranches** *int*: Number of elements in the extrusion direction.

3.43 extruder_en3

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

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

See also: [extruder \(3.40\)](#)

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* [\(3.41\)](#) for inheritance: Direction of the extrude operation.
- **nb_tranches** *int* for inheritance: Number of elements in the extrusion direction.

3.44 end

Synonymous: **fin**

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

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

Usage:

end

3.45 }

Description: Block's end.

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

Usage:

}

3.46 imposer_vit_bords_ale

Description: For the Arbitrary Lagrangian-Eulerian framework: block to indicate the number of mobile boundaries of the domain and specify the speed that must be imposed on them.

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

Usage:

imposer_vit_bords_ale dom bloc

where

- **dom** *str*: Name of domain.
- **bloc** *bloc_lecture* (3.6): between the braces, you must specify the numbers of the mobile borders of the domain then list these mobile borders and indicate the speed which must be imposed on them
Example: `Imposer_vit_bords_ALE dom_name { 1 boundary_name Champ_front_ALE 2 -(y-0.1)*0.01 (x-0.1)*0.01 }`

3.47 imprimer_flux

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

See also: [interpret](#) (3) [imprimer_flux_sum](#) (3.48)

Usage:

imprimer_flux domain_name noms_bord

where

- **domain_name** *str*: Name of the domain.
- **noms_bord** *bloc_lecture* (3.6): List of boundaries, for ex: { Bord1 Bord2 }

3.48 imprimer_flux_sum

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

See also: imprimer_flux ([3.47](#))

Usage:

imprimer_flux_sum **domain_name** **noms_bord**

where

- **domain_name** *str*: Name of the domain.
- **noms_bord** *bloc_lecture* ([3.6](#)): List of boundaries, for ex: { Bord1 Bord2 }

3.49 integrer_champ_med

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

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

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

Usage:

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

where

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

3.50 interprete_geometrique_base

Description: Class for interpreting a data file

See also: interprete ([3](#)) create_domain_from_sous_zone ([3.17](#))

Usage:

interprete_geometrique_base

3.51 lata_to_med

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

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

Usage:

lata_to_med [**format**] **file** **file_med**

where

- **format** *format_lata_to_med* (3.52): generated file post_med.data use format (MED or LATA or LML keyword).
- **file** *str*: LATA file to convert to the new format.
- **file_med** *str*: Name of the MED file.

3.52 format_lata_to_med

Description: not_set

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

Usage:

mot [**format**]

where

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

3.53 lata_to_other

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

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

Usage:

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

where

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

3.54 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.

3.55 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* ([3.56](#)): Instructions to mesh.

3.56 list_bloc_mailler

Description: List of block mesh.

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

Usage:

{ *object1* , *object2* }

list of *mailler_base* ([3.56.1](#)) separated with ,

3.56.1 mailler_base

Description: Basic class to mesh.

See also: [objet_lecture \(35\)](#) [pave \(3.56.2\)](#) [epsilon \(3.56.12\)](#) [domain \(3.56.13\)](#)

Usage:

3.56.2 pave

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

See also: [mailler_base \(3.56.1\)](#)

Usage:

pave name bloc list_bord

where

- **name** *str*: Name of the pave (block).
- **bloc** *bloc_pave* ([3.56.3](#)): Definition of the pave (block).
- **list_bord** *list_bord* ([3.56.4](#)): Domain boundaries definition.

3.56.3 bloc_pave

Description: Class to create a pave.

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

Usage:

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

where

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

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

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

3.56.4 list_bord

Description: The block sides.

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

Usage:

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

list of *bord_base* ([3.56.5](#))

3.56.5 bord_base

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

See also: objet_lecture ([35](#)) bord ([3.56.6](#)) raccord ([3.56.10](#)) internes ([3.56.11](#))

Usage:

3.56.6 bord

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

See also: bord_base ([3.56.5](#))

Usage:

bord nom defbord

where

- **nom** *str*: Name of block side.
- **defbord** *defbord* ([3.56.7](#)): Definition of block side.

3.56.7 defbord

Description: Class to define an edge.

See also: objet_lecture ([35](#)) defbord_2 ([3.56.8](#)) defbord_3 ([3.56.9](#))

Usage:

3.56.8 defbord_2

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

See also: (3.56.7)

Usage:

dir eq pos pos2_min inf1 dir2 inf2 pos2_max

where

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

3.56.9 defbord_3

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

See also: (3.56.7)

Usage:

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

where

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

3.56.10 raccord

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

See also: bord_base (3.56.5)

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* (3.56.7): Definition of block side.

3.56.11 internes

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

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

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

See also: `bord_base` (3.56.5)

Usage:

internes nom defbord

where

- **nom** *str*: Name of block side.
- **defbord** *defbord* (3.56.7): Definition of block side.

3.56.12 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` (3.56.1)

Usage:

epsilon eps

where

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

3.56.13 domain

Description: Class to reuse a domain.

See also: `mailler_base` (3.56.1)

Usage:

domain domain_name

where

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

3.57 maillerparallel

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:

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

where

- **domain** *str*: the name of the domain to mesh (it must be an empty domain object).
- **nb_nodes** *n n1 n2 ... nn*: dimension defines the spatial dimension (currently only dimension=3 is supported), and nX, nY and nZ defines the total number of nodes in the mesh in each direction.
- **splitting** *n n1 n2 ... nn*: dimension is the spatial dimension and npartsX, npartsY and npartsZ are the number of parts created. The product of the number of parts must be equal to the number of processors used for the computation.
- **ghost_thickness** *int*: 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*

3.58 **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.

3.59 **moyenne_volumique**

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

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

Usage:

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

```
}
```

where

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

- **localisation** *str* into [*'elem'*, *'som'*]: indicates where the convolution product should be computed: either on the elements or on the nodes of the destination domain.
- **function_filtre** *bloc_lecture* (3.6): 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.

3.60 nettoiepasnoeuds

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

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

Usage:

nettoiepasnoeuds **domain_name**

where

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

3.61 option_vdf

Description: Class of VDF options.

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

Usage:

option_vdf {

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

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

}
where

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

3.62 orientefacesbord

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

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

Usage:

orientefacesbord **domain_name**
where

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

3.63 partition

Synonymous: **decouper**

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

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

Usage:

partition **domaine** **bloc_decouper**
where

- **domaine** *str*: Name of the domain to be cut.
- **bloc_decouper** *bloc_decouper (3.64)*: Description how to cut a domain.

3.64 bloc_decouper

Description: Auxiliary class to cut a domain.

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

Usage:

```
{  
    [ Partition_tool|partitionneur partitionneur_deriv]  
    [ larg_joint int]  
    [ zones_name|nom_zones str]  
    [ ecrire_decoupage str]  
    [ ecrire_lata str]  
    [ nb_parts_tot int]  
    [ formatte ]  
}
```

```

[ periodique n word1 word2 ... wordn]
[ reorder int]
}
where

```

- **Partition_toolpartitionneur** *partitionneur_deriv* (26): Defines the partitionning algorithm (the effective C++ object used is 'Partitionneur_ALGORITHM_NAME').
- **larg_joint** *int*: This keyword specifies the thickness of the virtual ghost zone (data known by one processor though not owned by it). The default value is 1 and is generally correct for all algorithms except the QUICK convection scheme that require a thickness of 2. Since the 1.5.5 version, the VEF discretization imply also a thickness of 2 (except VEF P0). Any non-zero positive value can be used, but the amount of data to store and exchange between processors grows quickly with the thickness.
- **zones_namelnom_zones** *str*: Name of the files containing the different partition of the domain. The files will be :
name_0001.Zones
name_0002.Zones
...
name_000n.Zones. If this keyword is not specified, the geometry is not written on disk (you might just want to generate a 'ecrire_decoupage' or 'ecrire_lata').
- **ecrire_decoupage** *str*: After having called the partitionning algorithm, the resulting partition is written on disk in the specified filename. See also partitionneur Fichier_Decoupage. This keyword is useful to change the partition numbers: first, you write the partition into a file with the option *ecrire_decoupage*. This file contains the zone number for each element's mesh. Then you can easily permute zone numbers in this file. Then read the new partition to create the .Zones files with the Fichier_Decoupage keyword.
- **ecrire_lata** *str*
- **nb_parts_tot** *int*: Keyword to generates N .Zone files, instead of the default number M obtained after the partitionning algorithm. N must be greater or equal to M. This option might be used to perform coupled parallel computations. Supplemental empty zones from M to N-1 are created. This keyword is used when you want to run a parallel calculation on several domains with for example, 2 processors on a first domain and 10 on the second domain because the first domain is very small compare to second one. You will write Nb_parts 2 and Nb_parts_tot 10 for the first domain and Nb_parts 10 for the second domain.
- **formatte** : Optional keyword to have formatted format for .Zones files. By default, it is binary format.
- **periodique** *n word1 word2 ... wordn*: N BOUNDARY_NAME_1 BOUNDARY_NAME_2 ... : N is the number of boundary names given. Periodic boundaries must be declared by this method. The partitionning algorithm will ensure that facing nodes and faces in the periodic boundaries are located on the same processor.
- **reorder** *int*: If this option is set to 1 (0 by default), the partition is renumbered in order that the processes which communicate the most are nearer on the network. This may slightly improves parallel performance.

3.65 pilote_icoco

Description: not_set

See also: interpret (3)

Usage:

```

pilote_icoco {
    pb_name str

```

```

    main str
}
where

• pb_name str
• main str

```

3.66 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 ([3.67](#)): Surface and volume porosity values.

3.67 bloc_lecture_poro

Description: Surface and volume porosity values.

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

Usage:

```

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

```

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

3.68 porosites_champ

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

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

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

Usage:

porosites_champ pb ch
where

- **pb** *str*: Name of the problem to which the sub-area is attached.
- **ch** *champ_base* ([16.1](#)): field used to define the porosity field

3.69 postraiter_domaine

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

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

Usage:

postraiter_domaine {
 format *str* into ['lml', 'lata', 'lata_v1', 'lata_v2', 'med']
 [**filefichier** *str*]
 [**domaine** *str*]
 [**domaines** *bloc_lecture*]
 [**joints_non_postraites** *int* into [0, 1]]
 [**binaire** *int* into [0, 1]]
 [**ecrire_frontiere** *int* into [0, 1]]
}

where

- **format** *str* into ['lml', 'lata', 'lata_v1', 'lata_v2', 'med']: File format.
- **filefichier** *str*: The file name can be changed with the fichier option.
- **domaine** *str*: Name of domain
- **domaines** *bloc_lecture* ([3.6](#)): 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)

3.70 precisiongeom

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

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

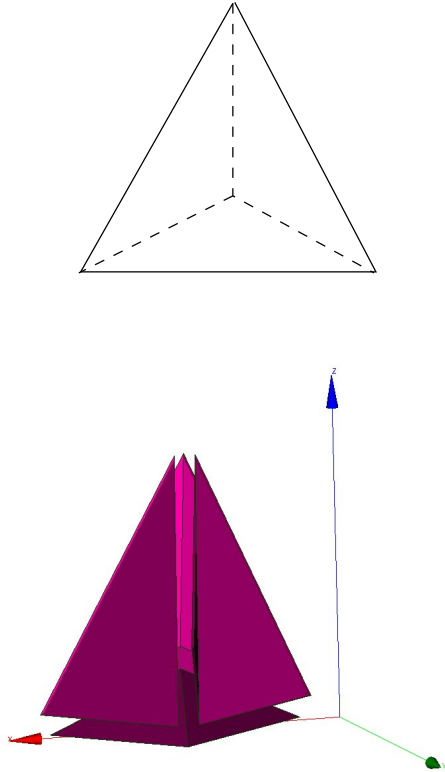
Usage:

precisiongeom precision
where

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

3.71 raffiner_anisotrope

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



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

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

Usage:

raffiner_anisotrope **domain_name**

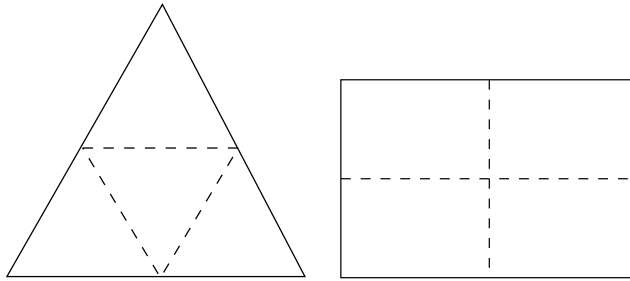
where

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

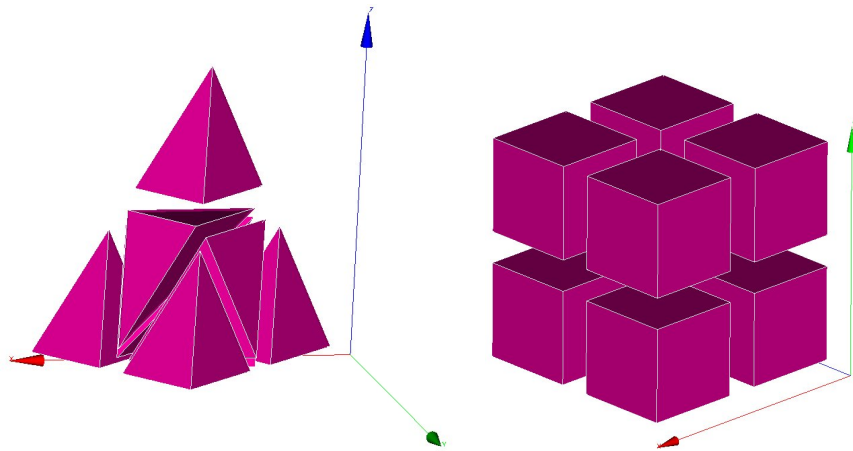
3.72 raffiner_isotrope

Synonymous: **raffiner_simplexes**

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



For 3D elements:



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

Usage:

raffiner_isotrope domain_name

where

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

3.73 read

Synonymous: **lire**

Description: Interpreter to read the **a_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.

3.74 read_file

Synonymous: **lire_fichier**

Description: Keyword to read the object name_obj contained in the file filename.

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

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

See also: interpret (3) read_unsupported_ascii_file_from_icem (3.77) read_file_binary (3.75)

Usage:

read_file name_obj filename

where

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

3.75 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 (3.74)

Usage:

read_file_binary name_obj filename

where

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

3.76 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.

3.77 read_unsupported_ascii_file_from_icem

Description: not_set

See also: read_file (3.74)

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.

3.78 **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.

3.79 **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.

3.80 **refine_mesh**

Description: not_set

See also: interpret (3)

Usage:

refine_mesh **domaine**

where

- **domaine** *str*

3.81 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* [\(3.6\)](#): { Bound1 Bound2 }

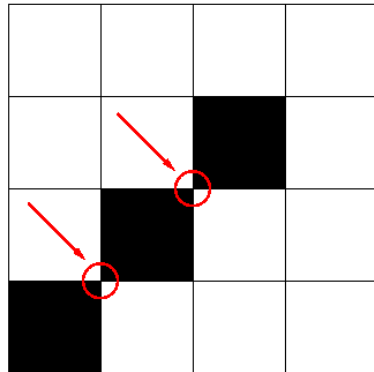
3.82 remove_elem

Description: Keyword to remove element from a VDF mesh (named domaine_name), either from an explicit list of elements or from a geometric condition defined by a condition $f(x,y)>0$ in 2D and $f(x,y,z)>0$ in 3D. All the new borders generated are gathered in one boundary called : newBord (to rename it, use RegroupeBord keyword). To split it to different boundaries, use DecoupeBord_Pour_Rayonnement keyword). Example of a removed zone of radius 0.2 centered at $(x,y)=(0.5,0.5)$:

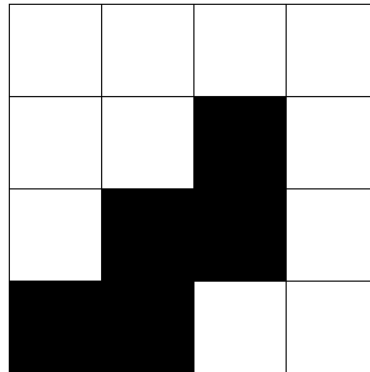
Remove_elem dom { fonction $0.2 * 0.2 - (x - 0.5)^2 - (y - 0.5)^2 > 0$ }

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

UNCORRECT – 2 SINGULAR NODES



CORRECT



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

Usage:

remove_elem **domaine** **bloc**

where

- **domaine** *str*: Name of domain
- **bloc** *remove_elem_bloc* [\(3.83\)](#)

3.83 remove_elem_bloc

Description: not_set

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

Usage:

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

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

3.84 **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.

3.85 **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.

3.86 **reorienter_tetraedres**

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

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

Usage:

```
reorienter_tetraedres  domain_name  
where
```

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

3.87 reorienter_triangles

Description: not_set

See also: interpreté (3)

Usage:

reorienter_triangles domain_name

where

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

3.88 reordonner

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

Read_file dom fichier.geom

Reordonner dom

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

See also: interpreté (3)

Usage:

reordonner domain_name

where

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

3.89 rotation

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

See also: interpreté (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)

3.90 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 \(3.91\)](#) [scattermed \(3.92\)](#)

Usage:

scatter file domaine

where

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

3.91 scatterformatte

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

See also: [scatter \(3.90\)](#)

Usage:

scatterformatte file domaine

where

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

3.92 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 \(3.90\)](#)

Usage:

scattermed file domaine

where

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

3.93 solve

Synonymous: **resoudre**

Description: Interpreter to start calculation with TRUST.

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

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

Usage:

solve pb

where

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

3.94 **supprime_bord**

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

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

Usage:

supprime_bord **domaine** **bords**

where

- **domaine** *str*: Name of domain
- **bords** *list_nom* ([3.95](#)): { Boundary_name1 Boundaray_name2 }

3.95 **list_nom**

Description: List of name.

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

Usage:

{ object1 object2 }

list of *nom_anonyme* ([25.1](#))

3.96 **system**

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

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

Usage:

system **cmd**

where

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

3.97 **test_solveur**

Description: To test several solvers

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

Usage:

test_solveur {

[**fichier_secmem** *str*]

[**fichier_matrice** *str*]

[**fichier_solution** *str*]

[**nb_test** *int*]

[**impr**]

[**solveur** *solveur_sys_base*]

[**fichier_solveur** *str*]

[**genere_fichier_solveur** *float*]

```

[ seuil_verification float ]
[ pas_de_solution_initiale ]
[ ascii ]
}
where

```

- **fichier_secmem** *str*: Filename containing the second member B
- **fichier_matrice** *str*: Filename containing the matrix A
- **fichier_solution** *str*: Filename containing the solution x
- **nb_test** *int*: Number of tests to measure the time resolution (one preconditionnement)
- **impr** : To print the convergence solver
- **solveur** *solveur_sys_base* (10.16): 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

3.98 testeur

Description: not_set

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

Usage:

testeur data

where

- **data** *bloc_lecture* (3.6)

3.99 testeur_medcoupling

Description: not_set

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

Usage:

testeur_medcoupling pb_name field_name

where

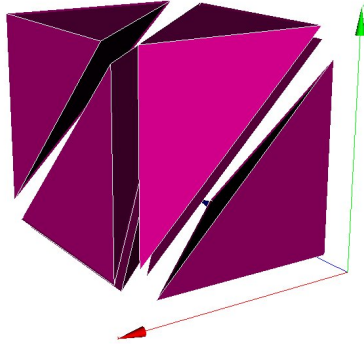
- **pb_name** *str*: Name of domain.
- **field_name** *str*: Name of domain.

3.100 tetraedriser

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

See also: [interprete \(3\)](#) [tetraedriser_homogene \(3.101\)](#) [tetraedriser_homogene_fin \(3.103\)](#) [tetraedriser_homogene_compact \(3.102\)](#) [tetraedriser_par_prisme \(3.104\)](#)

Usage:



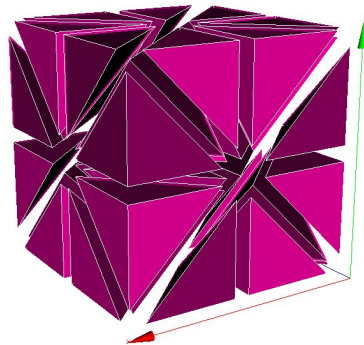
tetraedriser domain_name

where

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

3.101 tetraedriser_homogene

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



See also: tetraedriser ([3.100](#))

Usage:

tetraedriser_homogene domain_name

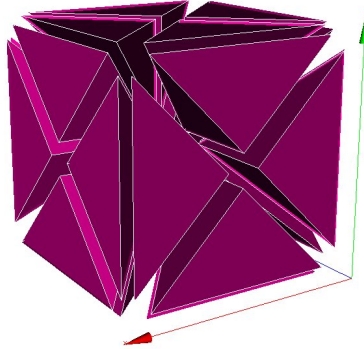
where

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

3.102 tetraedriser_homogene_compact

Description: This new discretization generates tetrahedral elements from cartesian or non-cartesian hexahedral elements. The process cut each hexahedral in 6 pyramids, each of them being cut then in 4 tetrahedral.

So, in comparison with `tetra_homogene`, less elements (*24 instead of*40) with more homogeneous volumes are generated. Moreover, this process is done in a faster way. Initial block is divided in 24 tetrahedra:



See also: `tetraedriser` ([3.100](#))

Usage:

`tetraedriser_homogene_compact` **`domain_name`**

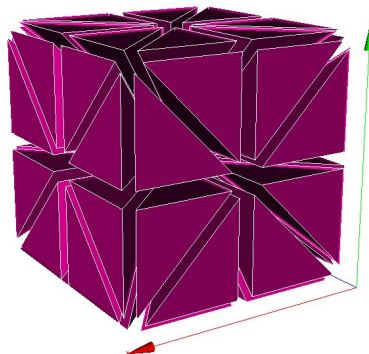
where

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

3.103 `tetraedriser_homogene_fin`

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

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



See also: `tetraedriser` ([3.100](#))

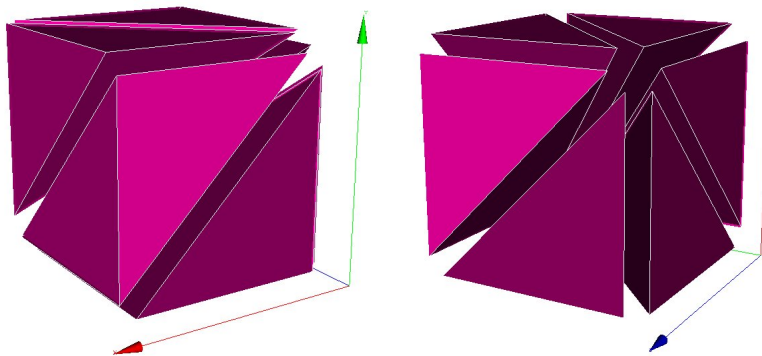
Usage:

tetraedriser_homogene_fin **domain_name**
 where

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

3.104 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 ([3.100](#))

Usage:

tetraedriser_par_prisme **domain_name**
 where

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

3.105 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

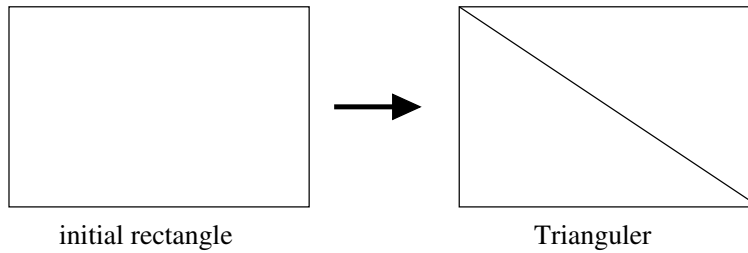
See also: interpret ([3](#))

Usage:

transformer **domain_name** **formule**
 where

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

Function_forz



3.106 **triangler**

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) [triangler_h \(3.108\)](#) [triangler_fin \(3.107\)](#)

Usage:

triangler domain_name

where

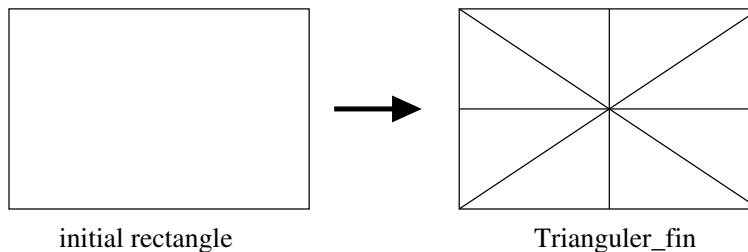
- **domain_name** *str*: Name of domain.

3.107 **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 discretization 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 realize statistical analysis in plane channel configuration for instance). Principle:



See also: [triangler \(3.106\)](#)

Usage:

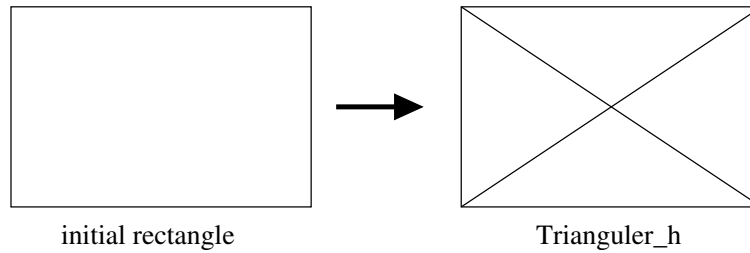
triangler_fin domain_name

where

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

3.108 **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** ([3.106](#))

Usage:

triangler_h **domain_name**

where

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

3.109 **verifier_qualite_raffinements**

Description: **not_set**

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

Usage:

verifier_qualite_raffinements **domain_names**

where

- **domain_names** *vect_nom* ([3.110](#))

3.110 **vect_nom**

Description: Vect of name.

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

Usage:

n object1 object2

list of *nom_anonyme* ([25.1](#))

3.111 **verifier_simplexes**

Description: Keyword to raffine a simplexes

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

Usage:

verifier_simplexes domain_name

where

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

3.112 verifiercoin

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

The expert_only option deactivates, into the VEFPreP1B divergence operator, the test of inconsistent cells.

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

Usage:

verifiercoin domain_name bloc

where

- **domain_name** *str*: Name of the domaine
- **bloc** *verifiercoin_bloc* ([3.113](#))

3.113 verifiercoin_bloc

Description: not_set

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

Usage:

```
{  
    [ Lire_fichier|Read_file str ]  
    [ expert_only ]  
}
```

where

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

3.114 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.

3.115 **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](#) (3.29)

Usage:

ecrire_fichier_bin name_obj filename

where

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

3.116 **ecrire_med**

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

See also: [interpret](#) (3) [ecrire_medfile](#) (3.117)

Usage:

ecrire_med nom_dom file

where

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

3.117 **ecrire_medfile**

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

See also: [ecrire_med](#) (3.116)

Usage:

ecrire_medfile nom_dom file

where

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

4 **pb_gen_base**

Description: Basic class for problems.

See also: [objet_u](#) (36) [Pb_base](#) (4.1) [probleme_couple](#) (4.7) [pbc_med](#) (4.38) [pb_mg](#) (4.23)

Usage:

4.1 Pb_base

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

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

See also: `pb_gen_base` (4) `pb_thermohydraulique` (4.26) `pb_hydraulique` (4.16) `pb_conduction` (4.13) `pb_thermohydraulique_qc` (4.31) `pb_hydraulique_concentration` (4.18) `pb_thermohydraulique_concentration` (4.27) `pb_avec_passif` (4.11) `pb_post` (4.25) `problem_read_generic` (4.40) `pb_conduction_milieu_variable` (4.14) `pb_hydraulique_turbulent` (4.22) `pb_thermohydraulique_turbulent` (4.34) `pb_hydraulique_concentration_turbulent` (4.20) `pb_thermohydraulique_concentration_turbulent` (4.29) `pb_thermohydraulique_turbulent_qc` (4.35) `pb_phase_field` (4.24) `modele_rayo_semi_transp` (4.9) `pb_hydraulique_ALE` (4.17)

Usage:

```
Pb_base obj Lire obj {  
    [ Post_processing|postraitement corps_postraitement]  
    [ Post_processings|postraitements post_processings]  
    [ liste_de_postraitements liste_post_ok]  
    [ liste_postraitements liste_post]  
    [ sauvegarde format_file]  
    [ sauvegarde_simple format_file]  
    [ reprise format_file]  
    [ resume_last_time format_file]  
}
```

where

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

4.2 corps_postraitement

Description: not_set

See also: post_processing ([4.4.3](#))

Usage:

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

where

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

4.2.1 definition_champs

Description: List of definition champ

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

Usage:

```
{ object1 object2 .... }  
list of definition_champ (4.2.2)
```

4.2.2 definition_champ

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

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

Usage:

name **champ_generique**

where

- **name** *str*: The name of the new created field.
- **champ_generique** *champ_generique_base* (8)

4.2.3 sondes

Description: List of probes.

See also: *listobj* (34.3)

Usage:

{ object1 object2 }

list of *sonde* (4.2.4)

4.2.4 sonde

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

See also: *objet_lecture* (35)

Usage:

nom_sonde [**special**] **nom_inco** **mperiode** **prd** **type**

where

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

4.2.5 sonde_base

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

See also: [objet_lecture \(35\)](#) [points \(4.2.6\)](#) [numero_elem_sur_maitre \(4.2.10\)](#) [position_like \(4.2.11\)](#) [segment \(4.2.12\)](#) [plan \(4.2.13\)](#) [volume \(4.2.14\)](#) [circle \(4.2.15\)](#) [circle_3 \(4.2.16\)](#) [segmentfacesx \(4.2.17\)](#) [segmentfacesy \(4.2.18\)](#) [segmentfacesz \(4.2.19\)](#)

Usage:

sonde_base

4.2.6 points

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

See also: [sonde_base \(4.2.5\)](#) [point \(4.2.8\)](#) [segmentpoints \(4.2.9\)](#)

Usage:

points points

where

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

4.2.7 listpoints

Description: Points.

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

Usage:

n object1 object2

list of *un_point* ([3.15.3](#))

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* ([3.15.3](#)): First outer probe segment point.
- **point_fin** *un_point* ([3.15.3](#)): 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* ([3.15.3](#)): First point defining the angle. This angle should be positive.
- **point_fin** *un_point* ([3.15.3](#)): Second point defining the angle. This angle should be positive.
- **point_fin_2** *un_point* ([3.15.3](#)): 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* ([3.15.3](#)): Point of origin.
- **point_fin** *un_point* ([3.15.3](#)): Point defining the first direction (from point of origin).
- **point_fin_2** *un_point* ([3.15.3](#)): Point defining the second direction (from point of origin).
- **point_fin_3** *un_point* ([3.15.3](#)): 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* ([3.15.3](#)): 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* ([3.15.3](#)): Center of the circle.
- **direction** *int into [0, 1, 2]*: Axis normal to the circle plane (0:x axis, 1:y axis, 2:z axis).
- **radius** *float*: Radius of the circle.
- **theta1** *float*: First angle.
- **theta2** *float*: Second angle.

4.2.17 segmentfacesx

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

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

Usage:

segmentfacesx **nbr** **point_deb** **point_fin**

where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* ([3.15.3](#)): First outer probe segment point.
- **point_fin** *un_point* ([3.15.3](#)): Second outer probe segment point.

4.2.18 segmentfacesy

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

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

Usage:

segmentfacesy **nbr** **point_deb** **point_fin**

where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* ([3.15.3](#)): First outer probe segment point.
- **point_fin** *un_point* ([3.15.3](#)): Second outer probe segment point.

4.2.19 segmentfacesz

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

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

Usage:

segmentfacesz **nbr** **point_deb** **point_fin**

where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* ([3.15.3](#)): First outer probe segment point.
- **point_fin** *un_point* ([3.15.3](#)): Second outer probe segment point.

4.2.20 champs_posts

Description: Field's write mode.

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

Usage:

[**format**] **mot** **period** **fields|champs**

where

- **format** *str into ['binaire', 'formatte']*: Type of file.

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

4.2.21 champs_a_post

Description: Fields to be post-processed.

See also: listobj (34.3)

Usage:

{ object1 object2 }

list of *champ_a_post* (4.2.22)

4.2.22 champ_a_post

Description: Field to be post-processed.

See also: objet_lecture (35)

Usage:

champ [**localisation**]

where

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

4.2.23 stats_posts

Description: Field's write mode.

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

dts: frequency value.

t_deb value: Start of integration time

t_fin value: End of integration time

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

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

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

Example:

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

It will write every **dt_post** the mean, standard deviation and correlation value:

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

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

Usage:

mot period fields|champs

where

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

4.2.24 list_stat_post

Description: Post-processing for statistics

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

Usage:

{ object1 object2 }

list of *stat_post_deriv* [\(4.2.25\)](#)

4.2.25 stat_post_deriv

Description: not_set

See also: [objet_lecture \(35\)](#) [t_deb \(4.2.26\)](#) [t_fin \(4.2.27\)](#) [moyenne \(4.2.28\)](#) [ecart_type \(4.2.29\)](#) [correlation \(4.2.30\)](#)

Usage:

stat_post_deriv

4.2.26 t_deb

Description: not_set

See also: [stat_post_deriv \(4.2.25\)](#)

Usage:

t_deb **val**

where

- **val** *float*

4.2.27 **t_fin**

Description: not_set

See also: stat_post_deriv ([4.2.25](#))

Usage:

t_fin **val**

where

- **val** *float*

4.2.28 **moyenne**

Synonymous: **champ_post_statistiques_moyenne**

Description: not_set

See also: stat_post_deriv ([4.2.25](#))

Usage:

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

where

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

4.2.29 **ecart_type**

Synonymous: **champ_post_statistiques_ecart_type**

Description: not_set

See also: stat_post_deriv ([4.2.25](#))

Usage:

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

where

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

4.2.30 correlation

Synonymous: **champ_post_statistiques_correlation**

Description: not_set

See also: stat_post_deriv (4.2.25)

Usage:

correlation first_field second_field [localisation]

where

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

4.2.31 stats_serie_posts

Description: Post-processing for statistics.

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

dt_integr value : Period of integration and write period.

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

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

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

Example:

```

Statistiques_en_serie Dt_integr dtst {
Moyenne Pression
}

```

Will calculate and write every dtst seconds the mean value:

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

See also: objet_lecture (35)

Usage:

mot dt_integr stat

where

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

4.3 post_processings

Synonymous: **postraitements**

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

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

Usage:

{ object1 object2 }

list of *un_postraitement* ([4.3.1](#))

4.3.1 un_postraitement

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

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

Usage:

nom post

where

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

4.4 liste_post_ok

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

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

Usage:

{ object1 object2 }

list of *nom_postraitement* ([4.4.1](#))

4.4.1 nom_postraitement

Description:

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

Usage:

nom post

where

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

4.4.2 postraitement_base

Description: not_set

See also: objet_lecture ([35](#)) post_processing ([4.4.3](#)) postraitement_ft_lata ([4.4.4](#))

Usage:

4.4.3 post_processing

Synonymous: **postraitement**

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

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

Usage:

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

where

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

4.4.4 postraitement_ft_lata

Description: `not_set`

See also: `postraitement_base` (4.4.2)

Usage:

```
postraitement_ft_lata bloc  
where
```

- **bloc** *str*

4.5 liste_post

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

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

Usage:

{ object1 object2 }

list of *un_postraitement_spec* ([4.5.1](#))

4.5.1 un_postraitement_spec

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

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

Usage:

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

where

- **type_un_post** *type_un_post* ([4.5.2](#))
- **type_postraitement_ft_lata** *type_postraitement_ft_lata* ([4.5.3](#))

4.5.2 type_un_post

Description: not_set

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

Usage:

type post

where

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

4.5.3 type_postraitement_ft_lata

Description: not_set

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

Usage:

type nom bloc

where

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

4.6 format_file

Description: File formatted.

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

Usage:

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

where

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

4.7 probleme_couple

Description: This instruction causes a `probleme_couple` type object to be created. This type of object has an associated problem list, that is, the coupling of n problems among them may be processed. Coupling between these problems is carried out explicitly via conditions at particular contact limits. Each problem may be associated either with the `Associate` keyword or with the `Read/groupes` keywords. The difference is that in the first case, the four problems exchange values then calculate their timestep, rather in the second case, the same strategy is used for all the problems listed inside one group, but the second group of problem exchange values with the first group of problems after the first group did its timestep. So, the first case may then also be written like this:

`Probleme_Couple pbc`

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

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

Each problem is resolved in a domain.

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

See also: `pb_gen_base` (4) `pb_couple_rayonnement` (4.41) `pb_couple_rayo_semi_transp` (4.15)

Usage:

probleme_couple obj Lire obj {

 [**groupes** *list_list_nom*]

}

where

- **groupes** *list_list_nom* (4.8): { groupes { { pb1 , pb2 } , { pb3 , pb4 } } }

4.8 list_list_nom

Description: pour les groupes

See also: `listobj` (34.3)

Usage:

{ object1 , object2 }

list of *list_un_pb* (34.1) separated with ,

4.9 modele_rayo_semi_transp

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

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

See also: Pb_base (4.1)

Usage:

```
modele_rayo_semi_transp obj Lire obj {  
    [ eq_rayo_semi_transp eq_rayo_semi_transp ]  
    [ Post_processing|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

- **eq_rayo_semi_transp** *eq_rayo_semi_transp* (4.10): Irradiancy G equation. Radiative flux equals $-\text{grad}(G)/3/\kappa$.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N <> P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.10 eq_rayo_semi_transp

Description: Irradiancy equation.

See also: objet_lecture (35)

Usage:

```
{
```

```

    solveur solveur_sys_base
    [ boundary_conditions|conditions_limite condlims]
}
where

```

- **solveur** *solveur_sys_base* (10.16): Solver of the irradiancy equation.
- **boundary_conditions|conditions_limite condlims** (4.10.1): Boundary conditions.

4.10.1 condlims

Description: Boundary conditions.

See also: listobj (34.3)

Usage:

```

{ object1 object2 .... }
list of condlimlu (4.10.2)

```

4.10.2 condlimlu

Description: Boundary condition specified.

See also: objet_lecture (35)

Usage:

```

bord cl
where

```

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

4.11 pb_avec_passif

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

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

See also: Pb_base (4.1) pb_thermohydraulique_concentration_scalaires_passifs (4.28) pb_thermohydraulique_scalaires_passifs (4.33) pb_hydraulique_concentration_scalaires_passifs (4.19) pb_thermohydraulique_qc_fraction_massique (4.32) pb_thermohydraulique_concentration_turbulent_scalaires_passifs (4.30) pb_thermohydraulique_turbulent_scalaires_passifs (4.37) pb_hydraulique_concentration_turbulent_scalaires_passifs (4.21) pb_thermohydraulique_turbulent_qc_fraction_massique (4.36)

Usage:

```

pb_avec_passif obj Lire obj {
    equations_scalaires_passifs listeqn
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
}

```

```
[ reprise format_file]
[ resume_last_time format_file]

}
```

where

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

4.12 listeqn

Description: List of equations.

See also: listobj (34.3)

Usage:

```
{ object1 object2 .... }
list of eqn_base (5.23)
```

4.13 pb_conduction

Description: Resolution of the heat equation.

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

See also: Pb_base (4.1)

Usage:

```
pb_conduction obj Lire obj {  
    [ conduction conduction ]  
    [ Post_processing|postraitement corps_postraitement ]  
    [ Post_processings|postraitements post_processings ]  
    [ liste_de_postraitements liste_post_ok ]  
    [ liste_postraitements liste_post ]  
    [ sauvegarde format_file ]  
    [ sauvegarde_simple format_file ]  
    [ reprise format_file ]  
    [ resume_last_time format_file ]  
}
```

where

- **conduction** *conduction* (5.8): Heat equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.14 pb_conduction_milieu_variable

Description: Resolution of the heat equation.

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

See also: Pb_base (4.1)

Usage:

```
pb_conduction_milieu_variable obj Lire obj {
```

```

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

```

}

where

- **conduction_milieu_variable** *conduction_milieu_variable* (5.9): Heat equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.15 pb_couple_rayo_semi_transp

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

You have to associate a modele_rayo_semi_transp

You have to add a radiative term source in energy equation

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

See also: probleme_couple (4.7)

Usage:

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

where

- **groupes** *list_list_nom* (4.8) for inheritance: { groupes { { pb1 , pb2 } , { pb3 , pb4 } } }

4.16 pb_hydraulique

Description: Resolution of the Navier-Stokes equations.

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

See also: Pb_base (4.1)

Usage:

```
pb_hydraulique obj Lire obj {  
    navier_stokes_standard navier_stokes_standard  
    [ Post_processing|postraitement corps_postraitement ]  
    [ Post_processings|postraitements post_processings ]  
    [ liste_de_postraitements liste_post_ok ]  
    [ liste_postraitements liste_post ]  
    [ sauvegarde format_file ]  
    [ sauvegarde_simple format_file ]  
    [ reprise format_file ]  
    [ resume_last_time format_file ]  
}
```

where

- **navier_stokes_standard** *navier_stokes_standard* (5.32): Navier-Stokes equations.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the

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

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

4.17 pb_hydraulique_ALE

Description: Resolution of hydraulic problems for ALE

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

See also: Pb_base (4.1)

Usage:

```
pb_hydraulique_ALE obj Lire obj {
    navier_stokes_standard_ALE 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_ALE** *navier_stokes_standard* (5.32): Navier-Stokes equations for ALE problems
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.

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

4.18 pb_hydraulique_concentration

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

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

See also: Pb_base (4.1)

Usage:

```
pb_hydraulique_concentration obj Lire obj {
    [ navier_stokes_standard navier_stokes_standard]
    [ convection_diffusion_concentration convection_diffusion_concentration]
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
    [ reprise format_file]
    [ resume_last_time format_file]
}
```

where

- **navier_stokes_standard** *navier_stokes_standard* (5.32): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.12): Constituent transport vectorial equation (concentration diffusion convection).
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.

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

4.19 pb_hydraulique_concentration_scalaires_passifs

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

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

See also: **pb_avec_passif** (4.11)

Usage:

```
pb_hydraulique_concentration_scalaires_passifs obj Lire obj {
    [ navier_stokes_standard navier_stokes_standard]
    [ convection_diffusion_concentration convection_diffusion_concentration]
    equations_scalaires_passifs listeqn
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
    [ reprise format_file]
    [ resume_last_time format_file]
}
```

where

- **navier_stokes_standard** *navier_stokes_standard* (5.32): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.12): Constituent transport equations (concentration diffusion convection).
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_masseN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.

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

4.20 pb_hydraulique_concentration_turbulent

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

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

See also: *Pb_base* (4.1)

Usage:

```
pb_hydraulique_concentration_turbulent obj Lire obj {
    [ navier_stokes_turbulent navier_stokes_turbulent ]
    [ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_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* (5.33): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (5.14): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.

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

4.21 pb_hydraulique_concentration_turbulent_scalaires_passifs

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

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

See also: pb_avec_passif (4.11)

Usage:

```
pb_hydraulique_concentration_turbulent_scalaires_passifs obj Lire obj {
    [ navier_stokes_turbulent   navier_stokes_turbulent]
    [ convection_diffusion_concentration_turbulent   convection_diffusion_concentration_turbulent]
    equations_scalaires_passifs   listeqn
    [ Post_processing|postraitement   corps_postraitement]
    [ Post_processings|postraitements   post_processings]
    [ liste_de_postraitements   liste_post_ok]
    [ liste_postraitements   liste_post]
    [ sauvegarde   format_file]
    [ sauvegarde_simple   format_file]
    [ reprise   format_file]
    [ resume_last_time   format_file]
```

```
}
```

where

- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.33): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (5.14): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_masseN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This

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

4.22 pb_hydraulique_turbulent

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

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

See also: Pb_base (4.1)

Usage:

```
pb_hydraulique_turbulent obj Lire obj {
    navier_stokes_turbulent navier_stokes_turbulent
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
    [ reprise format_file]
    [ resume_last_time format_file]
}
where
```

- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.33): Navier-Stokes equations as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This

block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.

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

4.23 pb_mg

Description: Multi-grid problem.

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

See also: pb_gen_base (4)

Usage:

pb_mg

4.24 pb_phase_field

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

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

See also: Pb_base (4.1)

Usage:

pb_phase_field obj Lire obj {

```
[ navier_stokes_phase_field  navier_stokes_phase_field]
[ convection_diffusion_phase_field  convection_diffusion_phase_field]
[ Post_processing|postraitements  corps_postraitements]
[ Post_processings|postraitements  post_processings]
[ liste_de_postraitements  liste_post_ok]
[ liste_postraitements  liste_post]
[ sauvegarde  format_file]
[ sauvegarde_simple  format_file]
[ reprise  format_file]
[ resume_last_time  format_file]
```

}
where

- **navier_stokes_phase_field** *navier_stokes_phase_field* (5.30): Navier Stokes equation for the Phase Field problem.
- **convection_diffusion_phase_field** *convection_diffusion_phase_field* (5.17): Cahn-Hilliard equation of the Phase Field problem. The unknown of this equation is the concentration C.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.25 pb_post

Description: not_set

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

See also: Pb_base (4.1)

Usage:

```
pb_post obj Lire obj {
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
    [ reprise format_file]
    [ resume_last_time format_file]
}
```

}
where

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

4.26 pb_thermohydraulique

Description: Resolution of thermohydraulic problem.

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

See also: Pb_base (4.1)

Usage:

```
pb_thermohydraulique obj Lire obj {
    [ navier_stokes_standard navier_stokes_standard]
    [ convection_diffusion_temperature convection_diffusion_temperature]
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
    [ reprise format_file]
    [ resume_last_time format_file]
}
```

where

- **navier_stokes_standard** *navier_stokes_standard* (5.32): Navier-Stokes equations.
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.18): Energy equation (temperature diffusion convection).

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

4.27 pb_thermohydraulique_concentration

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

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

See also: Pb_base (4.1)

Usage:

```
pb_thermohydraulique_concentration obj Lire obj {
    [ navier_stokes_standard navier_stokes_standard]
    [ convection_diffusion_concentration convection_diffusion_concentration]
    [ convection_diffusion_temperature convection_diffusion_temperature]
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
    [ reprise format_file]
    [ resume_last_time format_file]
}
```

where

- **navier_stokes_standard** *navier_stokes_standard* (5.32): Navier-Stokes equations.

- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.12): Constituent transport equations (concentration diffusion convection).
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.18): Energy equation (temperature diffusion convection).
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.28 pb_thermohydraulique_concentration_scalaires_passifs

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

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

See also: pb_avec_passif (4.11)

Usage:

```
pb_thermohydraulique_concentration_scalaires_passifs obj Lire obj {
    [ navier_stokes_standard navier_stokes_standard]
    [ convection_diffusion_concentration convection_diffusion_concentration]
    [ convection_diffusion_temperature convection_diffusion_temperature]
    equations_scalaires_passifs listeqn
    [ Post_processing|postraitement corps_postraitement]
    [ Post_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_standard** *navier_stokes_standard* (5.32): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.12): Constituent transport equations (concentration diffusion convection).
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.18): Energy equations (temperature diffusion convection).
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.29 pb_thermohydraulique_concentration_turbulent

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

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

See also: Pb_base (4.1)

Usage:

```
pb_thermohydraulique_concentration_turbulent obj Lire obj {
```

```
[ navier_stokes_turbulent navier_stokes_turbulent]
```

```

[ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent]
[ convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent]
[ Post_processing|postraitement corps_postraitement]
[ Post_processings|postraitements post_processings]
[ liste_de_postraitements liste_post_ok]
[ liste_postraitements liste_post]
[ sauvegarde format_file]
[ sauvegarde_simple format_file]
[ reprise format_file]
[ resume_last_time format_file]
}
where

```

- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.33): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (5.14): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (5.22): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.30 pb_thermohydraulique_concentration_turbulent_scalaires_passifs

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

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

See also: `pb_avec_passif` (4.11)

Usage:

```
pb_thermohydraulique_concentration_turbulent_scalaires_passifs obj Lire obj {
    [ navier_stokes_turbulent navier_stokes_turbulent ]
    [ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent ]
    [ convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent ]
    equations_scalaires_passifs listeqn
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file ]
    [ sauvegarde_simple format_file ]
    [ reprise format_file ]
    [ resume_last_time format_file ]
}
```

where

- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.33): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (5.14): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (5.22): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_masseN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on

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

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

4.31 pb_thermohydraulique_qc

Description: Resolution of thermohydraulic problem under low Mach number.

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

masse_volumique : density

enthalpie : enthalpy

pression : reduced pressure

pression_tot : total pressure.

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

See also: Pb_base (4.1)

Usage:

```
pb_thermohydraulique_qc obj Lire obj {
    navier_stokes_qc navier_stokes_qc
    convection_diffusion_chaleur_qc convection_diffusion_chaleur_qc
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_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* (5.31): Navier-Stokes equations under low Mach number.
- **convection_diffusion_chaleur_qc** *convection_diffusion_chaleur_qc* (5.10): Energy equation under low Mach number.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.

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

4.32 pb_thermohydraulique_qc_fraction_massique

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

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

See also: pb_avec_passif (4.11)

Usage:

```
pb_thermohydraulique_qc_fraction_massique obj Lire obj {
    navier_stokes_qc navier_stokes_qc
    convection_diffusion_chaleur_qc convection_diffusion_chaleur_qc
    equations_scalaires_passifs listeqn
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file ]
    [ sauvegarde_simple format_file ]
    [ reprise format_file ]
    [ resume_last_time format_file ]
}
```

where

- **navier_stokes_qc** *navier_stokes_qc* (5.31): Navier-Stokes equations under low Mach number.
- **convection_diffusion_chaleur_qc** *convection_diffusion_chaleur_qc* (5.10): Energy equation under low Mach number.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This

block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory `lata` used in this example should be created before running the computation or the `lata` files will be lost.

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

4.33 pb_thermohydraulique_scalaires_passifs

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

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

See also: `pb_avec_passif` (4.11)

Usage:

```
pb_thermohydraulique_scalaires_passifs obj Lire obj {
    [ navier_stokes_standard navier_stokes_standard]
    [ convection_diffusion_temperature convection_diffusion_temperature]
    equations_scalaires_passifs listeqn
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
    [ reprise format_file]
    [ resume_last_time format_file]
}
```

where

- **navier_stokes_standard** *navier_stokes_standard* (5.32): Navier-Stokes equations.
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.18): Energy equations (temperature diffusion convection).
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number `N` are named `temperatureN` or `concentrationN` or `fraction_masseN`. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.

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

4.34 pb_thermohydraulique_turbulent

Description: Resolution of thermohydraulic problem, with turbulence modelling.

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

See also: Pb_base (4.1)

Usage:

```
pb_thermohydraulique_turbulent obj Lire obj {
    navier_stokes_turbulent navier_stokes_turbulent
    convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
    [ reprise format_file]
    [ resume_last_time format_file]
}
```

where

- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.33): Navier-Stokes equations as well as the associated turbulence model equations.

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

4.35 pb_thermohydraulique_turbulent_qc

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

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

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

See also: Pb_base (4.1)

Usage:

```
pb_thermohydraulique_turbulent_qc obj Lire obj {
    navier_stokes_turbulent_qc navier_stokes_turbulent_qc
    convection_diffusion_chaleur_turbulent_qc convection_diffusion_chaleur_turbulent_qc
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
    [ reprise format_file]
    [ resume_last_time format_file]
```

```
}
```

where

- **navier_stokes_turbulent_qc** *navier_stokes_turbulent_qc* (5.34): Navier-Stokes equations under low Mach number as well as the associated turbulence model equations.
- **convection_diffusion_chaleur_turbulent_qc** *convection_diffusion_chaleur_turbulent_qc* (5.11): Energy equation under low Mach number as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.36 pb_thermohydraulique_turbulent_qc_fraction_massique

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

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

See also: pb_avec_passif (4.11)

Usage:

pb_thermohydraulique_turbulent_qc_fraction_massique obj Lire obj {

```

    navier_stokes_turbulent_qc navier_stokes_turbulent_qc
    convection_diffusion_chaleur_turbulent_qc convection_diffusion_chaleur_turbulent_qc
    equations_scalaires_passifs listeqn
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processing|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
    [ reprise format_file]

```

```
[ resume_last_time format_file]
}
```

where

- **navier_stokes_turbulent_qc** *navier_stokes_turbulent_qc* (5.34): Navier-Stokes equations under low Mach number as well as the associated turbulence model equations.
- **convection_diffusion_chaleur_turbulent_qc** *convection_diffusion_chaleur_turbulent_qc* (5.11): Energy equation under low Mach number as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.37 pb_thermohydraulique_turbulent_scalaires_passifs

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

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

See also: pb_avec_passif (4.11)

Usage:

```
pb_thermohydraulique_turbulent_scalaires_passifs obj Lire obj {
```

```
[ navier_stokes_turbulent navier_stokes_turbulent]
```

```

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

```

- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.33): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (5.22): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.38 pbc_med

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

See also: pb_gen_base (4)

Usage:

pbc_med list_info_med

where

- **list_info_med** *list_info_med* (4.39)

4.39 list_info_med

Description: not_set

See also: listobj (34.3)

Usage:

{ object1 , object2 }

list of *info_med* (4.39.1) separated with ,

4.39.1 info_med

Description: not_set

See also: objet_lecture (35)

Usage:

file_med domaine pb_post

where

- **file_med** *str*: Name of the MED file.
- **domaine** *str*: Name of domain.
- **pb_post** *pb_post* (4.25)

4.40 problem_read_generic

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

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

See also: Pb_base (4.1) `probleme_ft_disc_gen` (4.42)

Usage:

problem_read_generic obj Lire obj {

[**Post_processing|postraitement** *corps_postraitement*]

[**Post_processings|postraitements** *post_processings*]

[**liste_de_postraitements** *liste_post_ok*]

[**liste_postraitements** *liste_post*]

[**sauvegarde** *format_file*]

```

[ sauvegarde_simple format_file]
[ reprise format_file]
[ resume_last_time format_file]
}
where

```

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

4.41 pb_couple_rayonnement

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

See also: probleme_couple (4.7)

Usage:

```

pb_couple_rayonnement obj Lire obj {
    [ groupes list_list_nom]
}
where

```

- **groupes** *list_list_nom* (4.8) for inheritance: { groupes { { pb1 , pb2 } , { pb3 , pb4 } } }

4.42 probleme_ft_disc_gen

Description: The generic Front-Tracking problem in the discontinuous version. It differs from the rest of the TRUST code : The problem does not state the number of equations that are enclosed in the problem. Two equations are compulsory : a momentum balance equation (alias Navier-Stokes equation) and an interface tracking equation. The list of equations to be solved is declared in the beginning of the data file. Another difference with more classical TRUST data file, lies in the fluids definition. The two-phase fluid (Fluide_Diphasique) is made with two usual single-phase fluids (Fluide_Incompressible). As the list of equations to be solved in the generic Front-Tracking problem is declared in the data file and not pre-defined in the structure of the problem, each equation has to be distinctively associated with the problem with the Associer keyword.

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

See also: `problem_read_generic` (4.40)

Usage:

```
probleme_ft_disc_gen obj Lire obj {  
    [ Post_processing|postraitement corps_postraitement]  
    [ Post_processings|postraitements post_processings]  
    [ liste_de_postraitements liste_post_ok]  
    [ liste_postraitements liste_post]  
    [ sauvegarde format_file]  
    [ sauvegarde_simple format_file]  
    [ reprise format_file]  
    [ resume_last_time format_file]  
}
```

where

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

5 mor_eqn

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

See also: `objet_u` (36) `eqn_base` (5.23)

Usage:

5.1 Transport_K_Eps_Realisable

Description: Realizable K-Epsilon Turbulence Model Transport Equations for K and Epsilon.

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

See also: `eqn_base` (5.23)

Usage:

```
Transport_K_Eps_Realisable 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* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.2 bloc_convection

Description: not_set

See also: objet_lecture (35)

Usage:

aco operateur acof

where

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

5.2.1 convection_deriv

Description: not_set

See also: objet_lecture (35) [amount](#) (5.2.2) [amount_old](#) (5.2.3) [centre](#) (5.2.4) [centre4](#) (5.2.5) [centre_old](#) (5.2.6) [di_l2](#) (5.2.7) [ef](#) (5.2.8) [muscl3](#) (5.2.10) [ef_stab](#) (5.2.11) [generic](#) (5.2.14) [kquick](#) (5.2.15) [muscl](#) (5.2.16) [muscl_old](#) (5.2.17) [muscl_new](#) (5.2.18) [negligeable](#) (5.2.19) [quick](#) (5.2.20) [ale](#) (5.2.21) [btd](#) (5.2.22) [supg](#) (5.2.23) [RT](#) (5.2.24)

Usage:

convection_deriv

5.2.2 amount

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

See also: convection_deriv (5.2.1)

Usage:

amount

5.2.3 amount_old

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

See also: convection_deriv (5.2.1)

Usage:

amount_old

5.2.4 centre

Description: For VDF and VEF discretizations.

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

Usage:

centre

5.2.5 centre4

Description: For VDF and VEF discretizations.

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

Usage:

centre4

5.2.6 centre_old

Description: Only for VEF discretization.

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

Usage:

centre_old

5.2.7 di_l2

Description: Only for VEF discretization.

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

Usage:

di_l2

5.2.8 ef

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

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

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

Remark:

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

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

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* (5.2.9)

5.2.9 bloc_ef

Description: not_set

See also: objet_lecture (35)

Usage:

mot1 val1 mot2 val2 mot3 val3 mot4 val4

where

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

5.2.10 muscl3

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

See also: convection_deriv (5.2.1)

Usage:

muscl3 {

 [**alpha** *float*]

}

where

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

5.2.11 ef_stab

Description: Keyword for a VEF convective scheme.

See also: convection_deriv (5.2.1)

Usage:

ef_stab {

 [**alpha** *float*]

 [**test** *int*]

 [**tdivu**]

```

[ old ]
[ volumes_etendus ]
[ volumes_non_etendus ]
[ amont_sous_zone str]
[ alpha_sous_zone listsous_zone_valeur]
}
where

```

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

5.2.12 listsous_zone_valeur

Description: List of groups of two words.

See also: listobj (34.3)

Usage:

n object1 object2

list of *sous_zone_valeur* (5.2.13)

5.2.13 sous_zone_valeur

Description: Two words.

See also: objet_lecture (35)

Usage:

sous_zone valeur

where

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

5.2.14 generic

Description: Keyword for generic calling of upwind and muscl convective scheme in VEF discretization. For muscl scheme, limiters and order for fluxes calculations have to be specified. The available limiters are : minmod - vanleer - vanalbada - chakravarthy - superbee, and the order of accuracy is 1 or 2. Note that chakravarthy is a non-symmetric limiter and superbee may engender results out of physical limits. By

consequence, these two limiters are not recommended.

Examples:

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

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

See also: convection_deriv ([5.2.1](#))

Usage:

generic type [limiteur] [ordre] [alpha]

where

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

5.2.15 kquick

Description: Only for VEF discretization.

See also: convection_deriv ([5.2.1](#))

Usage:

kquick

5.2.16 muscl

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

See also: convection_deriv ([5.2.1](#))

Usage:

muscl

5.2.17 muscl_old

Description: Only for VEF discretization.

See also: convection_deriv ([5.2.1](#))

Usage:

muscl_old

5.2.18 muscl_new

Description: Only for VEF discretization.

See also: convection_deriv ([5.2.1](#))

Usage:

muscl_new

5.2.19 **negligeable**

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

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

Usage:

negligeable

5.2.20 **quick**

Description: Only for VDF discretization.

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

Usage:

quick

5.2.21 **ale**

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

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

Usage:

ale opconv

where

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

5.2.22 **btd**

Description: Only for EF discretization.

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

Usage:

btd {

btd *float*

facteur *float*

}

where

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

5.2.23 supg

Description: Only for EF discretization.

See also: convection_deriv (5.2.1)

Usage:

```
supg {  
    facteur float  
}
```

where

- **facteur** *float*

5.2.24 RT

Description: Keyword to use RT projection for P1NCP0RT discretization

See also: convection_deriv (5.2.1)

Usage:

RT

5.3 bloc_diffusion

Description: not_set

See also: objet_lecture (35)

Usage:

```
aco [ opérateur ] [ op_implicite ] acof  
where
```

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

5.3.1 diffusion_deriv

Description: not_set

See also: objet_lecture (35) negligeable (5.3.2) p1b (5.3.3) p1ncp1b (5.3.4) stab (5.3.5) standard (5.3.6) option (5.3.8)

Usage:

diffusion_deriv

5.3.2 negligible

Description: the diffusivity will not taken in count

See also: `diffusion_deriv` ([5.3.1](#))

Usage:

negligeable

5.3.3 p1b

Description: `not_set`

See also: `diffusion_deriv` ([5.3.1](#))

Usage:

p1b

5.3.4 p1ncp1b

Description: `not_set`

See also: `diffusion_deriv` ([5.3.1](#))

Usage:

5.3.5 stab

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

See also: `diffusion_deriv` ([5.3.1](#))

Usage:

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

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

- **nu_transp** *int*: (respectively **nut_transp** 1) takes the molecular viscosity (resp. eddy viscosity) into account in the transposed velocity gradient part of the diffusion expression (by default **nu_transp**=0 and **nut_transp**=1)
- **nut_transp** *int*

5.3.6 standard

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

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

See also: `diffusion_deriv` (5.3.1)

Usage:

standard [**mot1**] [**bloc_diffusion_standard**]

where

- **mot1** *str* into [*'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* (5.3.7)

5.3.7 bloc_diffusion_standard

Description: `grad_Ubar 1` makes the gradient calculated through the filtered values of velocity (P1-conform). `nu 1` (respectively `nut 1`) takes the molecular viscosity (eddy viscosity) into account in the velocity gradient part of the diffusion expression.

`nu_transp 1` (respectively `nut_transp 1`) takes the molecular viscosity (eddy viscosity) into account according in the TRANPOSED velocity gradient part of the diffusion expression.

`filtrer_resu 1` allows to filter the resulting diffusive fluxes contribution.

See also: `objet_lecture` (35)

Usage:

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

where

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

5.3.8 option

Description: not_set

See also: diffusion_deriv (5.3.1)

Usage:

option bloc_lecture

where

- **bloc_lecture** *bloc_lecture* (3.6)

5.3.9 op_implicite

Description: not_set

See also: objet_lecture (35)

Usage:

implicite mot solveur

where

- **implicite** *str* into ['implicite']
- **mot** *str* into ['solveur']
- **solveur** *solveur_sys_base* (10.16)

5.4 condinits

Description: Initial conditions.

See also: objet_lecture (35)

Usage:

aco condinit acof

where

- **aco** *str* into ['{']: Opening curly bracket.
- **condinit** *condinit* (5.4.1): CI
- **acof** *str* into ['}']: Closing curly bracket.

5.4.1 condinit

Description: Initial condition.

See also: objet_lecture (35)

Usage:

nom ch

where

- **nom** *str*: Name of initial condition field.
- **ch** *champ_base* (16.1): Type field and the initial values.

5.5 sources

Description: The sources.

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

Usage:

{ object1 , object2 }

list of *source_base* ([30](#)) separated with ,

5.6 ecrire_fichier_xyz_valeur_param

Description: not_set

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

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

Usage:

n object1 , object2

list of *ecrire_fichier_xyz_valeur_item* ([5.6.1](#)) 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 \(35\)](#)

Usage:

name dt_ecrire_fic [bords]

where

- **name** *str*: Name of the field to write (*Champ_Inc*, *Champ_Fonc* or a post_processed field).
- **dt_ecrire_fic** *float*: Time period for printing in the file.
- **bords** *bords_ecrire* ([5.6.2](#)): to post-process only on some boundaries

5.6.2 bords_ecrire

Description: not_set

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

Usage:

chaîne bords

where

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

5.7 parametre_equation_base

Description: Basic class for parametre_equation

See also: objet_lecture (35) 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* (10.16): 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 conduction

Description: Heat equation.

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

See also: `eqn_base` (5.23)

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* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.9 conduction_milieu_variable

Description: Heat equation.

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

See also: eqn_base (5.23)

Usage:

```
conduction_milieu_variable obj Lire obj {
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ initial_conditions|conditions_initiales condinits]
    [ boundary_conditions|conditions_limites condlims]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
    [ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
}
```

where

- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.10 convection_diffusion_chaleur_qc

Description: Energy equation under low Mach number.

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

See also: eqn_base (5.23) convection_diffusion_chaleur_turbulent_qc (5.11)

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* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 only for some boundaries in a text file with the following format:
n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbnome_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (5.6) for inheritance: This keyword is used to write the values of a field only for some boundaries in a binary file with the following


```

format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat

```

- **parametre_equation** *parametre_equation_base* (5.7) 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.11 convection_diffusion_chaleur_turbulent_qc

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

Keyword Discretize should have already been used to read the object.
See also: convection_diffusion_chaleur_qc (5.10)

Usage:

```

convection_diffusion_chaleur_turbulent_qc obj Lire obj {
    [ modele_turbulence modele_turbulence_scal_base]
    [ mode_calcul_convection str into ['ancien', 'divuT_moins_Tdivu', 'divrhout_moins_Tdivrhout']]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ initial_conditions|conditions_initiales condinits]
    [ boundary_conditions|conditions_limites condlims]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
    [ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
}

```

where

- **modele_turbulence** *modele_turbulence_scal_base* (24): Turbulence model for the energy equation.
- **mode_calcul_convection** *str into ['ancien', 'divuT_moins_Tdivu', 'divrhout_moins_Tdivrhout']* for inheritance: Option to set the form of the convective operator
divrhout_moins_Tdivrhout (the default since 1.6.8): $\rho \cdot u \cdot \text{grad}T = \text{div}(\rho \cdot u \cdot T) - T \cdot \text{div}(\rho \cdot u)$
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* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.12 convection_diffusion_concentration

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

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

See also: eqn_base (5.23) convection_diffusion_concentration_ft_disc (5.13) convection_diffusion_concentration_turbulent (5.14) convection_diffusion_phase_field (5.17)

Usage:

```
convection_diffusion_concentration obj Lire obj {
    [ nom_inconnue str]
    [ masse_molaire float]
    [ alias str]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ initial_conditions|conditions_initiales condinits]
    [ boundary_conditions|conditions_limites condlims]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
    [ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
}
```

where

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

- **alias** *str*
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.13 convection_diffusion_concentration_ft_disc

Description: not_set

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

See also: convection_diffusion_concentration (5.12)

Usage:

```
convection_diffusion_concentration_ft_disc obj Lire obj {
    [ equation_interface str]
    phase int into [0, 1]
    [ option str]
    [ nom_inconnue str]
    [ masse_molaire float]
    [ alias str]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ initial_conditions|conditions_initiales condinits]
    [ boundary_conditions|conditions_limites condlims]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
```

```

[ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
}
where

```

- **equation_interface** *str*: this is the name of the interface tracking equation to watch. The scalar will not diffuse through the interface of this equation.
- **phase** *int into [0, 1]*: tells whether the scalar must be confined in phase 0 or in phase 1
- **option** *str*: Experimental features used to prevent the concentration to leak through the interface between phases due to numerical diffusion.
RIEN: do nothing
RAMASSE_MIETTES_SIMPLE: at each timestep, this algorithm takes all the mass located in the opposite phase and spreads it uniformly in the given phase.
- **nom_inconnue** *str* for inheritance: Keyword *Nom_inconnue* will rename the unknown of this equation with the given name. In the postprocessing part, the concentration field will be accessible with this name. This is useful if you want to track more than one concentration (otherwise, only the concentration field in the first concentration equation can be accessed).
- **masse_molaire** *float* for inheritance
- **alias** *str* for inheritance
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limite** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.14 convection_diffusion_concentration_turbulent

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

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

See also: `convection_diffusion_concentration` (5.12)

Usage:

```
convection_diffusion_concentration_turbulent obj Lire obj {
    [ modele_turbulence modele_turbulence_scal_base]
    [ nom_inconnue str]
    [ masse_molaire float]
    [ alias str]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ initial_conditions|conditions_initiales condinits]
    [ boundary_conditions|conditions_limites condlims]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
    [ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
}
```

where

- **modele_turbulence** *modele_turbulence_scal_base* (24): Turbulence model to be used in the constituent transport equations. The only model currently available is Schmidt.
- **nom_inconnue** *str* for inheritance: Keyword `Nom_inconnue` will rename the unknown of this equation with the given name. In the postprocessing part, the concentration field will be accessible with this name. This is 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* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.15 convection_diffusion_fraction_massique_qc

Description: not_set

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

See also: eqn_base (5.23)

Usage:

convection_diffusion_fraction_massique_qc obj Lire obj {

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

}

where

- **espece** *espece* (15)
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.

Navier_Sokes_Standard

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

5.16 convection_diffusion_fraction_massique_turbulent_qc

Description: not_set

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

See also: eqn_base (5.23)

Usage:

convection_diffusion_fraction_massique_turbulent_qc obj Lire obj {

[**modele_turbulence** *modele_turbulence_scal_base*]

espece *espece*

[**convection** *bloc_convection*]

[**diffusion** *bloc_diffusion*]

[**initial_conditions|conditions_initiales** *condinits*]

[**boundary_conditions|conditions_limites** *condlims*]

[**sources** *sources*]

[**ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param*]

[**ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param*]

[**parametre_equation** *parametre_equation_base*]

[**equation_non_resolue** *str*]

}

where

- **modele_turbulence** *modele_turbulence_scal_base* (24): Turbulence model to be used.
- **espece** *espece* (15)
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time `t0` and `t1`.
`Navier_Sokes_Standard`
`{ equation_non_resolue (t>t0)*(t<t1) }`

5.17 convection_diffusion_phase_field

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

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

See also: `convection_diffusion_concentration` (5.12)

Usage:

convection_diffusion_phase_field `obj Lire obj {`

```

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

- **mu_1** *float*: Dynamic viscosity of the first phase.
- **mu_2** *float*: Dynamic viscosity of the second phase.
- **rho_1** *float*: Density of the first phase.
- **rho_2** *float*: Density of the second phase.
- **potentiel_chimique_generalise** *str into ['avec_energie_cinetique', 'sans_energie_cinetique']*: To define (chaîne set to `avec_energie_cinetique`) or not (chaîne set to `sans_energie_cinetique`) if the Cahn-Hilliard equation contains the cinetic energy term.

- **nom_inconnue** *str* for inheritance: Keyword `Nom_inconnue` will rename the unknown of this equation with the given name. In the postprocessing part, the concentration field will be accessible with this name. This is useful if you want to track more than one concentration (otherwise, only the concentration field in the first concentration equation can be accessed).
- **masse_molaire** *float* for inheritance
- **alias** *str* for inheritance
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time `t0` and `t1`.
`Navier_Sokes_Standard`
`{ equation_non_resolue (t>t0)*(t<t1) }`

5.18 convection_diffusion_temperature

Description: Energy equation (temperature diffusion convection).

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

See also: `eqn_base` (5.23) `convection_diffusion_temperature_ft_disc` (5.20)

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* (5.19): to activate or not (the default is Direct Forcing method) the Penalized Direct Forcing method to impose the specified temperature on the solid-fluid interface.
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limite** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.19 pp

Description: not_set

See also: listobj (34.3)

Usage:

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

list of *penalisation_l2_ftd_lec* (5.19.1)

5.19.1 penalisation_l2_ftd_lec

Description: not_set

See also: objet_lecture (35)

Usage:

bord *val*

where

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

5.20 convection_diffusion_temperature_ft_disc

Description: not_set

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

See also: convection_diffusion_temperature (5.18)

Usage:

convection_diffusion_temperature_ft_disc obj Lire obj {

```
[ equation_interface str]  
phase int into [0, 1]  
[ equation_navier_stokes str]  
[ stencil_width int]  
[ maintien_temperature objet_lecture_maintien_temperature]  
[ penalisation_l2_ftd pp]  
[ convection bloc_convection]  
[ diffusion bloc_diffusion]  
[ initial_conditions|conditions_initiales condinits]  
[ boundary_conditions|conditions_limites condlims]  
[ sources sources]  
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]  
[ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]  
[ parametre_equation parametre_equation_base]  
[ equation_non_resolue str]
```

}

where

- **equation_interface** *str*: The name of the interface equation should be given.
- **phase** *int into [0, 1]*: Phase in which the temperature equation will be solved. The temperature, which may be postprocessed with the keyword temperature_EquationName, in the other phase may be negative: the code only computes the temperature field in the specified phase. The other phase is supposed to physically stay at saturation temperature. The code uses a ghost fluid numerical method to work on a smooth temperature field at the interface. In the opposite phase (1-X) the temperature will therefore be extrapolated in the vicinity of the interface and have the opposite sign, saturation temperature is zero by convention).
- **equation_navier_stokes** *str*: The name of the Navier Stokes equation of the problem should be given.
- **stencil_width** *int*: distance in mesh elements over which the temperature field should be extrapolated in the opposite phase.
- **maintien_temperature** *objet_lecture_maintien_temperature* (5.21): **maintien_temperature SOUS_ZONE_NAME VALUE** : experimental, this acts as a dynamic source term that heats or cools the fluid to maintain the average temperature to VALUE within the specified region. At this time, this is done by multiplying the temperature within the SOUS_ZONE by an appropriate uniform value at each timestep. This feature might be implemented in a separate source term in the future.

- **penalisation_l2_ftd** *pp* (5.19) for inheritance: to activate or not (the default is Direct Forcing method) the Penalized Direct Forcing method to impose the specified temperature on the solid-fluid interface.
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.21 objet_lecture_maintien_temperature

Description: not_set

See also: objet_lecture (35)

Usage:

sous_zone **temperature_moyenne**

where

- **sous_zone** *str*
- **temperature_moyenne** *float*

5.22 convection_diffusion_temperature_turbulent

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

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

See also: `eqn_base` (5.23)

Usage:

```
convection_diffusion_temperature_turbulent obj Lire obj {
    [ modele_turbulence modele_turbulence_scal_base]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ initial_conditions|conditions_initiales condinits]
    [ boundary_conditions|conditions_limites condlims]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
    [ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
}
```

where

- **modele_turbulence** *modele_turbulence_scal_base* (24): Turbulence model for the energy equation.
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.23 eqn_base

Description: Basic class for equations.

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

See also: `mor_eqn` (5) `navier_stokes_standard` (5.32) `convection_diffusion_temperature` (5.18) `conduction` (5.8) `convection_diffusion_chaleur_qc` (5.10) `convection_diffusion_concentration` (5.12) `convection_diffusion_fraction_massique_qc` (5.15) `conduction_milieu_variable` (5.9) `transport_interfaces_ft_disc` (5.35) `transport_marqueur_ft` (5.41) `convection_diffusion_temperature_turbulent` (5.22) `convection_diffusion_fraction_massique_turbulent_qc` (5.16) `transport_k_epsilon` (5.40) `Transport_K_Eps_Realisable` (5.1)

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* (5.2): Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3): Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4): Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1): 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 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 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 equations are not solved between time t_0 and t_1 .
`Navier_Sokes_Standard`
`{ equation_non_resolue (t>t0)*(t<t1) }`

5.24 navier_stokes_ft_disc

Description: Two-phase momentum balance equation.

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

See also: `navier_stokes_turbulent` (5.33)

Usage:

```
navier_stokes_ft_disc obj Lire obj {
    [ equation_interfaces_proprietes_fluide str]
    [ equation_interfaces_vitesse_imposee str]
    [ equations_interfaces_vitesse_imposee n word1 word2 ... wordn]
    [ clipping_courbure_interface int]
    [ terme_gravite str into ['rho_g', 'grad_i']]
    [ equation_temperature_mpoint str]
    [ matrice_pression_invariante ]
    [ penalisation_forcage penalisation_forcage]
    [ equation_temperature_mpoint_vapeur str]
    [ mpoint_inactif_sur_qdm ]
    [ mpoint_vapeur_inactif_sur_qdm ]
    [ modele_turbulence modele_turbulence_hyd_deriv]
    [ methode_calcul_pression_initiale str into ['avec_les_cl', 'avec_sources', 'avec_sources_et-
_operateurs', 'sans_rien']]
    [ projection_initiale int]
    [ solveur_pression solveur_sys_base]
    [ solveur_bar solveur_sys_base]
    [ dt_projection deuxmots]
    [ seuil_divU floatfloat]
    [ traitement_particulier traitement_particulier]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ initial_conditions|conditions_initiales condinits]
    [ boundary_conditions|conditions_limites condlims]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
    [ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
}
```

where

- **equation_interfaces_proprietes_fluide** *str*: This keyword is used for liquid-gas, liquid-vapor and fluid-fluid deformable interface, which transported at the Eulerian velocity. When this case is selected, the keyword sequence `Methode_transport_vitesse_interpolee` is used in the block `Transport_Interfaces_FT_Disc` to define the velocity field for the displacement of the interface.
- **equation_interfaces_vitesse_imposee** *str*: This keyword is used to specify the velocity field to be used when using an interface that mimics a solid interface moving with a given solid speed of displacement. When this case is selected, the keyword sequence `Methode_transport_vitesse_imposee` in the `Transport_Interfaces_FT_Disc` block will define the velocity field for the displacement of the interface.
- **equations_interfaces_vitesse_imposee** *n word1 word2 ... wordn*: This keyword is used to specify the velocity field to be used when using an interface that mimics a solid interface moving with a given solid speed of displacement. When this case is selected, the keyword sequence `Methode_transport_vitesse_imposee` in the `Transport_Interfaces_FT_Disc` block will define the velocity field for the displacement of the interface. If two or more solid interfaces are defined, then the keyword `equations_interfaces_vitesse_imposee` should be used.

- **clipping_courbure_interface** *int*: This keyword is used to numerically limit the values of curvature used in the momentum balance equation. Curvature is computed as usual, but values exceeding the clipping value are replaced by this threshold, before using the clipped curvature in the momentum balance. Each time a curvature value is clipped, a counter is increased by one unity and the value of the counter is written in the .err file at the end of the time step. This clipping allows not reducing drastically the time stepping when a geometrical singularity occurs in the interface mesh. However, physical phenomena may be concealed with the use of such a clipping.
- **terme_gravite** *str into ['rho_g', 'grad_i']*: The Terme_gravite keyword changes the numerical scheme used for the gravity source term. The default is grad_i, which is designed to remove spurious currents around the interface. In this case, the pressure field does not contain the hydrostatic part but only a jump across the interface. This scheme seems not to work very well in vef. The rho_g option uses the more traditional source term, equal to rho*g in the volume. In this case, the hydrostatic pressure is visible in the pressure field and the boundary conditions in pressure must be set accordingly. This model produces spurious currents in the vicinity of the fluid-fluid interfaces and with the immersed boundary conditions.
- **equation_temperature_mpoint** *str*: The equation_temperature_mpoint should be used in the case of liquid-vapor flow with phase-change (see the TRUST_ROOT/doc/TRUST/ft_chgt_phase.pdf written in French for more information about the model). The name of the temperature equation, defined with the convection_diffusion_temperature_ft_disc keyword, should be given.
- **matrice_pression_invariante**: This keyword is a shortcut to be used only when the flow is a single-phase one, with interface tracking only used for solid-fluid interfaces. In this peculiar case, the density of the fluid does not evolve during the computation and the pressure matrix does not need to be actuated at each time step.
- **penalisation_forage** *penalisation_forage* (5.25): This keyword is used to specify a strong formulation (value set to 0) or a weak formulation (value set to 1) for an imposed pressure boundary condition. The first formulation converges quicker and is stable in general cases except some rare cases (see Ecoulement_Neumann test case for example) where the second one should be used despite of its slow convergence.
- **equation_temperature_mpoint_vapeur** *str*
- **mpoint_inactif_sur_qdm**
- **mpoint_vapeur_inactif_sur_qdm**
- **modele_turbulence** *modele_turbulence_hyd_deriv* (5.26) for inheritance: Turbulence model for Navier-Stokes equations.
- **methode_calcul_pression_initiale** *str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']* for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : avec_les_cl (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), avec_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicit when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (10.16) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (10.16) 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* (5.27) 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* (5.28) for inheritance: value factor : this keyword is intended to minimise the number of iterations during the pressure system resolution. The convergence criteria during this step

('seuil' in solveur_pression) is dynamically adapted according to the mass conservation. At t_n , the linear system $Ax=B$ is considered as solved if the residual $\|Ax-B\| < \text{seuil}(t_n)$. For t_{n+1} , the threshold value $\text{seuil}(t_{n+1})$ will be evaluated as:

```
If ( lmax(DivU)*dtl<value )
Seuil(tn+1)= Seuil(tn)*factor
Else
Seuil(tn+1)= Seuil(tn)*factor
Endif
```

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

- **traitement_particulier** *traitement_particulier* (5.29) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.25 penalisation_forage

Description: penalisation_forage

See also: objet_lecture (35)

Usage:

```
{
    [ pression_reference float]
    [ domaine_flottant_fluide x1 x2 (x3)]
}
```

where

- **pression_reference** *float*
- **domaine_flottant_fluide** *x1 x2 (x3)*

5.26 modele_turbulence_hyd_deriv

Description: Basic class for turbulence model for Navier-Stokes equations.

See also: objet_lecture (35) NUL (5.26.2) mod_turb_hyd_ss_maille (5.26.3) mod_turb_hyd_rans (5.26.19)

Usage:

```
modele_turbulence_hyd_deriv {  
    [ correction_visco_turb_pour_controle_pas_de_temps ]  
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]  
    [ turbulence_parois turbulence_parois_base]  
    [ dt_impr_ustar float]  
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]  
    [ nut_max float]  
}
```

where

- **correction_visco_turb_pour_controle_pas_de_temps** : Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float*: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_parois** *turbulence_parois_base* (32): Keyword to set the wall law.
- **dt_impr_ustar** *float*: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1): This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float*: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.1 dt_impr_ustar_mean_only

Description: not_set

See also: objet_lecture (35)

Usage:

```
{
```

```

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

```

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

5.26.2 NUL

Description: not_set

See also: modele_turbulence_hyd_deriv (5.26)

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 ]
where

```

- **correction_visco_turb_pour_controle_pas_de_temps** : Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float*: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32): Keyword to set the wall law.
- **dt_impr_ustar** *float*: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1): This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float*: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.3 mod_turb_hyd_ss_maille

Description: Class for sub-grid turbulence model for Navier-Stokes equations.

See also: modele_turbulence_hyd_deriv (5.26) sous_maille_wale (5.26.5) sous_maille_smago (5.26.6) combinaison (5.26.7) longueur_melange (5.26.8) sous_maille (5.26.9) sous_maille_selectif_mod (5.26.10) sous_maille_selectif (5.26.13) sous_maille_elt (5.26.14) sous_maille_axi (5.26.16) sous_maille_smago_filtre (5.26.17) sous_maille_smago_dyn (5.26.18)

Usage:

```

mod_turb_hyd_ss_maille {

```

```

[ formulation_a_nb_points form_a_nb_points]
[ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]
[ correction_visco_turb_pour_controle_pas_de_temps ]
[ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
[ turbulence_paro turbulence_paro_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
}
where

```

- **formulation_a_nb_points** *form_a_nb_points* (5.26.4): The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homegeneity planes. Example for channel flows, planes parallel to the walls.
- **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* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.4 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 homegeneity planes. Example for channel flows, planes parallel to the walls.

See also: objet_lecture (35)

Usage:

nb dir1 dir2

where

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

5.26.5 sous_maille_wale

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

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

See also: `mod_turb_hyd_ss_maille` ([5.26.3](#))

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]
```

}

where

- **cw** *float*: The unique parameter (constant) of the WALE-model (by default value 0.5).
- **formulation_a_nb_points** *form_a_nb_points* ([5.26.4](#)) for inheritance: The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
 - volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 - volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 - scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 - arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when

permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.

- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.6 sous_maille_smago

Description: Smagorinsky sub-grid turbulence model.

$Nut = Cs1 * Cs1 * l * \sqrt{2 * S * S}$

$K = Cs2 * Cs2 * l * l * 2 * S$

See also: `mod_turb_hyd_ss_maille` (5.26.3)

Usage:

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

where

- **cs** *float*: This is an optional keyword and the value is used to set the constant used in the Smagorinsky model (This is currently only valid for Smagorinsky models and it is set to 0.18 by default) .
- **formulation_a_nb_points** *form_a_nb_points* (5.26.4) for inheritance: The structure fonction is calculated on `nb_points` and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
`volume` : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
`volume_sans_lissage` : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).

scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.

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

- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity; it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.7 combinaison

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

See also: `mod_turb_hyd_ss_maille` (5.26.3)

Usage:

```
combinaison {
    [ nb_var  n word1 word2 ... wordn]
    [ fonction  str]
    [ formulation_a_nb_points  form_a_nb_points]
    [ longueur_maille  str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre  float]
    [ turbulence_paro  turbulence_paro_base]
    [ dt_impr_ustar  float]
    [ dt_impr_ustar_mean_only  dt_impr_ustar_mean_only]
    [ nut_max  float]
}
```

where

- **nb_var** *n word1 word2 ... wordn*: Number and names of variables which will be used in the turbulent viscosity definition (by default 0)
- **fonction** *str*: Fonction for turbulent viscosity. X,Y,Z and variables defined previously can be used.

- **formulation_a_nb_points** *form_a_nb_points* (5.26.4) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str* into ['volume', 'volume_sans_lissage', 'scotti', 'arrete'] for inheritance: different ways to calculate the characteristic length may be specified :
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.8 longueur_melange

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

$$\nu_{t} = (\kappa y)^2 \frac{dU}{dy}$$

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 : $y = d_{max} \cdot \exp[-2 \cdot (dist_w - d_{max}) / d_{max}]$

See also: mod_turb_hyd_ss_maille (5.26.3)

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]
}
where

```

- **canalx** *float*: [height] : plane channel according to Ox direction (for the moment, formulation in the code relies on fixed height : H=2).
- **tuyauz** *float*: [diameter] : pipe according to Oz direction (for the moment, formulation in the code relies on fixed diameter : D=2).
- **verif_dparoi** *str*
- **dmax** *float*: Maximum distance.
- **fichier** *str*
- **fichier_ecriture_K_Eps** *str*: When a resume with k-epsilon model is envisaged, this keyword allows to generate external MED-format file with evaluation of k and epsilon quantities (based on eddy turbulent viscosity and turbulent characteristic length returned by mixing length model). The frequency of the MED file print is set equal to dt_impr_ustar. Moreover, k-eps MED field is automatically saved at the last time step. MED file is then used for resuming a K-Epsilon calculation with the Champ_Fonc_Med keyword.
- **formulation_a_nb_points** *form_a_nb_points* (5.26.4) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str* into ['volume', 'volume_sans_lissage', 'scotti', 'arrete'] for inheritance: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.

- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.9 sous_maille

Description: Structure sub-grid function model.

See also: `mod_turb_hyd_ss_maille` (5.26.3)

Usage:

```
sous_maille {
    [ formulation_a_nb_points form_a_nb_points ]
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete'] ]
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float ]
    [ turbulence_parois turbulence_parois_base ]
    [ dt_impr_ustar float ]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only ]
    [ nut_max float ]
}
```

where

- **formulation_a_nb_points** *form_a_nb_points* (5.26.4) for inheritance: The structure fonction is calculated on `nb_points` and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
`volume` : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
`volume_sans_lissage` : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
`scotti` : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
`arete` : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_parois** *turbulence_parois_base* (32) for inheritance: Keyword to set the wall law.

- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values ($U +, d+, u*$) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.10 sous_maille_selectif_mod

Description: Selective structure sub-grid function model (modified).

See also: `mod_turb_hyd_ss_maille` (5.26.3)

Usage:

```
sous_maille_selectif_mod {
    [ thi deuxentiers]
    [ canal floatentier]
    [ formulation_a_nb_points form_a_nb_points]
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
    [ turbulence_paro turbulence_paro_base]
    [ dt_impr_ustar float]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
    [ nut_max float]
}
```

where

- **thi** *deuxentiers* (5.26.11): For homogeneous isotropic turbulence (THI), two integers k_i and k_c are needed in VDF (not in VEF).
- **canal** *floatentier* (5.26.12): `h_dir_faces_paro`: For a channel flow, the half width h and the orientation of the wall `dir_faces_paro` are needed.
- **formulation_a_nb_points** *form_a_nb_points* (5.26.4) for inheritance: The structure function is calculated on `nb_points` and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is

calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.

- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.11 deuxentiers

Description: Two integers.

See also: `objet_lecture` (35)

Usage:

int1 int2

where

- **int1** *int*: First integer.
- **int2** *int*: Second integer.

5.26.12 floatentier

Description: A real and an integer.

See also: `objet_lecture` (35)

Usage:

the_float the_int

where

- **the_float** *float*: Real.
- **the_int** *int*: Integer.

5.26.13 sous_maille_selectif

Description: Selective structure sub-grid function model (a filter is applied to the structure function).

See also: `mod_turb_hyd_ss_maille` (5.26.3)

Usage:

sous_maille_selectif {

```

[ formulation_a_nb_points form_a_nb_points]
[ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]
[ correction_visco_turb_pour_controle_pas_de_temps ]
[ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
[ turbulence_paro turbulence_paro_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
}
where

```

- **formulation_a_nb_points** *form_a_nb_points* (5.26.4) for inheritance: The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.14 sous_maille_1elt

Description: Turbulence model sous_maille_1elt.

See also: mod_turb_hyd_ss_maille (5.26.3) sous_maille_1elt_selectif_mod (5.26.15)

Usage:

```
sous_maille_1elt {  
    [ formulation_a_nb_points form_a_nb_points ]  
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete'] ]  
    [ correction_visco_turb_pour_controle_pas_de_temps ]  
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float ]  
    [ turbulence_paro turbulence_paro_base ]  
    [ dt_impr_ustar float ]  
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only ]  
    [ nut_max float ]  
}
```

where

- **formulation_a_nb_points** *form_a_nb_points* (5.26.4) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.15 sous_maille_1elt_selectif_mod

Description: Turbulence model sous_maille_1elt_selectif_mod.

See also: sous_maille_1elt (5.26.14)

Usage:

```
sous_maille_1elt_selectif_mod {  
    [ formulation_a_nb_points form_a_nb_points]  
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]  
    [ correction_visco_turb_pour_controle_pas_de_temps ]  
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]  
    [ turbulence_parois turbulence_parois_base]  
    [ dt_impr_ustar float]  
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]  
    [ nut_max float]  
}  
where
```

- **formulation_a_nb_points** *form_a_nb_points* (5.26.4) 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_parois** *turbulence_parois_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.

- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.16 sous_maille_axi

Description: Structure sub-grid function turbulence model available in cylindrical co-ordinates.

See also: `mod_turb_hyd_ss_maille` (5.26.3)

Usage:

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

where

- **formulation_a_nb_points** *form_a_nb_points* (5.26.4) 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 homegeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value

is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.

- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.17 sous_maille_smago_filtre

Description: Smagorinsky sub-grid turbulence model should be used with low-filter.

See also: mod_turb_hyd_ss_maille (5.26.3)

Usage:

```
sous_maille_smago_filtre {
    [ formulation_a_nb_points form_a_nb_points ]
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete'] ]
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float ]
    [ turbulence_paroit turbulence_paroit_base ]
    [ dt_impr_ustar float ]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only ]
    [ nut_max float ]
}
```

where

- **formulation_a_nb_points** *form_a_nb_points* (5.26.4) for inheritance: The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paroit** *turbulence_paroit_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.

- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.18 sous_maille_smago_dyn

Description: Dynamic Smagorinsky sub-grid turbulence model (available in VDF discretization only).

See also: `mod_turb_hyd_ss_maille` (5.26.3)

Usage:

```
sous_maille_smago_dyn {
    [ stabilise str into ['6_points', 'moy_euler', 'plans_paralleles']]
    [ nb_points int]
    [ formulation_a_nb_points form_a_nb_points]
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
    [ turbulence_paroit turbulence_paroit_base]
    [ dt_impr_ustar float]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
    [ nut_max float]
}
```

where

- **stabilise** *str into ['6_points', 'moy_euler', 'plans_paralleles']*
- **nb_points** *int*
- **formulation_a_nb_points** *form_a_nb_points* (5.26.4) for inheritance: The structure function is calculated on `nb_points` and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.

- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.19 mod_turb_hyd_rans

Description: Class for RANS turbulence model for Navier-Stokes equations.

See also: `modele_turbulence_hyd_deriv` (5.26) `k_epsilon` (5.26.20) `K_Epsilon_Realisable` (5.26.27)

Usage:

```
mod_turb_hyd_rans {
    [ eps_min float]
    [ eps_max float]
    [ k_min float]
    [ quiet ]
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
    [ turbulence_paro turbulence_paro_base]
    [ dt_impr_ustar float]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
    [ nut_max float]
}
```

where

- **eps_min** *float*: Lower limitation of epsilon (default value 1.e-10).
- **eps_max** *float*: Upper limitation of epsilon (default value 1.e+10).
- **k_min** *float*: Lower limitation of k (default value 1.e-10).
- **quiet** : To disable printing of information about k and epsilon.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32) for inheritance: Keyword to set the wall law.

- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values ($U +$, $d+$, u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.20 k_epsilon

Description: Turbulence model (k-eps).

See also: `mod_turb_hyd_rans` (5.26.19)

Usage:

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

where

- **transport_k_epsilon** *transport_k_epsilon* (5.40): Keyword to define the (k-eps) transportation equation.
- **modele_fonc_bas_reynolds** *modele_fonction_bas_reynolds_base* (5.26.21): This keyword is used to set the bas Reynolds model used.
- **cmu** *float*: Keyword to modify the Cmu constant of k-eps model : $Nut = Cmu * k^2 / \epsilon$ Default value is 0.09
- **prandtl_k** *float*: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float*: Keyword to change the Pre value (default 1.3).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **eps_max** *float* for inheritance: Upper limitation of epsilon (default value 1.e+10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **quiet** for inheritance: To disable printing of information about k and epsilon.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary

flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.

- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.26.21 modele_fonction_bas_reynolds_base

Description: `not_set`

See also: `objet_lecture` (35) `Laundry_Sharma` (5.26.22) `Lam_Bremhorst` (5.26.23) `Jones_Laundry` (5.26.26)

Usage:

5.26.22 Laundry_Sharma

Description: Model described in 'Laundry, B. E. and Sharma, B. I. (1974), Application of the Energy-Dissipation Model of Turbulence to the Calculation of Flow Near a Spinning Disc, Letters in Heat and Mass Transfer, Vol. 1, No. 2, pp. 131-138.'

See also: `modele_fonction_bas_reynolds_base` (5.26.21)

Usage:

5.26.23 Lam_Bremhorst

Description: Model described in 'C.K.G.Lam and K.Bremhorst, A modified form of the k- epsilon model for predicting wall turbulence, ASME J. Fluids Engng., Vol.103, p456, (1981)'. Only in VEF.

See also: `modele_fonction_bas_reynolds_base` (5.26.21) `EASM_Baglietto` (5.26.24) `standard_KEps` (5.26.25)

Usage:

```
Lam_Bremhorst {
    [ fichier_distance_paro str]
    [ reynolds_stress_isotrope int]
```

```
}
```

where

- **fichier_distance_paro** *str*: refer to `distance_paro` keyword
- **reynolds_stress_isotrope** *int*: keyword for isotropic Reynolds stress

5.26.24 EASM_Baglietto

Description: Model described in ' E. Baglietto and H. Ninokata , A turbulence model study for simulating flow inside tight lattice rod bundles, Nuclear Engineering and Design, 773–784 (235), 2005. '

See also: Lam_Bremhorst ([5.26.23](#))

Usage:

```
EASM_Baglietto {  
    [ fichier_distance_paroi str]  
    [ reynolds_stress_isotrope int]  
}
```

where

- **fichier_distance_paro**i *str* for inheritance: refer to distance_paro keyword
- **reynolds_stress_isotrope** *int* for inheritance: keyword for isotropic Reynolds stress

5.26.25 standard_KEps

Description: Model described in ' E. Baglietto , CFD and DNS methodologies development for fuel bundle simulations, Nuclear Engineering and Design, 1503–1510 (236), 2006. '

See also: Lam_Bremhorst ([5.26.23](#))

Usage:

```
standard_KEps {  
    [ fichier_distance_paroi str]  
    [ reynolds_stress_isotrope int]  
}
```

where

- **fichier_distance_paro**i *str* for inheritance: refer to distance_paro keyword
- **reynolds_stress_isotrope** *int* for inheritance: keyword for isotropic Reynolds stress

5.26.26 Jones_Launders

Description: Model described in ' Jones, W. P. and Launders, B. E. (1972), The prediction of laminarization with a two-equation model of turbulence, Int. J. of Heat and Mass transfer, Vol. 15, pp. 301-314.'

See also: modele_fonction_bas_reynolds_base ([5.26.21](#))

Usage:

5.26.27 K_Epsilon_Realisable

Description: Realizable K-Epsilon Turbulence Model.

See also: mod_turb_hyd_rans ([5.26.19](#))

Usage:

```
K_Epsilon_Realisable {
```

```

transport_k_epsilon_realisable str
modele_fonc_realisable modele_fonc_realisable_base
prandtl_k float
prandtl_eps float
[ eps_min float]
[ eps_max float]
[ k_min float]
[ quiet ]
[ correction_visco_turb_pour_controle_pas_de_temps ]
[ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
[ turbulence_paroit turbulence_paroit_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
}

```

where

- **transport_k_epsilon_realisable** *str*: Keyword to define the realisable (k-eps) transportation equation.
- **modele_fonc_realisable** *modele_fonc_realisable_base* (10.2): This keyword is used to set the model used
- **prandtl_k** *float*: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float*: Keyword to change the Pre value (default 1.3)
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **eps_max** *float* for inheritance: Upper limitation of epsilon (default value 1.e+10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **quiet** for inheritance: To disable printing of information about k and epsilon.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paroit** *turbulence_paroit_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.26.1) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).

5.27 deuxmots

Description: Two words.

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

Usage:

mot_1 mot_2

where

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

5.28 floatfloat

Description: Two reals.

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

Usage:

a b

where

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

5.29 traitement_particulier

Description: Auxiliary class to post-process particular values.

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

Usage:

aco trait_part acof

where

- **aco** *str* into ['']: Opening curly bracket.
- **trait_part** *traitement_particulier_base* ([5.29.1](#)): Type of *traitement_particulier*.
- **acof** *str* into ['']: Closing curly bracket.

5.29.1 traitement_particulier_base

Description: Basic class to post-process particular values.

See also: [objet_lecture \(35\)](#) [temperature \(5.29.2\)](#) [canal \(5.29.3\)](#) [ec \(5.29.4\)](#) [thi \(5.29.5\)](#) [chmoy_faceperio \(5.29.7\)](#) [profils_thermo \(5.29.8\)](#) [brech \(5.29.9\)](#) [ceg \(5.29.10\)](#)

Usage:

5.29.2 temperature

Description: not_set

See also: [traitement_particulier_base \(5.29.1\)](#)

Usage:

temperature {

bord *str*

direction *int*

}

where

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

5.29.3 canal

Description: Keyword for statistics on a periodic plane channel.

See also: `traitement_particulier_base` ([5.29.1](#))

Usage:

```
canal {
    [ dt_impr_moy_spat float]
    [ dt_impr_moy_temp float]
    [ debut_stat float]
    [ fin_stat float]
    [ pulsation_w float]
    [ nb_points_par_phase int]
    [ reprise str]
}
```

where

- **dt_impr_moy_spat** *float*: Period to print the spatial average (default value is 1e6).
- **dt_impr_moy_temp** *float*: Period to print the temporal average (default value is 1e6).
- **debut_stat** *float*: Time to start the temporal averaging (default value is 1e6).
- **fin_stat** *float*: Time to end the temporal averaging (default value is 1e6).
- **pulsation_w** *float*: Pulsation for phase averaging (in case of pulsating forcing term) (no default value).
- **nb_points_par_phase** *int*: Number of samples to represent phase average all along a period (no default value).
- **reprise** *str*: `val_moy_temp_XXXXXX.sauv` : Keyword to resume a calculation with previous averaged quantities.

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

5.29.4 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` ([5.29.1](#))

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`.

5.29.5 thi

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

See also: `traitement_particulier_base` (5.29.1) `thi_thermo` (5.29.6)

Usage:

```

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

```

- **init_Ec** *int*: Keyword to renormalize initial velocity so that kinetic energy equals to the value given by keyword `val_Ec`.
- **val_Ec** *float*: Keyword to impose a value for kinetic energy by velocity renormalized if `init_Ec` value is 1.
- **facon_init** *int into [0, 1]*: Keyword to specify how kinetic energy is computed (0 or 1).
- **calc_spectre** *int into [0, 1]*: Calculate or not the spectrum of kinetic energy.
Files called `Sorties_THI` are written with inside four columns :
time:t global_kinetic_energy:Ec enstrophy:D skewness:S
If `calc_spectre` is set to 1, a file `Sorties_THI2_2` is written with three columns :
time:t kinetic_energy_at_kc=32 enstrophy_at_kc=32
If `calc_spectre` is set to 1, a file `spectre_XXXXX` is written with two columns at each time `xxxxx` :
frequency:k energy:E(k).
- **periode_calc_spectre** *float*: Period for calculating spectrum of kinetic energy
- **3D** *int into [0, 1]*: Calculate or not the 3D spectrum
- **1D** *int into [0, 1]*: Calculate or not the 1D spectrum
- **conservation_Ec** : If set to 1, velocity field will be changed as to have a constant kinetic energy (default 0)
- **longueur_boite** *float*: Length of the calculation domain

5.29.6 thi_thermo

Description: Treatment for the temperature field.

It offers the possibility to :

- evaluate the probability density function on temperature field,
- give in a file the temperature field for a future spectral analysis,
- monitor the evolution of the max and min temperature on the whole domain.

See also: thi ([5.29.5](#))

Usage:

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

where

- **init_Ec** *int* for inheritance: Keyword to renormalize initial velocity so that kinetic energy equals to the value given by keyword **val_Ec**.
- **val_Ec** *float* for inheritance: Keyword to impose a value for kinetic energy by velocity renormalized if **init_Ec** value is 1.
- **facon_init** *int into [0, 1]* for inheritance: Keyword to specify how kinetic energy is computed (0 or 1).
- **calc_spectre** *int into [0, 1]* for inheritance: Calculate or not the spectrum of kinetic energy.
Files called Sorties_THI are written with inside four columns :
time:t global_kinetic_energy:Ec enstrophy:D skewness:S
If **calc_spectre** is set to 1, a file Sorties_THI2_2 is written with three columns :
time:t kinetic_energy_at_kc=32 enstrophy_at_kc=32
If **calc_spectre** is set to 1, a file spectre_XXXXX is written with two columns at each time XXXXX :
frequency:k energy:E(k).
- **periode_calc_spectre** *float* for inheritance: Period for calculating spectrum of kinetic energy
- **3D** *int into [0, 1]* for inheritance: Calculate or not the 3D spectrum
- **1D** *int into [0, 1]* for inheritance: Calculate or not the 1D spectrum
- **conservation_Ec** for inheritance: If set to 1, velocity field will be changed as to have a constant kinetic energy (default 0)
- **longueur_boite** *float* for inheritance: Length of the calculation domain

5.29.7 chmoy_faceperio

Description: non documente

See also: traitement_particulier_base ([5.29.1](#))

Usage:

```
chmoy_faceperio bloc  
where
```

- **bloc** *bloc_lecture* (3.6)

5.29.8 profils_thermo

Description: non documente

See also: traitement_particulier_base (5.29.1)

Usage:

profils_thermo bloc

where

- **bloc** *bloc_lecture* (3.6)

5.29.9 brech

Description: non documente

See also: traitement_particulier_base (5.29.1)

Usage:

brech bloc

where

- **bloc** *bloc_lecture* (3.6)

5.29.10 ceg

Description: Keyword for a CEG (Gas Entrainment Criteria) calculation. An objective is deepening gas entrainment on the free surface. Numerical analysis can be performed to predict the hydraulic and geometric conditions that can handle gas entrainment from the free surface.

See also: traitement_particulier_base (5.29.1)

Usage:

ceg {

frontiere *str*

t_deb *float*

[**t_fin** *float*]

[**dt_post** *float*]

haspi *float*

[**debug** *int*]

[**areva** *ceg_areva*]

[**cea_jaea** *ceg_cea_jaea*]

}

where

- **frontiere** *str*: To specify the boundaries conditions representing the free surfaces
- **t_deb** *float*: value of the CEG's initial calculation time
- **t_fin** *float*: not_set time during which the CEG's calculation was stopped
- **dt_post** *float*: periode refers to the printing period, this value is expressed in seconds

- **haspi** *float*: The suction height required to calculate AREVA's criterion
- **debug** *int*
- **areva** *ceg_areva* (5.29.11): AREVA's criterion
- **cea_jaea** *ceg_cea_jaea* (5.29.12): CEA_JAEA's criterion

5.29.11 ceg_areva

Description: not_set

See also: objet_lecture (35)

Usage:

```
{
    [ c float]
```

```
}
```

where

- **c** *float*

5.29.12 ceg_cea_jaea

Description: not_set

See also: objet_lecture (35)

Usage:

```
{
    [ normalise int]
    [ nb_mailles_mini int]
    [ min_critere_q_sur_max_critere_q float]
```

```
}
```

where

- **normalise** *int*: renormalize (1) or not (0) values alpha and gamma
- **nb_mailles_mini** *int*: Sets the minimum number of cells for the detection of a vortex.
- **min_critere_q_sur_max_critere_q** *float*: Is an optional keyword used to correct the minimum values of Q's criterion taken into account in the detection of a vortex

5.30 navier_stokes_phase_field

Description: Navier Stokes equation for the Phase Field problem.

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

See also: navier_stokes_standard (5.32)

Usage:

navier_stokes_phase_field obj Lire obj {

```
    approximation_de_boussinesq str into ['oui', 'non']
    viscosite_dynamique_constant str into ['oui', 'non']
    gravite n x1 x2 ... xn
```

```

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

```

- **approximation_de_boussinesq** str into ['oui', 'non']: To use or not the Boussinesq approximation.
- **viscosite_dynamique_constante** str into ['oui', 'non']: To use or not a viscosity which will depends on concentration C (in fact, C is the unknown of Cahn-Hilliard equation).
- **gravite** n x1 x2 ... xn: Keyword to define gravity in the case Boussinesq approximation is not used.
- **methode_calcul_pression_initiale** str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']: for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : avec_les_cl (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), avec_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection_initiale** int for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur_pression** solveur_sys_base (10.16) for inheritance: Linear pressure system resolution method.
- **solveur_bar** solveur_sys_base (10.16) 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 (5.27) 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 (5.28) for inheritance: value factor : this keyword is intended to minimise the number of iterations during the pressure system resolution. The convergence criteria during this step ('seuil' in solveur_pression) is dynamically adapted according to the mass conservation. At tn , the linear system Ax=B is considered as solved if the residual ||Ax-B||<seuil(tn). For tn+1, the threshold value seuil(tn+1) will be 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* (5.29) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.31 navier_stokes_qc

Description: Navier-Stokes equations under low Mach number.

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

See also: navier_stokes_standard (5.32)

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 equations) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (10.16) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (10.16) 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* (5.27) 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* (5.28) for inheritance: value factor : this keyword is intended to minimise the number of iterations during the pressure system resolution. The convergence criteria during this step ('seuil' in solveur_pression) is dynamically adapted according to the mass conservation. At tn , the linear system Ax=B is considered as solved if the residual ||Ax-B||<seuil(tn). For tn+1, the threshold value seuil(tn+1) will be 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* (5.29) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.

```
Navier_Sokes_Standard
```

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

5.32 navier_stokes_standard

Description: Navier-Stokes equations.

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

See also: eqn_base (5.23) navier_stokes_qc (5.31) navier_stokes_turbulent (5.33) navier_stokes_phase_field (5.30)

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. Op-

tions are : avec_les_cl (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), avec_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.

- **projection_initiale** *int*: Keyword to suppress, if boolean equals 0, the initial projection which checks $\text{DivU}=0$. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (10.16): Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (10.16): 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* (5.27): 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* (5.28): 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* (5.29): Keyword to post-process particular values.
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time `t0` and `t1`.

`Navier_Sokes_Standard`

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

5.33 navier_stokes_turbulent

Description: Navier-Stokes equations as well as the associated turbulence model equations.

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

See also: `navier_stokes_standard` (5.32) `navier_stokes_ft_disc` (5.24) `navier_stokes_turbulent_qc` (5.34)

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* (5.26): Turbulence model for Navier-Stokes equations.
- **methode_calcul_pression_initiale** *str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']* for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : `avec_les_cl` (default option `lapP=0` is solved with Neuman boundary conditions on pressure if any), `avec_sources` (`lapP=f` is solved with Neuman boundaries conditions and `f` integrating the source terms of the Navier-Stokes equations) and `avec_sources_et_operateurs` (`lapP=f` is solved as with the previous option `avec_sources` but `f` integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks `DivU=0`. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (10.16) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (10.16) 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* (5.27) 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* (5.28) for inheritance: value factor : this keyword is intended to minimise the number of iterations during the pressure system resolution. The convergence criteria during this step ('seuil' in solveur_pression) is dynamically adapted according to the mass conservation. At tn , the linear system $Ax=B$ is considered as solved if the residual $\|Ax-B\| < \text{seuil}(tn)$. For tn+1, the threshold value $\text{seuil}(tn+1)$ will be evaluated as:
 If ($\text{lmax}(\text{DivU}) * dt < \text{value}$)
 Seuil(tn+1)= Seuil(tn)*factor
 Else
 Seuil(tn+1)= Seuil(tn)*factor
 Endif
 The first parameter (value) is the mass evolution the user is ready to accept per timestep, and the second one (factor) is the factor of evolution for 'seuil' (for example 1.1, so 10)
- **traitement_particulier** *traitement_particulier* (5.29) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limite** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
 Navier_Sokes_Standard
 { equation_non_resolue (t>t0)*(t<t1) }

5.34 navier_stokes_turbulent_qc

Description: Navier-Stokes equations under low Mach number as well as the associated turbulence model equations.

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

See also: `navier_stokes_turbulent` (5.33)

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* (5.26) 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 equations) and *avec_sources_et_operateurs* ($\text{lapP}=f$ is solved as with the previous option *avec_sources* but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks $\text{DivU}=0$. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (10.16) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (10.16) 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* (5.27) 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* (5.28) for inheritance: *value* factor : this keyword is intended to minimise the number of iterations during the pressure system resolution. The convergence criteria during this step ('seuil' in `solveur_pression`) is dynamically adapted according to the mass conservation. At t_n , the linear system $Ax=B$ is considered as solved if the residual $\|Ax-B\| < \text{seuil}(t_n)$. For t_{n+1} , the threshold value $\text{seuil}(t_{n+1})$ will be evaluated as:
If ($\text{lmax}(\text{DivU}) * \text{dt} < \text{value}$)

```

Seuil(tn+1)= Seuil(tn)*factor
Else
Seuil(tn+1)= Seuil(tn)*factor
Endif

```

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

- **traitement_particulier** *traitement_particulier* (5.29) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.

```

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

```

5.35 transport_interfaces_ft_disc

Description: Interface tracking equation for Front-Tracking problem in the discontinuous version.

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

See also: eqn_base (5.23)

Usage:

```

transport_interfaces_ft_disc obj Lire obj {
    [ initial_conditions|conditions_initiales bloc_lecture]
    [ methode_transport methode_transport_deriv]
    [ iterations_correction_volume int]
    [ n_iterations_distance int]
    [ maillage str]

```

```

[ remaillage bloc_lecture_remaillage]
[ collisions str]
[ methode_interpolation_v str into ['valeur_a_elem', 'vdf_lineaire']]
[ volume_impose_phase_1 float]
[ parcours_interface parcours_interface]
[ interpolation_repere_local ]
[ interpolation_champ_face interpolation_champ_face_deriv]
[ n_iterations_interpolation_ibc int]
[ type_vitesse_imposee str into ['uniforme', 'analytique']]
[ nombre_facettes_retenues_par_cellule int]
[ seuil_convergence_uzawa float]
[ nb_iteration_max_uzawa int]
[ injecteur_interfaces str]
[ vitesse_imposee_regularisee int]
[ indic_faces_modifiee bloc_lecture]
[ distance_projete_faces str into ['simplifiee', 'initiale', 'modifiee']]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
[ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
}
where

```

- **initial_conditions|conditions_initiales** *bloc_lecture* (3.6): The keyword `conditions_initiales` is used to define the shape of the initial interfaces through the zero level-set of a function, or through a mesh `fichier_geom`. Indicator function is set to 0, that is `fluide0`, where the function is negative; indicator function is set to 1, that is `fluide1`, where the function is positive; the interfaces are the level-set 0 of that function:

```

conditions_initiales { fonction
  ( -((x-0.002)2 + (y-0.002)2 + z2 - (0.00125)2)) * ((x-0.005)2 + (y-0.007)2 + z2 (0.00150)2)) *
  (0.020 - z))
}

```

In the above example, there are three interfaces: two bubbles in a liquid with a free surface. One bubble has a radius of 0.00125, i.e. 1.25 mm, and its center is {0.002, 0.002, 0.000}. The other bubble has a radius of 0.00150, i.e. 1.5 mm, and its center is {0.005, 0.007, 0.000}. The free surface is above the two bubble, at a level $z=0.02$.

Additional feature in this block concerns the keywords `ajout_phase0` and `ajout_phase1`. They can be used to simplify the composition of different interfaces. When using these keywords, the initial function defines the indicator function; `ajout_phase0` and `ajout_phase1` are used to modify this initial field. Each time `ajout_phase0` is used, the field is untouched where the function is positive whereas the indicator field is set to 0 where the function is negative. The keyword `ajout_phase1` has the symmetrical use, keeping the field value where the function is negative and setting the indicator field to 1 where the function is positive. The previous example can also be written:

```

conditions_initiales {
fonction z-0.020 , NL fonction ajout_phase1 (x - 0.002)2 + (y - 0.002)2 + z2 - (0.00125)2 ,

```


fonction ajout_phase1 $(x - 0.005)^2 + (y - 0.007)^2 + z^2 - (0.00150)^2$
}

- **methode_transport** *methode_transport_deriv* (5.36): Method of transport of interface.
- **iterations_correction_volume** *int*: Keyword to specify the number of iterations requested for the correction process that can be used to keep the volume of the phases constant during the transport process.
- **n_iterations_distance** *int*: Keyword to specify the number of iterations requested for the smoothing process of computing the field corresponding to the signed distance to the interfaces and located at the center of the Eulerian elements. This smoothing is necessary when there are more Lagrangian nodes than Eulerian two-phase cells.
- **maillage** *str*: This optional block is used to specify that we want a Gnuplot drawing of the initial mesh. There is only one keyword, *niveau_plot*, that is used only to define if a Gnuplot drawing is active (value 1) or not active (value -1). By default, skipping the block will produce non Gnuplot drawing. This option is to be used only in a debug process.
- **remaillage** *bloc_lecture_remaillage* (5.37): This block is used to specify the operations that are used to keep the solid interfaces in a proper condition. The remaillage block only contains parameter's values.
- **collisions** *str*: This block is used to specify the operations that are used when a collision occurs between two parts of interfaces. When this occurs, it is necessary to build a new mesh that has locally a clear definition of what is inside and what is outside of the mesh. The collisions can either be active or inactive. If the collisions are active (highly recommended), the keyword *juric_pour_tout* indicates that the Juric level-set reconstruction method will be used to re-create the new mesh after each coalescence or breakup. The next line (*type_remaillage*) is used to state whose field will be used for the level-set computation. Main option is Juric, a remeshing that is compatible with parallel computing. When using Juric level-set remeshing, the source field (*source_isevaleur*) that is used to compute the level-sets is then defined. It can be either the indicator function (*indicatrice*), a choice which is the default one and the most robust, or a geometrical distance computed from the mesh at the beginning of the time step (*fonction_distance*), a choice that may be more accurate in specific situations.

Type_remaillage Thomas is an enhancement of the Juric global remeshing algorithm designed to compensate for mass loss during remeshing. The mesh is always reconstructed with the indicator function (not with the distance function). After having reconstructed the mesh with the Juric algorithm, the difference between the old indicator function (before remeshing) and the new indicator function is computed. The differences occurring at a distance below or equal to *N* elements from the interface are summed up and used to move the interface in the normal direction. The displacement of the interface is such that the volume of each phase after displacement is equal to the volume of the phase before remeshing. *N* (default value 1) must be smaller than *n_iterations_distance* (suggested value: 2).

An alternate choice for the remeshing type (*type_remaillage*) is *collision_seq*, which is more complex and tries to sew the two meshes that have collided, once the collision zone has been removed. This algorithm does not work in parallel computation.

- **methode_interpolation_v** *str* into [*'valeur_a_elem'*, *'vdf_lineaire'*]: In this block, two keywords are possible for method to select the way the interpolation is performed. With the choice *valeur_a_elem* the speed of displacement of the nodes of the interfaces is the velocity at the center of the Eulerian element in which each node is located at the beginning of the time step. This choice is the default interpolation method. The choice *VDF_lineaire* is only available with a VDF discretization (VDF). In this case, the speed of displacement of the nodes of the interfaces is linearly interpolated on the 4 (in 2D) or the 6 (in 3D) Eulerian velocities closest the location of each node at the beginning of the time step. In peculiar situation, this choice may provide a better interpolated value. Of course, this choice is not available with a VEF discretization (VEFPreP1B).
- **volume_impose_phase_1** *float*: this keyword is used to specify the volume of one phase to keep the volume of the phases constant during the remeshing process. It is an alternate solution to trouble in mass conservation. This option is mainly realistic when only one inclusion of phase 1 is present in the domain. In most other situations, the *iterations_correction_volume* keyword seems easier to

justify. The volume to be kept is in m3 and should agree with initial condition.

- **parcours_interface** *parcours_interface* (5.38): *Parcours_interface* allows you to configure the algorithm that computes the surface mesh to volume mesh intersection. This algorithm has some serious trouble when the surface mesh points coincide with some faces of the volume mesh. Effects are visible on the indicator function, in VDF when a plane interface coincides with a volume mesh surface. To overcome these problems, the keyword *correction_parcours_thomas* keyword can be used: it allows the algorithm to slightly move some mesh points. This algorithm is experimental and is NOT activated by default.
- **interpolation_repere_local** : Triggers a new transport algorithm for the interface: the velocity vector of lagrangian nodes is computed in the moving frame of reference of the center of each connex component, in such a way that relative displacements of nodes within a connex component of the lagrangian mesh are minimized, hence reducing the necessity of barycentering, smooting and local remeshing. Very efficient for bubbly flows.
- **interpolation_champ_face** *interpolation_champ_face_deriv* (5.39): It is possible to compute the imposed velocity for the solid-fluid interface by direct affectation (*interpolation_scheme* would be set to *base*) or by multi-linear interpolation (*interpolation_scheme* would be set to *lineaire*). The default value is *base*.
- **n_iterations_interpolation_ibc** *int*: Useful only with *interpolation_champ_face* positioned to *lineaire*. Set the value concerning the width of the region of the linear interpolation. For the Penalized Direct Forcing model, a value equals to 1 is enough.
- **type_vitesse_imposee** *str* into [*'uniforme'*, *'analytique'*]: Useful only with *interpolation_champ_face* positioned to *lineaire*. Value of the keyword is *uniforme* (for an uniform solid-fluid interface's velocity, i.e. zero for instance) or *analytique* (for an analytic expression of the solid-fluid interface's velocity depending on the spatial coordinates). The default value is *uniforme*.
- **nombre_facettes_retenues_par_cellule** *int*: Keyword to specify the default number (3) of facets per cell used to describe the geometry of the solid-solid interface. This number should be increased if the geometry of the solid-solid interface is complex in each cell (eulerian mesh too coarse for example).
- **seuil_convergence_uzawa** *float*: Optional option to change the default value (10-8) of the threshold convergence for the Uzawa algorithm if used in the Penalized Direct Forcing model. Sometime, the value should be decreased to insure a better convergence to force equality between sequential and parallel results.
- **nb_iteration_max_uzawa** *int*: Optional option to change the default value (10-8) of the threshold convergence for the Uzawa algorithm if used in the Penalized Direct Forcing model. Sometime, the value should be decreased to insure a better convergence to force equality between sequential and parallel results.
- **injecteur_interfaces** *str*
- **vitesse_imposee_regularisee** *int*
- **indic_faces_modifiee** *bloc_lecture* (3.6)
- **distance_projeete_faces** *str* into [*'simplifiee'*, *'initiale'*, *'modifiee'*]
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limite** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.36 methode_transport_deriv

Description: Basic class for method of transport of interface.

See also: objet_lecture (35) loi_horaire (5.36.1) vitesse_imposee (5.36.2) vitesse_interpolee (5.36.3)

Usage:

methode_transport_deriv

5.36.1 loi_horaire

Description: not_set

See also: methode_transport_deriv (5.36)

Usage:

loi_horaire nom_loi

where

- **nom_loi** *str*

5.36.2 vitesse_imposee

Description: Class to specify that the speed of displacement of the nodes of the interfaces is imposed with an analytical formula.

See also: methode_transport_deriv (5.36)

Usage:

vitesse_imposee val

where

- **val** *word1 word2 (word3)*: Analytical formula.

5.36.3 vitesse_interpolee

Description: Class to specify that the interpolation will use the velocity field of the Navier-Stokes equation named *val* to compute the speed of displacement of the nodes of the interfaces.

See also: *methode_transport_deriv* (5.36)

Usage:

vitesse_interpolee *val*
where

- **val** *str*: Navier-Stokes equation.

5.37 bloc_lecture_remaillage

Description: Parameters for remeshing.

See also: *objet_lecture* (35)

Usage:

```
{  
  [ pas float]  
  [ pas_lissage float]  
  [ nb_iter_remaillage int]  
  [ nb_iter_barycentrage int]  
  [ relax_barycentrage float]  
  [ critere_arete float]  
  [ critere_remaillage float]  
  [ impr float]  
  [ facteur_longueur_ideale float]  
  [ nb_iter_correction_volume int]  
  [ seuil_dvolume_residuel float]  
  [ lissage_courbure_coeff float]  
  [ lissage_courbure_iterations int]  
  [ lissage_courbure_iterations_systematique int]  
  [ lissage_courbure_iterations_si_remaillage int]  
  [ critere_longueur_fixe float]  
}
```

where

- **pas** *float*: This keyword has default value -1.; when it is set to a negative value there is no remeshing. It is the time step in second (physical time) between two operations of remeshing.
- **pas_lissage** *float*: This keyword has default value -1.; when it is set to a negative value there is no smoothing of mesh. It is the time step in second (physical time) between two operations of smoothing of the mesh.
- **nb_iter_remaillage** *int*: This keyword has default value 0; when it is set to the zero value there is no remeshing. It is the number of iterations performed during a remeshing process.
- **nb_iter_barycentrage** *int*: This keyword has default value 0; when it is set to the zero value there is no operation of barycentrage. The barycentrage operation consists in moving each node of the mesh tangentially to the mesh surface and in a direction that let it closer the center of gravity of its neighbors. If **relax_barycentrage** is set to 1, the node is move to the center of gravity. For values lower than unity, the motion is limited to the corresponding fraction. The parameter **nb_iter_barycentrage** is the number of iteration of these node displacements.

- **relax_barycentrage** *float*: This keyword has default value 0; when it is set to the zero value there is no motion of the nodes. When $0 < \text{relax_barycentrage} \leq 1$, this parameter provides the relaxation ratio to be used in the barycentrage operation described for the keyword `nb_iter_barycentrage`.
- **critere_arete** *float*: This keyword is used to compute two sub-criteria : the minimum and the maximum edge length ratios used in the process of obtaining edges of length close to `critere_longueur_fixe`. Their respective values are set to $(1 - \text{critere_arete})^{**2}$ and $(1 + \text{critere_arete})^{**2}$. The default values of the minimum and the maximum are set respectively to 0.5 and 1.5. When an edge is longer than $\text{critere_longueur_fixe} * (1 + \text{critere_arete})^{**2}$, the edge is cut into two pieces; when its length is smaller than $\text{critere_longueur_fixe} * (1 - \text{critere_arete})^{**2}$, this edge has to be suppressed.
- **critere_remaillage** *float*: This keyword was previously used to compute two sub-criteria : the minimum and the maximum length used in the process of remeshing. Their respective values are set to $(1 - \text{critere_remaillage})^{**2}$ and $(1 + \text{critere_remaillage})^{**2}$. The default values of the minimum and the maximum are set respectively to 0.2 and 1.7. There are currently not used in data files.
- **impr** *float*: This keyword is followed by a value that specify the printing time period given. The default value is -1, which means no printing.
- **facteur_longueur_ideale** *float*: This keyword is used to set a ratio between edge length and the cube root of volume cell for the remeshing process. The default value is 1.0.
- **nb_iter_correction_volume** *int*: This keyword give the maximum number of iterations to be performed trying to satisfy the criterion `seuil_dvolume_residuel`. The default value is 0, which means no iteration.
- **seuil_dvolume_residuel** *float*: This keyword give the error volume (in m3) that is accepted to stop the iterations performed to keep the volume constant during the remeshing process. The default value is 0.0.
- **lissage_courbure_coeff** *float*: This keyword is used to specify the diffusion coefficient used in the diffusion process of the curvature in the curvature smoothing process with a time step. The default value is 0.05. That value usually provides a stable process. Too small values do not stabilize enough the interface, especially with several Lagrangian nodes per Eulerian cell. Too high values induce an additional macroscopic smoothing of the interface that should physically come from the surface tension and not from this numerical smoothing.
- **lissage_courbure_iterations** *int*: This keyword is used to specify the number of iterations to perform the curvature smoothing process. The default value is 1.
- **lissage_courbure_iterations_systematique** *int*: These keywords allow a finer control than the previous `lissage_courbure_iterations` keyword. N1 iterations are applied systematically at each timestep. For proper DNS computation, N1 should be set to 0.
- **lissage_courbure_iterations_si_remaillage** *int*: N2 iterations are applied only if the local or the global remeshing effectively changes the lagrangian mesh connectivity.
- **critere_longueur_fixe** *float*: This keyword is used to specify the ideal edge length for a remeshing process. The default value is -1., which means that the remeshing does not try to have all edge lengths to tend towards a given value.

5.38 parours_interface

Description: allows you to configure the algorithm that computes the surface mesh to volume mesh intersection. This algorithm has some serious trouble when the surface mesh points coincide with some faces of the volume mesh. Effects are visible on the indicator function, in VDF when a plane interface coincides with a volume mesh surface.

To overcome these problems, the keyword `correction_parours_thomas` keyword can be used: it allows the algorithm to slightly move some mesh points. This algorithm, which is experimental and is NOT activated by default, triggers a correction that avoids some errors in the computation of the indicator function for surface meshes that exactly cross some eulerian mesh edges (strongly suggested !).

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

Usage:

```
{
    [ correction_parcours_thomas ]
}
where
```

- **correction_parcours_thomas**

5.39 interpolation_champ_face_deriv

Description: not_set

See also: objet_lecture (35) base (5.39.1) lineaire (5.39.2)

Usage:

5.39.1 base

Description: not_set

See also: interpolation_champ_face_deriv (5.39)

Usage:

base

5.39.2 lineaire

Description: not_set

See also: interpolation_champ_face_deriv (5.39)

Usage:

lineaire {

```
    [ vitesse_fluide_explicite ]
```

```
}
```

where

- **vitesse_fluide_explicite**

5.40 transport_k_epsilon

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

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

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

See also: eqn_base (5.23)

Usage:

transport_k_epsilon obj Lire obj {

```

[ with_nu str into ['yes', 'no']]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ initial_conditions|conditions_initiales condinits]
[ boundary_conditions|conditions_limites condlims]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
[ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
}
where

```

- **with_nu** *str* into ['yes', 'no']: yes/no
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (5.4) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.41 transport_marqueur_ft

Description: not_set

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

See also: eqn_base (5.23)

Usage:

transport_marqueur_ft obj Lire obj {

```

[ initial_conditions|conditions_initiales bloc_lecture]
[ injection injection_marqueur]
[ transformation_bulles bloc_lecture]
[ phase_marquee int]
[ methode_transport str into ['vitesse_interpolee', 'vitesse_particules']]
[ methode_couplage str into ['suivi', 'one_way_coupling', 'two_way_coupling']]
[ nb_iterations int]
[ contribution_one_way int into [0, 1]]
[ implicite int into [0, 1]]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
[ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
}

```

where

- **initial_conditions|conditions_initiales** *bloc_lecture* (3.6): ne semble pas standard
- **injection** *injection_marqueur* (5.42): The keyword injection can be used to inject periodically during the calculation some other particles. The syntax for ensemble_points and proprietes_particles is the same than the initial conditions for the particles. The keyword t_debut_injection give the injection initial time (by default, given by t_debut_integration) and dt_injection gives the injection time period (by default given by dt_min).
- **transformation_bulles** *bloc_lecture* (3.6): This keyword will activate the transformation of an inclusion (small bubbles) into a particle. localisation gives the sub-zones (N number of sub-zones and their names) where the transformation may happen. The diameter size for the inclusion transformation is given by either diameter_min option, in this case the inclusion will be suppressed for a diameter less than diameter_size, either by the beta_transfo option, in this case the inclusion will be suppressed for a diameter less than diameter_size*cell_volume (cell_volume is the volume of the cell containing the inclusion). interface specifies the name of the inclusion interface and t_debut_transfo is the beginning time for the inclusion transformation operation (by default, it is t_debut_integr value) and dt_transfo is the period transformation (by default, it is dt_min value). In a two phase flow calculation, the particles will be suppressed when entering into the non marked phase
- **phase_marquee** *int*: Phase number giving the marked phase, where the particles are located (when they leave this phase, they are suppressed). By default, for a the two phase fluide, the particles are supposed to be into the phase 0 (liquid).
- **methode_transport** *str* into ['vitesse_interpolee', 'vitesse_particules']: Kind of transport method for the particles. With vitesse_interpolee, the velocity of the particles is the velocity a fluid interpolation velocity (option by default). With vitesse_particules, the velocity of the particles is governed by the resolution of a momentum equation for the particles.
- **methode_couplage** *str* into ['suivi', 'one_way_coupling', 'two_way_coupling']: Way of coupling between the fluid and the particles. By default, (keyword suivi), there is no interaction between both. With one_way_coupling keyword, the fluid act on the particles. With two_way_coupling keyword, besides, particles act on the fluid.
- **nb_iterations** *int*: Number of sub-timesteps to solve the momentum equation for the particles (1 per default).
- **contribution_one_way** *int* into [0, 1]: Activate (1, default) or not (0) the fluid forces on the particles when one_way_coupling or two_way_coupling coupling method is used.
- **implicite** *int* into [0, 1]: Impliciting (1) or not (0) the time scheme when weight added source term is used in the momentum equation
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.

- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (4.10.1) 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 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 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 equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

5.42 injection_marqueur

Description: not_set

See also: objet_lecture (35)

Usage:

```
{
    ensemble_points bloc_lecture
    proprietes_particules bloc_lecture
    [ t_debut_injection float]
    [ dt_injection float]
```

```
}
```

where

- **ensemble_points** *bloc_lecture* (3.6)
- **proprietes_particules** *bloc_lecture* (3.6)
- **t_debut_injection** *float*
- **dt_injection** *float*

6 algo_base

Description: Basic class for multi-grid algorithms.

See also: [objet_u \(36\)](#) [algo_couple_1 \(6.1\)](#)

Usage:

6.1 algo_couple_1

Description: not_set

See also: [algo_base \(6\)](#)

Usage:

```
algo_couple_1 obj Lire obj {  
    [ dt_uniforme ]
```

```
}
```

where

- **dt_uniforme**

7 /*

7.1 /*

Description: bloc of Comment in a data file.

See also: [objet_u \(36\)](#)

Usage:

```
/* comm
```

where

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

8 champ_generique_base

Description: not_set

See also: [objet_u \(36\)](#) [champ_post_de_champs_post \(8.1\)](#) [predefini \(8.15\)](#) [champ_post_refchamp \(8.17\)](#)

Usage:

8.1 champ_post_de_champs_post

Description: not_set

See also: [champ_generique_base \(8\)](#) [champ_post_operateur_eqn \(8.5\)](#) [champ_post_transformation \(8.19\)](#) [champ_post_operateur_base \(8.4\)](#) [champ_post_statistiques_base \(8.6\)](#) [champ_post_extraction \(8.10\)](#) [champ_post_morceau_equation \(8.13\)](#) [champ_post_tparoi_vef \(8.18\)](#) [champ_post_interpolation \(8.12\)](#) [champ_post_reduction_0d \(8.16\)](#)

Usage:

```
champ_post_de_champs_post obj Lire obj {
```

```

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

```

- **source** *champ_generique_base* (8): the source field.
- **nom_source** *str*: To name a source field with the `nom_source` keyword
- **source_reference** *str*
- **sources_reference** *list_nom_virgule* (8.2)
- **sources** *listchamp_generique* (8.3): sources { Champ_Post.... { ... } Champ_Post.. { ... } }

8.2 list_nom_virgule

Description: List of name.

See also: `listobj` (34.3)

Usage:
{ object1 , object2 }
list of *nom_anonyme* (25.1) separated with ,

8.3 listchamp_generique

Description: XXX

See also: `listobj` (34.3)

Usage:
{ object1 , object2 }
list of *champ_generique_base* (8) separated with ,

8.4 champ_post_operateur_base

Description: `not_set`

See also: `champ_post_de_champs_post` (8.1) `champ_post_operateur_gradient` (8.11) `champ_post_operateur-divergence` (8.8)

Usage:
champ_post_operateur_base obj Lire obj {

```

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

```

- **source** *champ_generique_base* (8) for inheritance: the source field.

- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

8.5 champ_post_operateur_eqn

Synonymous: **operateur_eqn**

Description: `not_set`

See also: `champ_post_de_champs_post` (8.1)

Usage:

```
champ_post_operateur_eqn obj Lire obj {
    [ numero_op int ]
    [ numero_source int ]
    [ sans_solveur_masse ]
    [ source champ_generique_base ]
    [ nom_source str ]
    [ source_reference str ]
    [ sources_reference list_nom_virgule ]
    [ sources listchamp_generique ]
}
```

where

- **numero_op** *int*
- **numero_source** *int*
- **sans_solveur_masse**
- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

8.6 champ_post_statistiques_base

Description: `not_set`

See also: `champ_post_de_champs_post` (8.1) `correlation` (8.7) `moyenne` (8.14) `ecart_type` (8.9)

Usage:

```
champ_post_statistiques_base obj Lire obj {
    t_deb float
    t_fin float
    [ source champ_generique_base ]
    [ nom_source str ]
    [ source_reference str ]
    [ sources_reference list_nom_virgule ]
}
```

```
[ sources listchamp_generique]
```

```
}
```

where

- **t_deb** *float*: Start of integration time
- **t_fin** *float*: End of integration time
- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

8.7 correlation

Synonymous: **champ_post_statistiques_correlation**

Description: to calculate the correlation between the two fields.

See also: `champ_post_statistiques_base` (8.6)

Usage:

correlation obj Lire obj {

```
    t_deb float
```

```
    t_fin float
```

```
    [ source champ_generique_base]
```

```
    [ nom_source str]
```

```
    [ source_reference str]
```

```
    [ sources_reference list_nom_virgule]
```

```
    [ sources listchamp_generique]
```

```
}
```

where

- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

8.8 champ_post_operateur_divergence

Synonymous: **divergence**

Description: To calculate divergency of a given field.

See also: `champ_post_operateur_base` (8.4)

Usage:

champ_post_operateur_divergence obj Lire obj {

```

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

```

- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

8.9 ecart_type

Synonymous: **champ_post_statistiques_ecart_type**

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

See also: `champ_post_statistiques_base` (8.6)

Usage:

```

ecart_type obj Lire obj {
    t_deb float
    t_fin float
    [ source champ_generique_base]
    [ nom_source str]
    [ source_reference str]
    [ sources_reference list_nom_virgule]
    [ sources listchamp_generique]
}
where

```

- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

8.10 champ_post_extraction

Synonymous: **extraction**

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

See also: `champ_post_de_champs_post` (8.1)

Usage:

```
champ_post_extraction obj Lire obj {  
    domaine str  
    nom_frontiere str  
    [ methode str into ['trace', 'champ_frontiere']]  
    [ source champ_generique_base]  
    [ nom_source str]  
    [ source_reference str]  
    [ sources_reference list_nom_virgule]  
    [ sources listchamp_generique]  
}
```

where

- **domaine** *str*: name of the volume field
- **nom_frontiere** *str*: boundary name where the values of the volume field will be picked
- **methode** *str* into ['trace', 'champ_frontiere']: name of the extraction method (trace by_default or champ_frontiere)
- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

8.11 champ_post_operateur_gradient

Synonymous: **gradient**

Description: To calculate gradient of a given field.

See also: champ_post_operateur_base (8.4)

Usage:

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

where

- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

8.12 champ_post_interpolation

Synonymous: **interpolation**

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

See also: champ_post_de_champs_post (8.1)

Usage:

champ_post_interpolation obj Lire obj {

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

}

where

- **localisation** *str*: type_loc indicate where is done the interpolation (elem for element or som for node).
- **methode** *str*: The optional keyword methode is limited to calculer_champ_post for the moment.
- **domaine** *str*: the domain name where the interpolation is done (by default, the calculation domain)
- **optimisation_sous_maillage** *str into* ['default', 'yes', 'no']
- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... }}

8.13 champ_post_morceau_equation

Synonymous: **morceau_equation**

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

See also: champ_post_de_champs_post (8.1)

Usage:

champ_post_morceau_equation obj Lire obj {

```
    type  str
    numero  int
    option str into ['stabilite', 'flux_bords', 'flux_surfacique_bords']
    [ compo  int]
    [ source champ_generique_base]
    [ nom_source str]
```

```

[ source_reference str]
[ sources_reference list_nom_virgule]
[ sources listchamp_generique]
}
where

```

- **type** *str*: can only be `operateur` for equation operators.
- **numero** *int*: numero will be 0 (diffusive operator) or 1 (convective operator).
- **option** *str* into [`'stabilite'`, `'flux_bords'`, `'flux_surfacique_bords'`]: option is stability for time steps or `flux_bords` for boundary fluxes or `flux_surfacique_bords` for boundary surfacic fluxes
- **compo** *int*: compo will specify the number component of the boundary flux (for boundary fluxes, in this case compo permits to specify the number component of the boundary flux choosen).
- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: sources { `Champ_Post.... { ... } Champ_Post.. { ... }` }

8.14 moyenne

Synonymous: **champ_post_statistiques_moyenne**

Description: to calculate the average of the field over time

See also: `champ_post_statistiques_base` (8.6)

Usage:

```

moyenne obj Lire obj {
    [ moyenne_convergee champ_base]
    t_deb float
    t_fin float
    [ source champ_generique_base]
    [ nom_source str]
    [ source_reference str]
    [ sources_reference list_nom_virgule]
    [ sources listchamp_generique]
}
where

```

- **moyenne_convergee** *champ_base* (16.1): This option allows to read a converged time averaged field in a .xyz file in order to calculate, when resuming the calculation, the statistics fields (rms, correlation) which depend on this average. In that case, the time averaged field is not updated during the resume of calculation. In this case, the time averaged field must be fully converged to avoid errors when calculating high order statistics.
- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: sources { `Champ_Post.... { ... } Champ_Post.. { ... }` }

8.15 predefini

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

See also: champ_generique_base (8)

Usage:

predefini obj Lire obj {

pb_champ *deuxmots*

}

where

- **pb_champ** *deuxmots* (5.27): { Pb_champ nom_pb nom_champ } : nom_pb is the problem name and nom_champ is the selected field name.

8.16 champ_post_reduction_0d

Synonymous: **reduction_0d**

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

See also: champ_post_de_champs_post (8.1)

Usage:

champ_post_reduction_0d obj Lire obj {

methode *str* into ['min', 'max', 'moyenne', 'average', 'moyenne_ponderee', 'weighted_average', 'somme', 'sum', 'somme_ponderee', 'weighted_sum', 'somme_ponderee_porosite', 'weighted_sum_porosity', 'euclidian_norm', 'normalized_euclidian_norm', 'L1_norm', 'L2_norm', 'valeur_a_gauche', 'left_value']

 [**source** *champ_generique_base*]

 [**nom_source** *str*]

 [**source_reference** *str*]

 [**sources_reference** *list_nom_virgule*]

 [**sources** *listchamp_generique*]

}

where

- **methode** *str* into ['min', 'max', 'moyenne', 'average', 'moyenne_ponderee', 'weighted_average', 'somme', 'sum', 'somme_ponderee', 'weighted_sum', 'somme_ponderee_porosite', 'weighted_sum_porosity', 'euclidian_norm', 'normalized_euclidian_norm', 'L1_norm', 'L2_norm', 'valeur_a_gauche', 'left_value']: name of the reduction method:
 - min for the minimum value,
 - max for the maximum value,
 - average (or moyenne) for a mean,
 - weighted_average (or moyenne_ponderee) for a mean ponderated by integration volumes, e.g: cell volumes for temperature and pressure in VDF, volumes around faces for velocity and temperature in VEF,
 - sum (or somme) for the sum of all the values of the field,

- `weighted_sum` (or `somme_ponderee`) for a weighted sum (integral),
- `weighted_average_porosity` (or `moyenne_ponderee_porosite`) and `weighted_sum_porosity` (or `somme_ponderee_porosite`) for the mean and sum weighted by the volumes of the elements, only for ELEM localisation,
- `euclidian_norm` for the euclidian norm,
- `normalized_euclidian_norm` for the euclidian norm normalized,
- `L1_norm` for norm L1,
- `L2_norm` for norm L2
- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

8.17 champ_post_refchamp

Synonymous: **refchamp**

Description: Field of prolem

See also: `champ_generique_base` (8)

Usage:

champ_post_refchamp obj Lire obj {

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

}

where

- **pb_champ** *deuxmots* (5.27): { `Pb_champ` `nom_pb` `nom_champ` } : `nom_pb` is the problem name and `nom_champ` is the selected field name.
- **nom_source** *str*: The alias name for the field

8.18 champ_post_tparoi_vef

Synonymous: **tparoi_vef**

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

See also: `champ_post_de_champs_post` (8.1)

Usage:

champ_post_tparoi_vef obj Lire obj {

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

```
[ sources listchamp_generique]
```

```
}
```

where

- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the *nom_source* keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

8.19 champ_post_transformation

Synonymous: **transformation**

Description: To create a field with a transformation.

See also: *champ_post_de_champs_post* (8.1)

Usage:

```
champ_post_transformation obj Lire obj {
```

```
    methode str into ['produit_scalaire', 'norme', 'vecteur', 'formule', 'composante']
```

```
    [ expression n word1 word2 ... wordn]
```

```
    [ numero int]
```

```
    [ localisation str]
```

```
    [ source champ_generique_base]
```

```
    [ nom_source str]
```

```
    [ source_reference str]
```

```
    [ sources_reference list_nom_virgule]
```

```
    [ sources listchamp_generique]
```

```
}
```

where

- **methode** *str* into [*'produit_scalaire'*, *'norme'*, *'vecteur'*, *'formule'*, *'composante'*]: methode *norme* : will calculate the norm of a vector given by a source field
methode *produit_scalaire* : will calculate the dot product of two vectors given by two sources fields
methode *composante* *numero* integer : will create a field by extracting the integer component of a field given by a source field
methode *formule* *expression* 1 : will create a scalar field located to elements using expressions with x,y,z,t parameters and field names given by a source field or several sources fields.
methode *vecteur* *expression* N *f1*(x,y,z,t) *fN*(x,y,z,t) : will create a vector field located to elements by defining its N components with N expressions with x,y,z,t parameters and field names given by a source field or several sources fields.
- **expression** *n word1 word2 ... wordn*: see methodes *formule* and *vecteur*
- **numero** *int*: see methode *composante*
- **localisation** *str*: *type_loc* indicate where is done the interpolation (elem for element or som for node). The optional keyword *methode* is limited to *calculer_champ_post* for the moment
- **source** *champ_generique_base* (8) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the *nom_source* keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (8.2) for inheritance
- **sources** *listchamp_generique* (8.3) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

9 chimie

Description: Keyword to describe the chemical reactions

See also: `objet_u` ([36](#))

Usage:

chimie obj Lire obj {

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

}

where

- **reactions** *reactions* ([9.1](#)): list of reactions
- **modele_micro_melange** *int*: `modele_micro_melange` (0 by default)
- **constante_modele_micro_melange** *float*: `constante` of `modele` (1 by default)
- **espece_en_competition_micro_melange** *str*: `espece` in competition in reactions

9.1 reactions

Description: list of reactions

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

Usage:

{ object1 , object2 }

list of *reaction* ([9.1.1](#)) separated with ,

9.1.1 reaction

Description: Keyword to describe reaction:

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

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` ([35](#))

Usage:

{

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

}
where

- **reactifs** *str*: LHS of equation (ex CH4+2*O2)
- **produits** *str*: RHS of equation (ex CO2+2*H2O)
- **constante_taux_reaction** *float*: constante of cinetic K
- **coefficients_activites** *bloc_lecture* (3.6): coefficients of activity (exemple { CH4 1 O2 2 })
- **enthalpie_reaction** *float*: DH
- **energie_activation** *float*: Ea
- **exposant_beta** *float*: Beta
- **contre_reaction** *float*: K_inv
- **contre_energie_activation** *float*: c_r_Ea

10 class_generic

Description: not_set

See also: objet_u (36) dt_start (10.9) solveur_sys_base (10.16) Modele_Fonc_Realisable_base (10.2)

Usage:

10.1 Modele_Fonc_Realisable

Description: Deriv for instantiation of functions necessary to Realizable K-Epsilon Turbulence Model

See also: Modele_Fonc_Realisable_base (10.2)

Usage:

10.2 Modele_Fonc_Realisable_base

Description: Base class for Functions necessary to Realizable K-Epsilon Turbulence Model

See also: class_generic (10) Modele_Fonc_Realisable (10.1) Shih_Zhu_Lumley (10.4) Modele_Shih_Zhu-Lumley_VDF (10.3)

Usage:

10.3 Modele_Shih_Zhu_Lumley_VDF

Description: Functions necessary to Realizable K-Epsilon Turbulence Model in VDF

See also: Modele_Fonc_Realisable_base (10.2)

Usage:

Modele_Shih_Zhu_Lumley_VDF obj Lire obj {

[**a0** *float*]

}
where

- **a0** *float*: value of parameter A0 in U* formula

10.4 Shih_Zhu_Lumley

Description: Functions necessary to Realizable K-Epsilon Turbulence Model in VEF

See also: `Modele_Fonc_Realisable_base` ([10.2](#))

Usage:

```
Shih_Zhu_Lumley obj Lire obj {
```

```
    [ a0 float ]
```

```
}
```

where

- **a0** *float*: value of parameter A0 in U* formula

10.5 cholesky

Description: Cholesky direct method.

See also: `solveur_sys_base` ([10.16](#))

Usage:

```
cholesky obj Lire obj {
```

```
    [ impr ]
```

```
    [ quiet ]
```

```
}
```

where

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

10.6 dt_calc

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

See also: `dt_start` ([10.9](#))

Usage:

```
dt_calc
```

10.7 dt_fixe

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

See also: `dt_start` ([10.9](#))

Usage:

```
dt_fixe value
```

where

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

10.8 dt_min

Description: The first iteration is based on dt_min.

See also: dt_start (10.9)

Usage:

dt_min

10.9 dt_start

Description: not_set

See also: class_generic (10) dt_calc (10.6) dt_min (10.8) dt_fixe (10.7)

Usage:

dt_start

10.10 gcp_ns

Description: not_set

See also: gcp (10.15)

Usage:

```
gcp_ns obj Lire obj {  
    solveur0 solveur_sys_base  
    solveur1 solveur_sys_base  
    [ precond precond_base ]  
    [ precond_nul ]  
    seuil float  
    [ impr ]  
    [ quiet ]  
    [ save_matrix | save_matrice ]  
    [ optimized ]  
    [ nb_it_max int ]  
}
```

where

- **solveur0** *solveur_sys_base* (10.16): Solver type.
- **solveur1** *solveur_sys_base* (10.16): Solver type.
- **precond** *precond_base* (27) for inheritance: Keyword to define system preconditioning in order to accelerate resolution by the conjugated gradient. Many parallel preconditioning methods are not equivalent to their sequential counterpart, and you should therefore expect differences, especially when you select a high value of the final residue (seuil). The result depends on the number of processors and on the mesh splitting. It is sometimes useful to run the solver with no preconditioning at all. In particular:
 - when the solver does not converge during initial projection,
 - when comparing sequential and parallel computations.With no preconditioning, except in some particular cases (no open boundary), the sequential and the parallel computations should provide exactly the same results within fpu accuracy. If not, there might be a coding error or the system of equations is singular.
- **precond_nul** for inheritance: Keyword to not use a preconditioning method.

- **seuil** *float* for inheritance: Value of the final residue. The gradient ceases iteration when the Euclidean residue standard $\|Ax-B\|$ is less than this value.
- **impr** for inheritance: Keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **quiet** for inheritance: To not displaying any outputs of the solver.
- **save_matrix|save_matrice** for inheritance: to save the matrix in a file.
- **optimized** for inheritance: This keyword triggers a memory and network optimized algorithms useful for strong scaling (when computing less than 100 000 elements per processor). The matrix and the vectors are duplicated, common items removed and only virtual items really used in the matrix are exchanged.
Warning: this is experimental and known to fail in some VEF computations (L2 projection step will not converge). Works well in VDF.
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gcp.

10.11 gen

Description: not_set

See also: solveur_sys_base ([10.16](#))

Usage:

```
gen obj Lire obj {
    solv_elem str
    precondition precondition_base
    [seuil float]
    [impr ]
    [save_matrix|save_matrice ]
    [quiet ]
    [nb_it_max int]
    [force ]
}
```

where

- **solv_elem** *str*: To specify a solver among gmres or bicgstab.
- **precondition** *precondition_base* ([27](#)): The only preconditionner that we can specify is ilu.
- **seuil** *float*: Value of the final residue. The solver ceases iterations when the Euclidean residue standard $\|Ax-B\|$ is less than this value. default value $1e-12$.
- **impr** : Keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **save_matrix|save_matrice** : To save the matrix in a file.
- **quiet** : To not displaying any outputs of the solver.
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the GEN solver.
- **force** : Keyword to set `ipar[5]=-1` in the GEN solver. This is helpful if you notice that the solver does not perform more than 100 iterations. If this keyword is specified in the datafile, you should provide `nb_it_max`.

10.12 gmres

Description: Gmres method (for non symmetric matrix).

See also: solveur_sys_base ([10.16](#))

Usage:

```
gmres obj Lire obj {  
    [ impr ]  
    [ quiet ]  
    [ seuil float]  
    [ diag ]  
    [ nb_it_max int]  
    [ controle_residu int into [0, 1]]  
    [ save_matrix|save_matrice ]  
    [ dim_espace_krilov int]  
}
```

where

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

10.13 optimal

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

See also: solveur_sys_base ([10.16](#))

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

10.14 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

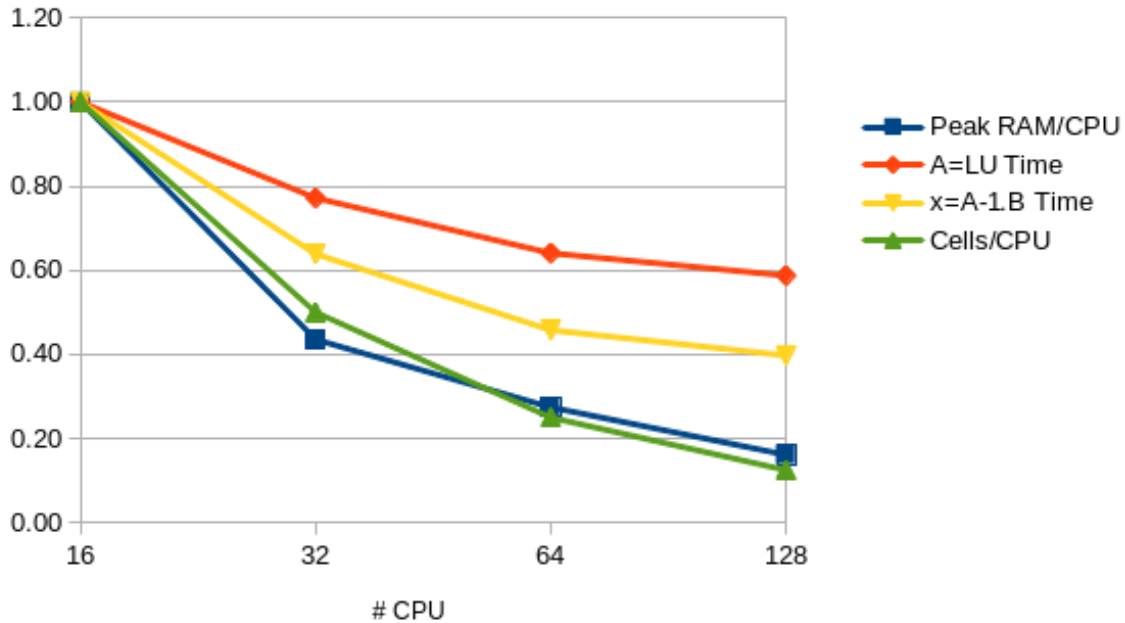
IBICGSTAB : Improved version of previous one for massive parallel computations (only a single global reduction operation instead of the usual 3 or 4).

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 only be used during a parallel calculation, whereas the two first are available for sequential or parallel calculations. It seems that the CPU cost of A=LU factorization but also of the backward/forward elimination steps may sometimes be reduced by selecting a different ordering (Scotch seems often the best for b/f elimination) than the default one. Notice that this solver requires a huge amount of memory compared to iterative methods. To know how many RAM you will need by core, then use the **impr** option to have detailed informations during the analysis phase and before the factorisation phase (in the following output, you will learn that the largest memory is taken by the 0th CPU with 108MB):

```
...
** Rank of proc needing largest memory in IC facto      :      0
** Estimated corresponding MBYTES for IC facto         :    108
...
```

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

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



CHOLESKY_OUT_OF_CORE : Same as the previous one but with a written LU decomposition of disk (save RAM memory but add an extra CPU cost during $Ax=B$ solve)

CHOLESKY_SUPERLU : Parallelized Cholesky from SUPERLU_DIST library (less CPU and RAM efficient than the previous one)

CHOLESKY_PASTIX : Parallelized Cholesky from PASTIX library

CHOLESKY_UMFPACK : Sequential Cholesky from UMFPACK library (seems fast).

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

trust datafile [N] -ksp_view -help

...

Preconditioner (PC) Options -----

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

HYPRE preconditioner options

-pc_hypre_type <pilut> (choose one of) pilut parasails boomeramg

HYPRE ParaSails Options

-pc_hypre_parasails_nlevels <1>: Number of number of levels (None)

-pc_hypre_parasails_thresh <0.1>: Threshold (None)

-pc_hypre_parasails_filter <0.1>: filter (None)

-pc_hypre_parasails_loadbal <0>: Load balance (None)

-pc_hypre_parasails_logging: <FALSE> Print info to screen (None)

-pc_hypre_parasails_reuse: <FALSE> Reuse nonzero pattern in preconditioner (None)
 -pc_hypre_parasails_sym <nonsymmetric> (choose one of) nonsymmetric SPD nonsymmetric, SPD

Krylov Method (KSP) Options -----

-ksp_type Krylov method:(one of) cg cgne stcg gltr richardson chebychev gmres tcqmr
 bcgs bcgsl cgs tfqmr cr lsqr preonly qcg bicg fgmres minres symmlq lgmres lcd (KSPSetType)
 -ksp_max_it <10000>: Maximum number of iterations (KSPSetTolerances)
 -ksp_rtol <0>: Relative decrease in residual norm (KSPSetTolerances)
 -ksp_atol <1e-12>: Absolute value of residual norm (KSPSetTolerances)
 -ksp_divtol <10000>: Residual norm increase cause divergence (KSPSetTolerances)
 -ksp_converged_use_initial_residual_norm: Use initial residual residual norm for computing relative convergence
 -ksp_monitor_singular_value <stdout>: Monitor singular values (KSPMonitorSet)
 -ksp_monitor_short <stdout>: Monitor preconditioned residual norm with fewer digits (KSPMonitorSet)
 -ksp_monitor_draw: Monitor graphically preconditioned residual norm (KSPMonitorSet)
 -ksp_monitor_draw_true_residual: Monitor graphically true residual norm (KSPMonitorSet)

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

Solveur_pression Petsc CLI { -ksp_type richardson -pc_type hypre -pc_hypre_type boomeramg -ksp_atol 1.e-7 }

Precond : Several preconditioners are available :

NULL { } : No preconditioner used

BLOCK_JACOBI_ICC { **level** k **ordering** *natural* | **rcm** } : Incomplete Cholesky factorization for symmetric matrix with the PETSc implementation. The integer k is the factorization level (default value, 1). In parallel, the factorization is done by block (one per processor by default). The ordering of the local matrix is **natural** by default, but **rcm** ordering, which reduces the bandwidth of the local matrix, may interestingly improve the quality of the decomposition and 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 disk (where NBROWS is the number of rows of the matrix and NCPUS the number of CPUs used).
- III) Partition your VEF mesh with a **largeur_joint** value of 1
- IV) Run your parallel calculation completely now and substitute the **save_matrix** option by the **read_matrix** option. Some interesting gains have been noticed when the cost of linear system solve with PETSc is small compared to all the other operations.

TIPS:

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

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

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

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

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

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

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

See also: `solveur_sys_base` ([10.16](#))

Usage:

petsc solveur option_solveur

where

- **solveur** *str*
- **option_solveur** *bloc_lecture* ([3.6](#))

10.15 gcp

Description: Preconditioned conjugated gradient.

See also: `solveur_sys_base` ([10.16](#)) `gcp_ns` ([10.10](#))

Usage:

gcp obj Lire obj {

```

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

```

- **precond** *precond_base* (27): Keyword to define system preconditioning in order to accelerate resolution by the conjugated gradient. Many parallel preconditioning methods are not equivalent to their sequential counterpart, and you should therefore expect differences, especially when you select a high value of the final residue (**seuil**). The result depends on the number of processors and on the mesh splitting. It is sometimes useful to run the solver with no preconditioning at all. In particular:
 - when the solver does not converge during initial projection,
 - when comparing sequential and parallel computations.
 With no preconditioning, except in some particular cases (no open boundary), the sequential and the parallel computations should provide exactly the same results within fpu accuracy. If not, there might be a coding error or the system of equations is singular.
- **precond_nul** : Keyword to not use a preconditioning method.
- **seuil** *float*: Value of the final residue. The gradient ceases iteration when the Euclidean residue standard $\|Ax-B\|$ is less than this value.
- **impr** : Keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **quiet** : To not displaying any outputs of the solver.
- **save_matrix|save_matrice** : to save the matrix in a file.
- **optimized** : This keyword triggers a memory and network optimized algorithms useful for strong scaling (when computing less than 100 000 elements per processor). The matrix and the vectors are duplicated, common items removed and only virtual items really used in the matrix are exchanged. Warning: this is experimental and known to fail in some VEF computations (L2 projection step will not converge). Works well in VDF.
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the Gcp.

10.16 solveur_sys_base

Description: Basic class to solve the linear system.

See also: [class_generic](#) (10) [optimal](#) (10.13) [gen](#) (10.11) [petsc](#) (10.14) [gcp](#) (10.15) [cholesky](#) (10.5) [gmres](#) (10.12)

Usage:

11

11.1

Description: Comments in a data file.

See also: [objet_u](#) (36)

Usage:

comm

where

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

12 **condlim_base**

Description: Basic class of boundary conditions.

See also: [objet_u \(36\)](#) [paroi_fixe \(12.54\)](#) [symetrie \(12.71\)](#) [periodique \(12.67\)](#) [paroi_adiabatique \(12.36\)](#) [dirichlet \(12.6\)](#) [neumann \(12.35\)](#) [paroi_contact \(12.37\)](#) [paroi_contact_fictif \(12.38\)](#) [paroi_echange_contact_vdf \(12.45\)](#) [paroi_echange_externe_impose \(12.49\)](#) [paroi_echange_global_impose \(12.53\)](#) [Paroi \(12.3\)](#) [paroi_flux_impose \(12.56\)](#) [frontiere_ouverte_fraction_massique_imposee \(12.16\)](#) [paroi_echange_contact_correlation_vdf \(12.41\)](#) [paroi_echange_contact_correlation_vef \(12.42\)](#) [Neumann_homogene \(12.1\)](#) [paroi_ft_disc \(12.60\)](#) [sortie_libre_rho_variable \(12.69\)](#) [frontiere_ouverte_k_eps_impose \(12.21\)](#) [paroi_decalee_robin \(12.39\)](#) [flux_radiatif \(12.11\)](#) [contact_vdf_vef \(12.4\)](#) [contact_vef_vdf \(12.5\)](#) [echange_contact_vdf_ft_disc_solid \(12.9\)](#) [echange_contact_vdf_ft_disc \(12.8\)](#)

Usage:

condlim_base

12.1 **Neumann_homogene**

Description: Homogeneous neumann boundary condition

See also: [condlim_base \(12\)](#) [Neumann_paroi_adiabatique \(12.2\)](#)

Usage:

Neumann_homogene

12.2 **Neumann_paroi_adiabatique**

Description: Adiabatic wall neumann boundary condition

See also: [Neumann_homogene \(12.1\)](#)

Usage:

Neumann_paroi_adiabatique

12.3 **Paroi**

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

See also: [condlim_base \(12\)](#)

Usage:

Paroi

12.4 **contact_vdf_vef**

Description: Boundary condition in the case of two problems (VDF -> VEF).

See also: `condlim_base` ([12](#))

Usage:

contact_vdf_vdf champ

where

- **champ** *champ_front_base* ([17.1](#)): Boundary field type.

12.5 contact_vdf_vdf

Description: Boundary condition in the case of two problems (VEF -> VDF).

See also: `condlim_base` ([12](#))

Usage:

contact_vdf_vdf champ

where

- **champ** *champ_front_base* ([17.1](#)): Boundary field type.

12.6 dirichlet

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

See also: `condlim_base` ([12](#)) `paroi_defilante` ([12.40](#)) `paroi_knudsen_non_negligeable` ([12.62](#)) `frontiere_ouverte_vitesse_imposee` ([12.33](#)) `frontiere_ouverte_temperature_imposee` ([12.30](#)) `frontiere_ouverte_concentration_imposee` ([12.15](#)) `paroi_temperature_imposee` ([12.64](#)) `scalaire_impose_paro` ([12.68](#)) `paroi_rugueuse` ([12.63](#))

Usage:

dirichlet

12.7 echange_contact_rayo_transp_vdf

Description: Exchange boundary condition in VDF between the transparent fluid and the solid for a problem coupled with radiation. Without radiation, it is the equivalent of the `Paroi_Echange_contact_VDF` exchange condition.

See also: `paroi_echange_contact_vdf` ([12.45](#))

Usage:

echange_contact_rayo_transp_vdf autrepb nameb temp h

where

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

12.8 `echange_contact_vdf_ft_disc`

Description: `echange_conatct_vdf` en precisant la phase

See also: `condlim_base` ([12](#))

Usage:

`echange_contact_vdf_ft_disc` obj Lire obj {

```
    autre_probleme  str
    autre_bord      str
    autre_champ_temperature  str
    nom_mon_indicatrice  str
    phase           int
```

}

where

- **`autre_probleme`** *str*: name of other problem
- **`autre_bord`** *str*: name of other boundary
- **`autre_champ_temperature`** *str*: name of other field
- **`nom_mon_indicatrice`** *str*: name of indicatrice
- **`phase`** *int*: phase

12.9 `echange_contact_vdf_ft_disc_solid`

Description: `echange_conatct_vdf` en precisant la phase

See also: `condlim_base` ([12](#))

Usage:

`echange_contact_vdf_ft_disc_solid` obj Lire obj {

```
    autre_probleme  str
    autre_bord      str
    autre_champ_temperature_indic1  str
    autre_champ_temperature_indic0  str
    autre_champ_indicatrice  str
```

}

where

- **`autre_probleme`** *str*: name of other problem
- **`autre_bord`** *str*: name of other boundary
- **`autre_champ_temperature_indic1`** *str*: name of temperature indic 1
- **`autre_champ_temperature_indic0`** *str*: name of temperature indic 0
- **`autre_champ_indicatrice`** *str*: name of indicatrice

12.10 `entree_temperature_imposee_h`

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

See also: `frontiere_ouverte_temperature_imposee` ([12.30](#))

Usage:

`entree_temperature_imposee_h` ch

where

- **ch** *champ_front_base* (17.1): Boundary field type.

12.11 flux_radiatif

Description: Boundary condition for radiation equation.

See also: *condlim_base* (12) *flux_radiatif_vdf* (12.12) *flux_radiatif_vef* (12.13)

Usage:

flux_radiatif na a ne emissivite

where

- **na** *str into ['A']*: Keyword for constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **a** *float*: Value of constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **ne** *str into ['emissivite']*: Keyword for wall emissivity.
- **emissivite** *champ_front_base* (17.1): Wall emissivity, value between 0 and 1.

12.12 flux_radiatif_vdf

Description: Boundary condition for radiation equation in VDF.

See also: *flux_radiatif* (12.11)

Usage:

flux_radiatif_vdf na a ne emissivite

where

- **na** *str into ['A']*: Keyword for constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **a** *float*: Value of constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **ne** *str into ['emissivite']*: Keyword for wall emissivity.
- **emissivite** *champ_front_base* (17.1): Wall emissivity, value between 0 and 1.

12.13 flux_radiatif_vef

Description: Boundary condition for radiation equation in VEF.

See also: *flux_radiatif* (12.11)

Usage:

flux_radiatif_vef na a ne emissivite

where

- **na** *str into ['A']*: Keyword for constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **a** *float*: Value of constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain).
- **ne** *str into ['emissivite']*: Keyword for wall emissivity.
- **emissivite** *champ_front_base* (17.1): Wall emissivity, value between 0 and 1.

12.14 `frontiere_ouverte`

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

See also: `neumann` (12.35) `frontiere_ouverte_rayo_transp` (12.26) `frontiere_ouverte_rayo_semi_transp` (12.25)

Usage:

`frontiere_ouverte var_name ch`

where

- **`var_name`** *str* into ['T_ext', 'C_ext', 'K_Eps_ext', 'Fluctu_Temperature_ext', 'Flux_Chaleur_Turb_ext', 'V2_ext']: Field name.
- **`ch`** *champ_front_base* (17.1): Boundary field type.

12.15 `frontiere_ouverte_concentration_imposee`

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

See also: `dirichlet` (12.6)

Usage:

`frontiere_ouverte_concentration_imposee ch`

where

- **`ch`** *champ_front_base* (17.1): Boundary field type.

12.16 `frontiere_ouverte_fraction_massique_imposee`

Description: `not_set`

See also: `condlim_base` (12)

Usage:

`frontiere_ouverte_fraction_massique_imposee ch`

where

- **`ch`** *champ_front_base* (17.1): Boundary field type.

12.17 `frontiere_ouverte_gradient_pression_impose`

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

See also: `neumann` (12.35) `frontiere_ouverte_gradient_pression_impose_vefprep1b` (12.18)

Usage:

`frontiere_ouverte_gradient_pression_impose ch`

where

- **`ch`** *champ_front_base* (17.1): Boundary field type.

12.18 `frontiere_ouverte_gradient_pression_impose_vefprep1b`

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

See also: `frontiere_ouverte_gradient_pression_impose` ([12.17](#))

Usage:

`frontiere_ouverte_gradient_pression_impose_vefprep1b` **`ch`**

where

- **`ch`** *champ_front_base* ([17.1](#)): Boundary field type.

12.19 `frontiere_ouverte_gradient_pression_libre_vef`

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

See also: `neumann` ([12.35](#))

Usage:

`frontiere_ouverte_gradient_pression_libre_vef`

12.20 `frontiere_ouverte_gradient_pression_libre_vefprep1b`

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

See also: `neumann` ([12.35](#))

Usage:

`frontiere_ouverte_gradient_pression_libre_vefprep1b`

12.21 `frontiere_ouverte_k_eps_impose`

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

See also: `condlim_base` ([12](#))

Usage:

`frontiere_ouverte_k_eps_impose` **`ch`**

where

- **`ch`** *champ_front_base* ([17.1](#)): Boundary field type.

12.22 `frontiere_ouverte_pression_imposee`

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

See also: `neumann` ([12.35](#))

Usage:

frontiere_ouverte_pression_imposee ch

where

- **ch** *champ_front_base* (17.1): Boundary field type.

12.23 frontiere_ouverte_pression_imposee_orlansky

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

See also: `neumann` (12.35)

Usage:

frontiere_ouverte_pression_imposee_orlansky

12.24 frontiere_ouverte_pression_moyenne_imposee

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

See also: `neumann` (12.35)

Usage:

frontiere_ouverte_pression_moyenne_imposee pext

where

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

12.25 frontiere_ouverte_rayo_semi_transp

Description: Keyword to set a boundary outlet temperature condition on the boundary called bord (edge) (diffusion flux zero) for a radiation problem with semi transparent gas.

See also: `frontiere_ouverte` (12.14)

Usage:

frontiere_ouverte_rayo_semi_transp var_name ch

where

- **var_name** *str* into ['T_ext', 'C_ext', 'K_Eps_ext', 'Fluctu_Temperature_ext', 'Flux_Chaleur_Turb_ext', 'V2_ext']: Field name.
- **ch** *champ_front_base* (17.1): Boundary field type.

12.26 frontiere_ouverte_rayo_transp

Description: Keyword to set a boundary outlet temperature condition on the boundary called bord (edge) (diffusion flux zero) for a radiation problem with transparent gas.

See also: `frontiere_ouverte` (12.14) `frontiere_ouverte_rayo_transp_vdf` (12.27) `frontiere_ouverte_rayo_transp_vef` (12.28)

Usage:

frontiere_ouverte_rayo_transp var_name ch

where

- **var_name** *str* into ['T_ext', 'C_ext', 'K_Eps_ext', 'Fluctu_Temperature_ext', 'Flux_Chaleur_Turb_ext', 'V2_ext']: Field name.
- **ch** *champ_front_base* (17.1): Boundary field type.

12.27 frontiere_ouverte_rayo_transp_vdf

Description: doit disparaitre

See also: *frontiere_ouverte_rayo_transp* (12.26)

Usage:

frontiere_ouverte_rayo_transp_vdf var_name ch

where

- **var_name** *str* into ['T_ext', 'C_ext', 'K_Eps_ext', 'Fluctu_Temperature_ext', 'Flux_Chaleur_Turb_ext', 'V2_ext']: Field name.
- **ch** *champ_front_base* (17.1): Boundary field type.

12.28 frontiere_ouverte_rayo_transp_vdf

Description: doit disparaitre

See also: *frontiere_ouverte_rayo_transp* (12.26)

Usage:

frontiere_ouverte_rayo_transp_vdf var_name ch

where

- **var_name** *str* into ['T_ext', 'C_ext', 'K_Eps_ext', 'Fluctu_Temperature_ext', 'Flux_Chaleur_Turb_ext', 'V2_ext']: Field name.
- **ch** *champ_front_base* (17.1): Boundary field type.

12.29 frontiere_ouverte_rho_u_impose

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

See also: *frontiere_ouverte_vitesse_imposee_sortie* (12.34)

Usage:

frontiere_ouverte_rho_u_impose ch

where

- **ch** *champ_front_base* (17.1): Boundary field type.

12.30 `frontiere_ouverte_temperature_imposee`

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

See also: `dirichlet` ([12.6](#)) `entree_temperature_imposee_h` ([12.10](#)) `frontiere_ouverte_temperature_imposee_rayo_transp` ([12.32](#)) `frontiere_ouverte_temperature_imposee_rayo_semi_transp` ([12.31](#))

Usage:

`frontiere_ouverte_temperature_imposee ch`

where

- **`ch`** `champ_front_base` ([17.1](#)): Boundary field type.

12.31 `frontiere_ouverte_temperature_imposee_rayo_semi_transp`

Description: Imposed temperature condition for a radiation problem with semi transparent gas.

See also: `frontiere_ouverte_temperature_imposee` ([12.30](#))

Usage:

`frontiere_ouverte_temperature_imposee_rayo_semi_transp ch`

where

- **`ch`** `champ_front_base` ([17.1](#)): Boundary field type.

12.32 `frontiere_ouverte_temperature_imposee_rayo_transp`

Description: Imposed temperature condition for a radiation problem with transparent gas.

See also: `frontiere_ouverte_temperature_imposee` ([12.30](#))

Usage:

`frontiere_ouverte_temperature_imposee_rayo_transp ch`

where

- **`ch`** `champ_front_base` ([17.1](#)): Boundary field type.

12.33 `frontiere_ouverte_vitesse_imposee`

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

See also: `dirichlet` ([12.6](#)) `frontiere_ouverte_vitesse_imposee_sortie` ([12.34](#))

Usage:

`frontiere_ouverte_vitesse_imposee ch`

where

- **`ch`** `champ_front_base` ([17.1](#)): Boundary field type.

12.34 **frontiere_ouverte_vitesse_imposee_sortie**

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

See also: [frontiere_ouverte_vitesse_imposee \(12.33\)](#) [frontiere_ouverte_rho_u_impose \(12.29\)](#)

Usage:

frontiere_ouverte_vitesse_imposee_sortie ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.35 **neumann**

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

See also: [condlim_base \(12\)](#) [frontiere_ouverte_gradient_pression_libre_vef \(12.19\)](#) [frontiere_ouverte_gradient_pression_libre_vefprep1b \(12.20\)](#) [frontiere_ouverte_gradient_pression_impose \(12.17\)](#) [frontiere_ouverte_pression_imposee \(12.22\)](#) [frontiere_ouverte_pression_imposee_orlansky \(12.23\)](#) [frontiere_ouverte_pression_moyenne_imposee \(12.24\)](#) [frontiere_ouverte \(12.14\)](#) [sortie_libre_temperature_imposee_h \(12.70\)](#)

Usage:

neumann

12.36 **paroi_adiabatique**

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

See also: [condlim_base \(12\)](#)

Usage:

paroi_adiabatique

12.37 **paroi_contact**

Description: Thermal condition between two domains. Important: the name of the boundaries in the two domains should be the same. (Warning: there is also an old limitation not yet fixed on the sequential algorithm in VDF to detect the matching faces on the two boundaries: faces should be ordered in the same way). The kind of condition depends on the discretization. In VDF, it is a heat exchange condition, and in VEF, a temperature condition.

Such a coupling requires coincident meshes for the moment. In case of non-coincident meshes, run is stopped and two external files are automatically generated in VEF (*connectivity_failed_boundary_name* and *connectivity_failed_pb_name.med*). In 2D, the keyword *Decouper_bord_coincident* associated to the *connectivity_failed_boundary_name* file allows to generate a new coincident mesh.

In 3D, for a first preliminary cut domain with HOMARD (fluid for instance), the second problem associated to *pb_name* (solide in a fluid/solid coupling problem) has to be submitted to HOMARD cutting procedure with *connectivity_failed_pb_name.med*.

Such a procedure works as while the primary refined mesh (fluid in our example) impacts the fluid/solid interface with a compact shape as described below (values 2 or 4 indicates the number of division from primary faces obtained in fluid domain at the interface after HOMARD cutting):

2-2-2-2-2-2

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

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

2-2-2-2-2 2-2

OK

2-2 2-2-2

2-4-2 2-2

2-2 2-2

NOT OK

See also: `condlim_base` ([12](#))

Usage:

paroi_contact autrepb nameb

where

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

12.38 **paroi_contact_fictif**

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

See also: `condlim_base` ([12](#))

Usage:

paroi_contact_fictif autrepb nameb conduct_fictif ep_fictive

where

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

12.39 **paroi_decalee_robin**

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

See also: `condlim_base` ([12](#))

Usage:

paroi_decalee_robin obj Lire obj {

delta *float*

}

where

- **delta** *float*

12.40 paroi_defilante

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

See also: [dirichlet \(12.6\)](#)

Usage:

paroi_defilante **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.41 paroi_echange_contact_correlation_vdf

Description: Class to define a thermohydraulic 1D model which will apply to a boundary of 2D or 3D domain.

Warning : For parallel calculation, the only possible partition will be according the axis of the model with the keyword Tranche.

See also: [condlim_base \(12\)](#)

Usage:

paroi_echange_contact_correlation_vdf obj Lire obj {

```
    dir int
    tinf float
    tsup float
    lambda str
    rho str
    cp float
    dt_impr float
    mu str
    debit float
    dh float
    volume str
    nu str
    [ reprise_correlation ]
```

}

where

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

- **nu** *str*: Nusselt number which may be a function of the Reynolds number (Re) and the Prandtl number (Pr).
- **reprise_correlation** : Keyword in the case of a resuming calculation with this correlation.

12.42 paroi_echange_contact_correlation_vef

Description: Class to define a thermohydraulic 1D model which will apply to a boundary of 2D or 3D domain.

Warning : For parallel calculation, the only possible partition will be according the axis of the model with the keyword `Tranche_geom`.

See also: `condlim_base` ([12](#))

Usage:

```
paroi_echange_contact_correlation_vef obj Lire obj {
    dir int
    tinf float
    tsup float
    lambda str
    rho str
    cp float
    dt_impr float
    mu str
    debit float
    dh float
    n int
    surface str
    nu str
    xinf float
    xsup float
    [ emissivite_pour_rayonnement_entre_deux_plaques_quasi_infinies float ]
    [ reprise_correlation ]
}
```

where

- **dir** *int*: Direction (0 : axis X, 1 : axis Y, 2 : axis Z) of the 1D model.
- **tinf** *float*: Inlet fluid temperature of the 1D model (oC or K).
- **tsup** *float*: Outlet fluid temperature of the 1D model (oC or K).
- **lambda** *str*: Thermal conductivity of the fluid (W.m-1.K-1).
- **rho** *str*: Mass density of the fluid (kg.m-3) which may be a function of the temperature T.
- **cp** *float*: Calorific capacity value at a constant pressure of the fluid (J.kg-1.K-1).
- **dt_impr** *float*: Printing period in `name_of_data_file_time.dat` files of the 1D model results.
- **mu** *str*: Dynamic viscosity of the fluid (kg.m-1.s-1) which may be a function of the temperature T.
- **debit** *float*: Surface flow rate (kg.s-1.m-2) of the fluid into the channel.
- **dh** *float*: Hydraulic diameter may be a function $f(x)$ with x position along the 1D axis ($x_{inf} \leq x \leq x_{sup}$).
- **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 ($x_{inf} \leq x \leq x_{sup}$).
- **nu** *str*: Nusselt number which may be a function of the Reynolds number (Re) and the Prandtl number (Pr).
- **xinf** *float*: Position of the inlet of the 1D mesh on the axis direction.

- **xsup** *float*: Position of the outlet of the 1D mesh on the axis direction.
- **emissivite_pour_rayonnement_entre_deux_plaques_quasi_infinies** *float*: Coefficient of emissivity for radiation between two quasi infinite plates.
- **reprise_correlation** : Keyword in the case of a resuming calculation with this correlation.

12.43 paroi_echange_contact_odvm_vdf

Description: not_set

See also: paroi_echange_contact_vdf ([12.45](#))

Usage:

paroi_echange_contact_odvm_vdf autrepb nameb temp h

where

- **autrepb** *str*: Name of other problem.
- **nameb** *str*: Name of bord.
- **temp** *str*: Name of field.
- **h** *float*: Value assigned to a coefficient (expressed in W.K-1m-2) that characterises the contact between the two mediums. In order to model perfect contact, h must be taken to be infinite. This value must obviously be the same in both the two problems blocks.
The surface thermal flux exchanged between the two mediums is represented by :
$$f_i = h (T_1 - T_2)$$
 where $1/h = d_1/\lambda_{a1} + 1/val_h_contact + d_2/\lambda_{a2}$
where d_i : distance between the node where T_i and the wall is found.

12.44 paroi_echange_contact_rayo_semi_transp_vdf

Description: Exchange boundary condition in VDF between the semi transparent fluid and the solid for a problem coupled with radiation.

See also: paroi_echange_contact_vdf ([12.45](#))

Usage:

paroi_echange_contact_rayo_semi_transp_vdf autrepb nameb temp h

where

- **autrepb** *str*: Name of other problem.
- **nameb** *str*: Name of bord.
- **temp** *str*: Name of field.
- **h** *float*: Value assigned to a coefficient (expressed in W.K-1m-2) that characterises the contact between the two mediums. In order to model perfect contact, h must be taken to be infinite. This value must obviously be the same in both the two problems blocks.
The surface thermal flux exchanged between the two mediums is represented by :
$$f_i = h (T_1 - T_2)$$
 where $1/h = d_1/\lambda_{a1} + 1/val_h_contact + d_2/\lambda_{a2}$
where d_i : distance between the node where T_i and the wall is found.

12.45 paroi_echange_contact_vdf

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

See also: condlim_base ([12](#)) paroi_echange_contact_vdf_ft ([12.46](#)) paroi_echange_contact_odvm_vdf ([12.43](#))

`exchange_contact_rayo_transp_vdf` (12.7) `paroi_echange_contact_rayo_semi_transp_vdf` (12.44)

Usage:

paroi_echange_contact_vdf **autrepb** **nameb** **temp** **h**

where

- **autrepb** *str*: Name of other problem.
- **nameb** *str*: Name of bord.
- **temp** *str*: Name of field.
- **h** *float*: Value assigned to a coefficient (expressed in W.K-1m-2) that characterises the contact between the two mediums. In order to model perfect contact, h must be taken to be infinite. This value must obviously be the same in both the two problems blocks.
The surface thermal flux exchanged between the two mediums is represented by :
$$f_i = h (T_1 - T_2)$$
 where $1/h = d_1/\lambda_{a1} + 1/\text{val_h_contact} + d_2/\lambda_{a2}$
where di : distance between the node where T_i and the wall is found.

12.46 paroi_echange_contact_vdf_ft

Description: This boundary condition is used between a conduction problem and a thermohydraulic problem with two phases flow (Front-Tracking method) to modelize heat exchange.

See also: `paroi_echange_contact_vdf` (12.45)

Usage:

paroi_echange_contact_vdf_ft **autrepb** **nameb** **temp** **h**

where

- **autrepb** *str*: Name of other problem.
- **nameb** *str*: Name of bord.
- **temp** *str*: Name of field.
- **h** *float*: Value assigned to a coefficient (expressed in W.K-1m-2) that characterises the contact between the two mediums. In order to model perfect contact, h must be taken to be infinite. This value must obviously be the same in both the two problems blocks.
The surface thermal flux exchanged between the two mediums is represented by :
$$f_i = h (T_1 - T_2)$$
 where $1/h = d_1/\lambda_{a1} + 1/\text{val_h_contact} + d_2/\lambda_{a2}$
where di : distance between the node where T_i and the wall is found.

12.47 paroi_echange_contact_vdf_zoom_fin

Description: External type exchange condition with a heat exchange coefficient and an imposed external temperature in the case of zoom (fine).

See also: `paroi_echange_externe_impose` (12.49)

Usage:

paroi_echange_contact_vdf_zoom_fin **h_imp** **himpc** **text** **ch**

where

- **h_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ_front_base* (17.1): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* (17.1): Boundary field type.

12.48 paroi_echange_contact_vdf_zoom_grossier

Description: External type exchange condition with a heat exchange coefficient and an imposed external temperature in the case of zoom (coarse).

See also: `paroi_echange_externer_impose` ([12.49](#))

Usage:

paroi_echange_contact_vdf_zoom_grossier h_imp himpc text ch
where

- **h_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ_front_base* ([17.1](#)): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.49 paroi_echange_externer_impose

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

See also: `condlim_base` ([12](#)) `paroi_echange_externer_impose_h` ([12.50](#)) `paroi_echange_externer_impose_rayo_transp` ([12.52](#)) `paroi_echange_externer_impose_rayo_semi_transp` ([12.51](#)) `paroi_echange_contact_vdf_zoom_grossier` ([12.48](#)) `paroi_echange_contact_vdf_zoom_fin` ([12.47](#))

Usage:

paroi_echange_externer_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* ([17.1](#)): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.50 paroi_echange_externer_impose_h

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

See also: `paroi_echange_externer_impose` ([12.49](#))

Usage:

paroi_echange_externer_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* ([17.1](#)): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.51 paroi_echange_externe_impose_rayo_semi_transp

Description: External type exchange condition for a coupled problem with radiation in semi transparent gas.

See also: `paroi_echange_externe_impose` ([12.49](#))

Usage:

paroi_echange_externe_impose_rayo_semi_transp h_imp himpc text ch
where

- **h_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ_front_base* ([17.1](#)): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.52 paroi_echange_externe_impose_rayo_transp

Description: External type exchange condition for a coupled problem with radiation in transparent gas.

See also: `paroi_echange_externe_impose` ([12.49](#))

Usage:

paroi_echange_externe_impose_rayo_transp h_imp himpc text ch
where

- **h_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ_front_base* ([17.1](#)): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.53 paroi_echange_global_impose

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

See also: `condlim_base` ([12](#))

Usage:

paroi_echange_global_impose h_imp himpc text ch
where

- **h_imp** *str*: Global exchange coefficient value. The global exchange coefficient value is expressed in W.m-2.K-1.
- **himpc** *champ_front_base* ([17.1](#)): Boundary field type.
- **text** *str*: External temperature value. The external temperature value is expressed in oC or K.
- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.54 paroi_fixe

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

See also: `condlim_base` ([12](#)) `paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets` ([12.55](#))

Usage:

paroi_fixe

12.55 `paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets`

Description: Boundary condition to obtain iso Geneppi2, without interest

See also: `paroi_fixe` ([12.54](#))

Usage:

paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets

12.56 `paroi_flux_impose`

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

See also: `condlim_base` ([12](#)) `paroi_flux_impose_rayo_transp` ([12.59](#)) `paroi_flux_impose_rayo_semi_transp_vdf` ([12.57](#)) `paroi_flux_impose_rayo_semi_transp_vef` ([12.58](#))

Usage:

paroi_flux_impose ch

where

- **ch** `champ_front_base` ([17.1](#)): Boundary field type.

12.57 `paroi_flux_impose_rayo_semi_transp_vdf`

Description: Normal flux condition at the wall called bord (edge) for a radiation problem in semi transparent gas (in VDF).

See also: `paroi_flux_impose` ([12.56](#))

Usage:

paroi_flux_impose_rayo_semi_transp_vdf ch

where

- **ch** `champ_front_base` ([17.1](#)): Boundary field type.

12.58 `paroi_flux_impose_rayo_semi_transp_vef`

Description: Normal flux condition at the wall called bord (edge) for a radiation problem in semi transparent gas (in VEF).

See also: `paroi_flux_impose` ([12.56](#))

Usage:

paroi_flux_impose_rayo_semi_transp_vef ch

where

- **ch** `champ_front_base` ([17.1](#)): Boundary field type.

12.59 paroi_flux_impose_rayo_transp

Description: Normal flux condition at the wall called bord (edge) for a radiation problem in transparent gas.

See also: paroi_flux_impose ([12.56](#))

Usage:

paroi_flux_impose_rayo_transp ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.60 paroi_ft_disc

Description: Boundary condition for Front-Tracking problem in the discontinuous version.

See also: condlim_base ([12](#))

Usage:

paroi_ft_disc type

where

- **type** *paroi_ft_disc_deriv* ([12.61](#)): Symetrie condition.

12.61 paroi_ft_disc_deriv

Description: not_set

See also: objet_lecture ([35](#)) symetrie ([12.61.1](#)) constant ([12.61.2](#))

Usage:

paroi_ft_disc_deriv

12.61.1 symetrie

Description: Symetrie condition in the case of two-phase flows

See also: paroi_ft_disc_deriv ([12.61](#))

Usage:

symetrie

12.61.2 constant

Description: condition contact angle fidex. The angle is measured between the wall and the interface in the phase 0.

See also: paroi_ft_disc_deriv ([12.61](#))

Usage:

constant ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.62 paroi_knudsen_non_negligeable

Description: Boundary condition for number of Knudsen (Kn) above 0.001 where slip-flow condition appears: the velocity near the wall depends on the shear stress : $Kn=l/L$ with l is the mean-free-path of the molecules and L a characteristic length scale.

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

Where k is a coefficient given by several laws:

Mawxell : $k=(2-s)*l/s$

Bestok&Karniadakis : $k=(2-s)/s*L*Kn/(1+Kn)$

Xue&Fan : $k=(2-s)/s*L*tanh(Kn)$

s is a value between 0 and 2 named accomodation coefficient. $s=1$ seems a good value.

Warning : The keyword is available for VDF calculation only for the moment.

See also: [dirichlet \(12.6\)](#)

Usage:

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

where

- **name_champ_1** *str into ['vitesse_paro', 'k']*: Field name.
- **champ_1** *champ_front_base (17.1)*: Boundary field type.
- **name_champ_2** *str into ['vitesse_paro', 'k']*: Field name.
- **champ_2** *champ_front_base (17.1)*: Boundary field type.

12.63 paroi_rugueuse

Description: Rough wall boundary

See also: [dirichlet \(12.6\)](#)

Usage:

paroi_rugueuse **obj** Lire **obj** {

erugu *float*

}

where

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

12.64 paroi_temperature_imposee

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

See also: [dirichlet \(12.6\)](#) [temperature_imposee_paro \(12.72\)](#) [paroi_temperature_imposee_rayo_transp \(12.66\)](#) [paroi_temperature_imposee_rayo_semi_transp \(12.65\)](#)

Usage:

paroi_temperature_imposee **ch**

where

- **ch** *champ_front_base (17.1)*: Boundary field type.

12.65 **paroi_temperature_imposee_rayo_semi_transp**

Description: Imposed temperature condition at the wall called bord (edge) for a radiation problem in semi transparent gas.

See also: `paroi_temperature_imposee` ([12.64](#))

Usage:

paroi_temperature_imposee_rayo_semi_transp **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.66 **paroi_temperature_imposee_rayo_transp**

Description: Imposed temperature condition at the wall called bord (edge) for a radiation problem in transparent gas.

See also: `paroi_temperature_imposee` ([12.64](#))

Usage:

paroi_temperature_imposee_rayo_transp **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.67 **periodique**

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

See also: `condlim_base` ([12](#))

Usage:

periodique

12.68 **scalaire_impose_paro**

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

See also: `dirichlet` ([12.6](#))

Usage:

scalaire_impose_paro **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.69 sortie_libre_rho_variable

Description: Class to define an outlet boundary condition at which the pressure is defined through the given field, whereas the density of the two-phase flow may varies (value of P/ρ given in $\text{Pa}/\text{kg}\cdot\text{m}^{-3}$).

See also: [condlim_base \(12\)](#)

Usage:

sortie_libre_rho_variable **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.70 sortie_libre_temperature_imposee_h

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

See also: [neumann \(12.35\)](#)

Usage:

sortie_libre_temperature_imposee_h **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

12.71 symetrie

Description: 1). For Navier-Stokes equations, this keyword is used to designate a symmetry condition concerning the velocity at the boundary called bord (edge) (normal velocity at the edge equal to zero and tangential velocity gradient at the edge equal to zero); 2). For scalar transport equation, this keyword is used to set a symmetry condition on scalar on the boundary named bord (edge).

See also: [condlim_base \(12\)](#)

Usage:

symetrie

12.72 temperature_imposee_paro

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

See also: [paroi_temperature_imposee \(12.64\)](#)

Usage:

temperature_imposee_paro **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13 discretisation_base

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

See also: objet_u ([36](#)) vdf ([13.3](#)) vef ([13.4](#)) polymac ([13.2](#)) ef ([13.1](#))

Usage:

13.1 ef

Description: Element Finite discretization.

See also: discretisation_base ([13](#))

Usage:

13.2 polymac

Description: polymac discretization.

See also: discretisation_base ([13](#))

Usage:

13.3 vdf

Description: Finite difference volume discretization.

See also: discretisation_base ([13](#))

Usage:

13.4 vef

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

Warning: it becomes an obsolete discretization.

See also: discretisation_base ([13](#)) vefprep1b ([13.5](#))

Usage:

13.5 vefprep1b

Description: Finite element volume discretization (P1NC/P1-bubble element). Since the 1.5.5 version, several new discretizations are available thanks to the optional keyword Read. By default, the VEFPreP1B keyword is equivalent to the former VEFPreP1B formulation (v1.5.4 and sooner). P0P1 (if used with the strong formulation for imposed pressure boundary) is equivalent to VEFPreP1B but the convergence is slower. VEFPreP1B dis is equivalent to VEFPreP1B dis Read dis { P0 P1 Changement_de_base_P1Bulle 1 Cl_pression_sommet_faible 0 }

See also: vef ([13.4](#))

Usage:

vefprep1b obj Lire obj {

```

[ changement_de_base_p1bulle int]
[ p0 ]
[ p1 ]
[ pa ]
[ rt ]
[ modif_div_face_dirichlet int]
[ cl_pression_sommet_faible int]
}
where

```

- **changement_de_base_p1bulle** *int*: (into=[0,1]) **changement_de_base_p1bulle** 1 This option may be used to have the P1NC/POP1 formulation (value set to 0) or the P1NC/P1Bulle formulation (value set to 1, the default).
- **p0** : Pressure nodes are added on element centres
- **p1** : Pressure nodes are added on vertices
- **pa** : Only available in 3D, pressure nodes are added on bones
- **rt** : For P1NCP1B
- **modif_div_face_dirichlet** *int*: (into=[0,1]) This option (by default 0) is used to extend control volumes for the momentum equation.
- **cl_pression_sommet_faible** *int*: (into=[0,1]) This option is used to specify a strong formulation (value set to 0, the default) or a weak formulation (value set to 1) for an imposed pressure boundary condition. The first formulation converges quicker and is stable in general cases. The second formulation should be used if there are several outlet boundaries with Neumann condition (see *Ecoulement_Neumann* test case for example).

14 domaine

Description: Keyword to create a domain.

See also: [objet_u \(36\)](#) [domaine_ale \(14.1\)](#)

Usage:

14.1 domaine_ale

Description: Domain with nodes at the interior of the domain which are displaced in an arbitrarily prescribed way thanks to ALE (Arbitrary Lagrangian-Eulerian) description.

Keyword to specify that the domain is mobile following the displacement of some of its boundaries.

See also: [domaine \(14\)](#)

Usage:

15 espece

Description: `not_set`

See also: [objet_u \(36\)](#)

Usage:

```

espece obj Lire obj {
    cp champ_base

```

```

    mu champ_base
    masse_molaire float
}
where

```

- **cp** *champ_base* (16.1): Specific heat value (J.kg-1.K-1).
- **mu** *champ_base* (16.1): Dynamic viscosity value (kg.m-1.s-1).
- **masse_molaire** *float*: Gas molar mass.

16 champ_base

16.1 champ_base

Description: Basic class of fields.

See also: [objet_u \(36\)](#) [champ_don_base \(16.5\)](#) [champ_ostwald \(16.18\)](#) [champ_input_base \(16.16\)](#) [champ_fonc_med \(16.9\)](#) [Champ_Fonc_MEDfile \(16.3\)](#) [field_uniform_keps_from_ud \(16.26\)](#)

Usage:

16.2 Champ_Fonc_MED_Tabule

Description: not_set

See also: [champ_fonc_med \(16.9\)](#)

Usage:

```

Champ_Fonc_MED_Tabule [ use_existing_domain ] [ last_time ] filename domain_name field_name location time
where

```

- **use_existing_domain** *str* into [*'use_existing_domain'*]
- **last_time** *str* into [*'last_time'*]: to use the last time of the MED file instead of the specified time.
- **filename** *str*: Name of the .med file.
- **domain_name** *str*: Name of the domain.
- **field_name** *str*: Name of the problem unknown.
- **location** *str* into [*'som'*, *'elem'*]: To indicate where the field has been post-processed.
- **time** *float*: Time of the field in the .med file.

16.3 Champ_Fonc_MEDfile

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

See also: [champ_base \(16.1\)](#)

Usage:

16.4 Champ_Tabule_Morceaux

Description: set Tabulated field by sub-zone

See also: [champ_don_base \(16.5\)](#)

Usage:

Champ_Tabule_Morceaux **dom_name** **nb_comp** **data**

where

- **dom_name** *str*: Name of the domain
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* (3.6): subzone_1 nb_comp InputFieldName { table_dim InputFieldVal_1 InputFieldVal_2 OutputFieldVal_1 OutputFieldVal_2 ... } subzone_2 nb_comp InputFieldName { table_dim InputFieldVal_1 InputFieldVal_2 OutputFieldVal_1 OutputFieldVal_2 ... } subzone_n nb_comp InputFieldName { table_dim InputFieldVal_1 InputFieldVal_2 OutputFieldVal_1 OutputFieldVal_2 ... }

16.5 champ_don_base

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

See also: [champ_base](#) (16.1) [uniform_field](#) (16.29) [champ_uniforme_morceaux](#) (16.22) [champ_fonc_xyz](#) (16.25) [champ_fonc_txyz](#) (16.24) [champ_don_lu](#) (16.6) [init_par_partie](#) (16.27) [champ_tabule_temps](#) (16.21) [champ_fonc_t](#) (16.12) [champ_fonc_tabule](#) (16.13) [champ_init_canal_sinal](#) (16.14) [champ_som_lu_vdf](#) (16.19) [champ_som_lu_vef](#) (16.20) [tayl_green](#) (16.28) [champ_fonc_reprise](#) (16.10) [Champ_Tabule_Morceaux](#) (16.4)

Usage:

16.6 champ_don_lu

Description: Field to read a data field (values located at the center of the cells) in a file.

See also: [champ_don_base](#) (16.5)

Usage:

champ_don_lu **dom** **nb_comp** **file**

where

- **dom** *str*: Name of the domain.
- **nb_comp** *int*: Number of field components.
- **file** *str*: Name of the file.
This file has the following format:
nb_val_lues -> Number of values readen in th file
Xi Yi Zi -> Coordinates readen in the file
Ui Vi Wi -> Value of the field

16.7 champ_fonc_fonction

Description: Field that is a function of another field.

See also: [champ_fonc_tabule](#) (16.13) [champ_fonc_fonction_txyz](#) (16.8)

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* (3.6): Values (the table (the value of the field at any time is calculated by linear interpolation from this table) or the analytical expression (with keyword expression to use an analytical expression)).

16.8 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 (16.7)

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* (3.6): Values (the table (the value of the field at any time is calculated by linear interpolation from this table) or the analytical expression (with keyword expression to use an analytical expression)).

16.9 champ_fonc_med

Description: Field to read a data field in a MED-format file .med at a specified time. It is very useful, for example, to resume a calculation with a new or refined geometry. The field post-processed on the new geometry at med format is used as initial condition for the resume.

See also: champ_base (16.1) Champ_Fonc_MED_Tabule (16.2)

Usage:

champ_fonc_med [use_existing_domain] [last_time] filename domain_name field_name location time

where

- **use_existing_domain** *str* into ['use_existing_domain']
- **last_time** *str* into ['last_time']: to use the last time of the MED file instead of the specified time.
- **filename** *str*: Name of the .med file.
- **domain_name** *str*: Name of the domain.
- **field_name** *str*: Name of the problem unknown.
- **location** *str* into ['som', 'elem']: To indicate where the field has been post-processed.
- **time** *float*: Time of the field in the .med file.

16.10 champ_fonc_reprise

Description: This field is used to read a data field in a save file (.xyz or .sauv) at a specified time. It is very useful, for example, to run a thermohydraulic calculation with velocity initial condition read into a save file from a previous hydraulic calculation.

See also: champ_don_base (16.5)

Usage:

champ_fonc_reprise [format] filename pb_name champ [fonction] temps

where

- **format** *str* into ['binaire', 'formatte', 'xyz']: Type of file (the file format). If xyz format is activated, the .xyz file from the previous calculation will be given for filename, and if formatte or binaire is choosen, the .sauv file of the previous calculation will be specified for filename. In the case of a parallel calculation, if the mesh partition does not changed between the previous calculation and the next one, the binaire format should be preferred, because is faster than the xyz format.
- **filename** *str*: Name of the save file.
- **pb_name** *str*: Name of the problem.
- **champ** *str*: Name of the problem unknown. It may also be the temporal average of a problem unknown (like moyenne_vitesse, moyenne_temperature,...)
- **fonction** *fonction_champ_reprise* (16.11): Optional keyword to apply a function on the field being read in the save file (e.g. to read a temperature field in Celsius units and convert it for the calculation on Kelvin units, you will use: fonction 1 273.+val)
- **temps** *str*: Time of the saved field in the save file or last_time. If you give the keyword last_time instead, the last time saved in the save file will be used.

16.11 fonction_champ_reprise

Description: not_set

See also: objet_lecture (35)

Usage:

mot fonction

where

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

16.12 champ_fonc_t

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

See also: champ_don_base (16.5)

Usage:

champ_fonc_t val

where

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

16.13 champ_fonc_tabule

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

See also: champ_don_base (16.5) champ_fonc_fonction (16.7)

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* (3.6): Values (the table (the value of the field at any time is calculated by linear interpolation from this table) or the analytical expression (with keyword expression to use an analytical expression)).

16.14 champ_init_canal_sinal

Description: For a parabolic profile on U velocity with an unpredictable disturbance on V and W and a sinusoidal disturbance on V velocity.

See also: *champ_don_base* (16.5)

Usage:

champ_init_canal_sinal *dim bloc*

where

- **dim** *int*: Number of field components.
- **bloc** *bloc_lec_champ_init_canal_sinal* (16.15): Parameters for the class *champ_init_canal_sinal*.

16.15 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* (35)

Usage:

```
{
    ucent float
    h float
    ampli_bruit float
    [ ampli_sin float ]
    omega float
    [ dir_flow int into [0, 1, 2] ]
    [ dir_wall int into [0, 1, 2] ]
    [ min_dir_flow float ]
    [ min_dir_wall float ]
}
```

where

- **ucent** *float*: Velocity value at the center of the channel.
- **h** *float*: Half length of the channel.
- **ampli_bruit** *float*: Amplitude for the disturbance.
- **ampli_sin** *float*: Amplitude for the sinusoidal disturbance (by default equals to ucent/10).
- **omega** *float*: Value of pulsation for the of the sinusoidal disturbance.

- **dir_flow** *int into [0, 1, 2]*: Flow direction for the initialization of the flow in a channel.
 - if dir_flow=0, the flow direction is X
 - if dir_flow=1, the flow direction is Y
 - if dir_flow=2, the flow direction is Z
 Default value for dir_flow is 0
- **dir_wall** *int into [0, 1, 2]*: Wall direction for the initialization of the flow in a channel.
 - if dir_wall=0, the normal to the wall is in X direction
 - if dir_wall=1, the normal to the wall is in Y direction
 - if dir_wall=2, the normal to the wall is in Z direction
 Default value for dir_flow is 1
- **min_dir_flow** *float*: Value of the minimum coordinate in the flow direction for the initialization of the flow in a channel. Default value for dir_flow is 0.
- **min_dir_wall** *float*: Value of the minimum coordinate in the wall direction for the initialization of the flow in a channel. Default value for dir_flow is 0.

16.16 champ_input_base

Description: not_set

See also: champ_base (16.1) champ_input_p0 (16.17)

Usage:

champ_input_base obj Lire obj {

```

    nb_comp  int
    nom      str
    [ initial_value  n x1 x2 ... xn]
    probleme  str
    [ sous_zone  str]

```

}

where

- **nb_comp** *int*
- **nom** *str*
- **initial_value** *n x1 x2 ... xn*
- **probleme** *str*
- **sous_zone** *str*

16.17 champ_input_p0

Description: not_set

See also: champ_input_base (16.16)

Usage:

champ_input_p0 obj Lire obj {

```

    nb_comp  int
    nom      str
    [ initial_value  n x1 x2 ... xn]
    probleme  str
    [ sous_zone  str]

```

}
where

- **nb_comp** *int* for inheritance
- **nom** *str* for inheritance
- **initial_value** *n x1 x2 ... xn* for inheritance
- **probleme** *str* for inheritance
- **sous_zone** *str* for inheritance

16.18 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 \(16.1\)](#)

Usage:
champ_ostwald

16.19 champ_som_lu_vdf

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

See also: [champ_don_base \(16.5\)](#)

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 ...

16.20 champ_som_lu_vdf

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

See also: [champ_don_base \(16.5\)](#)

Usage:
champ_som_lu_vdf domain_name dim tolerance file
where

- **domain_name** *str*: Name of the domain.
- **dim** *int*: Value of the dimension of the field.
- **tolerance** *float*: Value of the tolerance to check the coordinates of the nodes.

- **file** *str*: Name of the file.
This file has the following format:
Xi Yi Zi -> Coordinates of the node
Ui Vi Wi -> Value of the field on this node
Xi+1 Yi+1 Zi+1 -> Next point
Ui+1 Vi+1 Zi+1 -> Next value ...

16.21 champ_tabule_temps

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

See also: champ_don_base (16.5)

Usage:

champ_tabule_temps dim bloc
where

- **dim** *int*: Number of field components.
- **bloc** *bloc_lecture* (3.6): Values as a table. The value of the field at any time is calculated by linear interpolation from this table.

16.22 champ_uniforme_morceaux

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

See also: champ_don_base (16.5) champ_uniforme_morceaux_tabule_temps (16.23) valeur_totale_sur_volume (16.30)

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* (3.6): { 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.

16.23 champ_uniforme_morceaux_tabule_temps

Description: this type of field is constant in space on one or several sub_zones and tabulated as a function of time.

See also: champ_uniforme_morceaux (16.22)

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* (3.6): { 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.

16.24 champ_fonc_txyz

Description: Field defined by analytical functions. It makes it possible the definition of a field that depends on the time and the space.

See also: champ_don_base (16.5)

Usage:

champ_fonc_txyz **dom** **val**
where

- **dom** *str*: Name of domain of calculation.
- **val** *n word1 word2 ... wordn*: List of functions on (t,x,y,z).

16.25 champ_fonc_xyz

Description: Field defined by analytical functions. It makes it possible the definition of a field that depends on (x,y,z).

See also: champ_don_base (16.5)

Usage:

champ_fonc_xyz **dom** **val**
where

- **dom** *str*: Name of domain of calculation.
- **val** *n word1 word2 ... wordn*: List of functions on (x,y,z).

16.26 field_uniform_keps_from_ud

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

See also: champ_base (16.1)

Usage:

field_uniform_keps_from_ud **obj** Lire obj {

u *float*
d *float*

}

where

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

16.27 **init_par_partie**

Description: ne marche que pour `n_comp=1`

See also: `champ_don_base` ([16.5](#))

Usage:

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

where

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

16.28 **tayl_green**

Description: Class `Tayl_green`.

See also: `champ_don_base` ([16.5](#))

Usage:

tayl_green **dim**

where

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

16.29 **uniform_field**

Synonymous: **champ_uniforme**

Description: Field that is constant in space and stationary.

See also: `champ_don_base` ([16.5](#))

Usage:

uniform_field **val**

where

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

16.30 **valeur_totale_sur_volume**

Description: Similar as `Champ_Uniforme_Morceaux` with the same syntax. Used for source terms when we want to specify a source term with a value given for the volume (eg: heat in Watts) and not a value per volume unit (eg: heat in Watts/m3).

See also: `champ_uniforme_morceaux` ([16.22](#))

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* (3.6): { 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.

17 champ_front_base

17.1 champ_front_base

Description: Basic class for fields at domain boundaries.

See also: objet_u (36) champ_front_uniforme (17.29) champ_front_fonc_xyz (17.21) champ_front_fonc_txyz (17.20) champ_front_fonc_pois_ipsn (17.17) champ_front_fonc_pois_tube (17.18) champ_front_tabule (17.27) champ_front_fonction (17.22) champ_front_bruite (17.10) champ_front_tangentiel_vef (17.28) champ_front_lu (17.23) boundary_field_inward (17.5) champ_front_pression_from_u (17.25) champ_front_contact_vef (17.14) champ_front_calc (17.11) champ_front_recyclage (17.26) ch_front_input (17.7) champ_front_normal_vef (17.24) champ_front_fonc_t (17.19) champ_front_xyz_debit (17.31) champ_front_MED (17.9) champ_front_debit_massique (17.16) champ_front_debit (17.15) Champ_front_debit_QC_VDF (17.4) boundary_field_uniform_keps_from_ud (17.6) champ_front_vortex (17.30) champ_front_zoom (17.32) Champ_front_ale (17.3) Ch_front_input_ALE (17.2)

Usage:

17.2 Ch_front_input_ALE

Description: Class to define a boundary condition on a moving boundary of a mesh (only for the Arbitrary Lagrangian-Eulerian framework).

Example: Ch_front_input_ALE { nb_comp 3 nom VITESSE_IN_ALE probleme pb initial_value 3 1. 0. 0. }

See also: champ_front_base (17.1)

Usage:

17.3 Champ_front_ale

Description: Class to define a boundary condition on a moving boundary of a mesh (only for the Arbitrary Lagrangian-Eulerian framework).

See also: champ_front_base (17.1)

Usage:

Champ_front_ale **val**
where

- **val** *n word1 word2 ... wordn*:
Example: 2 -y*0.01 x*0.01

17.4 Champ_front_debit_QC_VDF

Description: This field is used to define a flow rate field for quasi-compressible fluids in VDF discretization. The flow rate is kept constant during a transient.

See also: `champ_front_base` ([17.1](#))

Usage:

Champ_front_debit_QC_VDF *dimension* *liste* [*moyen*] *pb_name*

where

- **dimension** *int*: Problem dimension
- **liste** *bloc_lecture* ([3.6](#)): List of the mass flow rate values [kg/s/m2] with the following syntaxe: { val1 ... valdim }
- **moyen** *str*: Option to use rho mean value
- **pb_name** *str*: Problem name

17.5 boundary_field_inward

Description: this field is used to define the normal vector field standard at the boundary in VDF or VEF discretization.

See also: `champ_front_base` ([17.1](#))

Usage:

boundary_field_inward *obj* Lire *obj* {

normal_value *str*

}

where

- **normal_value** *str*: normal vector value (positive value for a vector oriented outside to inside) which can depend of the time.

17.6 boundary_field_uniform_keps_from_ud

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

See also: `champ_front_base` ([17.1](#))

Usage:

boundary_field_uniform_keps_from_ud *obj* Lire *obj* {

u *float*

d *float*

}

where

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

17.7 ch_front_input

Description: not_set

See also: champ_front_base ([17.1](#)) ch_front_input_uniforme ([17.8](#))

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*

17.8 ch_front_input_uniforme

Description: for coupling, you can use ch_front_input_uniforme which is a champ_front_uniforme, which use an external value. It must be used with Problem.setInputField.

See also: ch_front_input ([17.7](#))

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

17.9 champ_front_MED

Description: Field allowing the loading of a boundary condition from a MED file using Champ_fonc_med

See also: champ_front_base ([17.1](#))

Usage:

champ_front_MED champ_fonc_med

where

- **champ_fonc_med** *champ_base* (16.1): a champ_fonc_med loading the values of the unknown on a domain boundary

17.10 champ_front_bruite

Description: Field which is variable in time and space in a random manner.

See also: champ_front_base (17.1)

Usage:

champ_front_bruite nb_comp bloc

where

- **nb_comp** *int*: Number of field components.
- **bloc** *bloc_lecture* (3.6): { [N val L val] Moyenne m_1.....[m_i] Amplitude A_1.....[A_i] }: Random noise: If N and L are not defined, the ith component of the field varies randomly around an average value m_i with a maximum amplitude A_i.
White noise: If N and L are defined, these two additional parameters correspond to L, the domain length and N, the number of nodes in the domain. Noise frequency will be between $2\pi/L$ and $2\pi N/(4L)$.
For example, formula for velocity: $u=U0(t)$ $v=U1(t)Uj(t)=Mj+2Aj*bruit_blanc$ where bruit_blanc (white_noise) is the formula given in the mettre_a_jour (update) method of the Champ_front_bruite (noise_boundary_field) (Refer to the Ch_fr_bruite.cpp file).

17.11 champ_front_calc

Description: This keyword is used on a boundary to get a field from another boundary. The local and remote boundaries should have the same mesh. If not, the Champ_front_recyclage keyword could be used instead. It is used in the condition block at the limits of equation which itself refers to a problem called pb1. We are working under the supposition that pb1 is coupled to another problem.

See also: champ_front_base (17.1)

Usage:

champ_front_calc problem_name bord field_name

where

- **problem_name** *str*: Name of the other problem to which pb1 is coupled.
- **bord** *str*: Name of the side which is the boundary between the 2 domains in the domain object description associated with the problem_name object.
- **field_name** *str*: Name of the field containing the value that the user wishes to use at the boundary. The field_name object must be recognized by the problem_name object.

17.12 champ_front_contact_rayo_semi_transp_vef

Description: This field is used on a boundary between a solid and fluid domain to exchange a calculated temperature at the contact face of the two domains according to the flux of the two problems with radiation in semi transparent fluid.

See also: `champ_front_contact_vef` ([17.14](#))

Usage:

champ_front_contact_rayo_semi_transp_vef **local_pb** **local_boundary** **remote_pb** **remote_boundary**

where

- **local_pb** *str*: Name of the problem.
- **local_boundary** *str*: Name of the boundary.
- **remote_pb** *str*: Name of the second problem.
- **remote_boundary** *str*: Name of the boundary in the second problem.

17.13 champ_front_contact_rayo_transp_vef

Description: This field is used on a boundary between a solid and fluid domain to exchange a calculated temperature at the contact face of the two domains according to the flux of the two problems with radiation in transparent fluid.

See also: `champ_front_contact_vef` ([17.14](#))

Usage:

champ_front_contact_rayo_transp_vef **local_pb** **local_boundary** **remote_pb** **remote_boundary**

where

- **local_pb** *str*: Name of the problem.
- **local_boundary** *str*: Name of the boundary.
- **remote_pb** *str*: Name of the second problem.
- **remote_boundary** *str*: Name of the boundary in the second problem.

17.14 champ_front_contact_vef

Description: This field is used on a boundary between a solid and fluid domain to exchange a calculated temperature at the contact face of the two domains according to the flux of the two problems.

See also: `champ_front_base` ([17.1](#)) `champ_front_contact_rayo_transp_vef` ([17.13](#)) `champ_front_contact_rayo_semi_transp_vef` ([17.12](#))

Usage:

champ_front_contact_vef **local_pb** **local_boundary** **remote_pb** **remote_boundary**

where

- **local_pb** *str*: Name of the problem.
- **local_boundary** *str*: Name of the boundary.
- **remote_pb** *str*: Name of the second problem.
- **remote_boundary** *str*: Name of the boundary in the second problem.

17.15 champ_front_debit

Description: This field is used to define a flow rate field instead of a velocity field for a Dirichlet boundary condition on Navier-Stokes equations.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_debit ch

where

- **ch** `champ_front_base` ([17.1](#)): uniform field in space to define the flow rate. It could be, for example, `champ_front_uniforme`, `ch_front_input_uniform` or `champ_front_fonc_txyz` that depends only on time.

17.16 champ_front_debit_massique

Description: This field is used to define a flow rate field using the density

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_debit_massique ch

where

- **ch** `champ_front_base` ([17.1](#)): uniform field in space to define the flow rate. It could be, for example, `champ_front_uniforme`, `ch_front_input_uniform` or `champ_front_fonc_txyz` that depends only on time.

17.17 champ_front_fonc_pois_ipsn

Description: Boundary field `champ_front_fonc_pois_ipsn`.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_fonc_pois_ipsn r_tube umoy r_loc

where

- **r_tube** *float*
- **umoy** *n x1 x2 ... xn*
- **r_loc** *x1 x2 (x3)*

17.18 champ_front_fonc_pois_tube

Description: Boundary field `champ_front_fonc_pois_tube`.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_fonc_pois_tube r_tube umoy r_loc r_loc_mult

where

- **r_tube** *float*
- **umoy** *n x1 x2 ... xn*
- **r_loc** *x1 x2 (x3)*
- **r_loc_mult** *n1 n2 (n3)*

17.19 **champ_front_fonc_t**

Description: Boundary field that depends only on time.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_fonc_t val

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

17.20 **champ_front_fonc_txyz**

Description: Boundary field which is not constant in space and in time.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_fonc_txyz val

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

17.21 **champ_front_fonc_xyz**

Description: Boundary field which is not constant in space.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_fonc_xyz val

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

17.22 **champ_front_fonction**

Description: boundary field that is function of another field

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_fonction dim inco expression

where

- **dim** *int*: Number of field components.

- **inco** *str*: Name of the field (for example: temperature).
- **expression** *str*: keyword to use a analytical expression like $10.*EXP(-0.1*val)$ where val be the keyword for the field.

17.23 champ_front_lu

Description: boundary field which is given from data issued from a read file. The format of this file has to be the same that the one generated by `Ecrire_fichier_xyz_valeur`

Example for K and epsilon quantities to be defined for inlet condition in a boundary named 'entree':

entree frontiere_ouverte_K_Eps_impose Champ_Front_lu dom 2pb_K_EPS_PERIO_1006.306198.dat

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_lu **domaine** **dim** **file**

where

- **domaine** *str*: Name of domain
- **dim** *int*: number of components
- **file** *str*: path for the read file

17.24 champ_front_normal_vef

Description: Field to define the normal vector field standard at the boundary in VEF discretization.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_normal_vef **mot** **vit_tan**

where

- **mot** *str* into [*'valeur_normale'*]: Name of vector field.
- **vit_tan** *float*: normal vector value (positive value for a vector oriented outside to inside).

17.25 champ_front_pression_from_u

Description: this field is used to define a pressure field depending of a velocity field.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_pression_from_u **expression**

where

- **expression** *str*: value depending of a velocity (like $2 * u_{moy}^2$).

17.26 champ_front_recyclage

Description: This keyword is used on a boundary to get a field from another boundary. New keyword since the 1.6.1 version which replaces and generalizes several obsolete ones:

Champ_front_calc_intern
 Champ_front_calc_recycl_fluct_pbperio
 Champ_front_calc_recycl_champ
 Champ_front_calc_intern_2pbs
 Champ_front_calc_recycl_fluct

It is to use, in a general way, on a boundary of a local_pb problem, a field calculated from a linear combination of an imposed field $g(x,y,z,t)$ with an instantaneous $f(x,y,z,t)$ and a spatial mean field $\langle f \rangle(t)$ or a temporal mean field $\langle f \rangle(x,y,z)$ extracted from a plane of a problem named pb (pb may be local_pb itself):

For each component i , the field F applied on the boundary will be:

$$F_i(x,y,z,t) = \alpha_i * g_i(x,y,z,t) + \chi_i * [f_i(x,y,z,t) - \beta_i * \langle f_i \rangle]$$

Usage:

```

Champ_front_recyclage {

    pb_champ_evaluateur problem_name field nb_comp
    [ distance_plan x1 x2 (x3) ]
    [ moyenne_imposee methode_moy [fichier file [second_file]] ]
    [ moyenne_recyclee methode_recyc [fichier file [second_file]] ]
    [ direction_anisotrope int ]
    [ ampli_moyenne_imposee n x1 x2 ... xn ]
    [ ampli_moyenne_recyclee n x1 x2 ... xn ]
    [ ampli_fluctuation n x1 x2 ... xn ]
}

```

where:

- **pb_champ_evaluateur** *problem_name field nb_comp*: To give the name of the problem, the name of the field of the problem and its number of components nb_comp.
- **distance_plan** *x1 x2 (x3)*: Vector which gives the distance between the boundary and the plane from where the field F will be extracted. By default, the vector is zero, that should imply the two domains have coincident boundaries.
- **ampli_moyenne_imposee** *2|3 alpha(0) alpha(1) [alpha(2)]*: α_i coefficients (by default =1)
- **ampli_moyenne_recyclee** *2|3 beta(0) beta(1) [beta(2)]*: β_i coefficients (by default =1)
- **ampli_fluctuation** *2|3 gamma(0) gamma(1) [gamma(2)]*: γ_i coefficients (by default =1)
- **direction_anisotrope** *int into [1,2,3]*: If an integer is given for direction (X:1, Y:2, Z:3, by default, direction is negative), the imposed field g will be 0 for the 2 other directions.
- **moyenne_imposee** *methode_moy*: Value of the imposed g field. The *methode_moy* option can be:

profil *[2|3] valx(x,y,z,t) valy(x,y,z,t) [valz(x,y,z,t)]*: To specify analytic profile for the imposed g field.

interpolation fichier file: To create an imposed field built by interpolation of values read from a file. The imposed field is applied on the direction given by the keyword **direction_anisotrope** (the field is zero for the other directions). The format of the file is:

```

pos(1) val(1)
pos(2) val(2)
...
pos(N) val(N)

```

If direction given by `direction_anisotrope` is 1 (or 2 or 3), then `pos` will be X (or Y or Z) coordinate and `val` will be X value (or Y value, or Z value) of the imposed field.

connexion_approchee fichier file: To read the imposed field from a file where positions and values are given (it is not necessary that the coordinates of points match the coordinates of the boundary faces, indeed, the nearest point of each face of the boundary will be used). The format of the file is:

```
N
x(1) y(1) [z(1)] valx(1) valy(1) [valz(1)]
x(2) y(2) [z(2)] valx(2) valy(2) [valz(2)]
...
x(N) y(N) [z(N)] valx(N) valy(N) [valz(N)]
```

connection_exacte fichier file second_file: To read the imposed field from two files. The first file contains the points coordinates (which should be the same as the coordinates of the boundary faces) and the `second_file` contains the mean values. The format of the first file is:

```
N
1 x(1) y(1) [z(1)]
2 x(2) y(2) [z(2)]
...
N x(N) y(N) [z(N)]
```

while the format of the `second_file` is:

```
N
1 valx(1) valy(1) [valz(1)]
2 valx(2) valy(2) [valz(2)]
...
N valx(N) valy(N) [valz(N)]
```

logarithmique diametre float u_tau float visco_cin float direction int: To specify the imposed field (in this case, velocity) by an analytical logarithmic law of the wall:

$g(x,y,z) = u_tau * (\log(0.5*diametre*u_tau/visco_cin)/Kappa + 5.1)$

with $g(x,y,z)=u(x,y,z)$ if **direction** is set to 1 ($g=v(x,y,z)$ if **direction** is set to 2, and $g=w(x,y,z)$ if it is set to 3)

- **moyenne_recylee methode_recyc:** Method used to perform a spatial or a temporal averaging of `f` field to specify `<f>`. `<f>` can be the surface mean of `f` on the plane (surface option, see below) or it can be read from several files (for example generated by the `chmoy_faceperio` option of the `Traitement_particulier` keyword to obtain a temporal mean field). The option `methode_recyc` can be:

surfactive: Surface mean for `<f>` from `f` values on the plane

Or one of the following `methode_moy` options applied to read a temporal mean field `<f>(x,y,z)`:

interpolation

connexion_approchee

connexion_exacte

See also: `champ_front_base` (17.1)

Usage:

champ_front_recyclage bloc

where

- **bloc str**

17.27 champ_front_tabule

Description: Constant field on the boundary, tabulated as a function of time.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_tabule nb_comp bloc

where

- **nb_comp** *int*: Number of field components.
 - **bloc** *bloc_lecture* ([3.6](#)): {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.

17.28 champ_front_tangentiel_vef

Description: Field to define the tangential velocity vector field standard at the boundary in VEF discretization.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_tangentiel_vef mot vit_tan

where

- **mot** *str into* [*'vitesse_tangentielle'*]: Name of vector field.
- **vit_tan** *float*: Vector field standard [m/s].

17.29 champ_front_uniforme

Description: Boundary field which is constant in space and stationary.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_uniforme val

where

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

17.30 champ_front_vortex

Description: not_set

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_vortex dom geom nu utau

where

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

- **geom** *str*
- **nu** *float*
- **utau** *float*

17.31 champ_front_xyz_debit

Description: This field is used to define a flow rate field with a velocity profil which will be normalized to match the flow rate chosen.

See also: `champ_front_base` ([17.1](#))

Usage:

```
champ_front_xyz_debit obj Lire obj {
    [ velocity_profil champ_front_base]
    flow_rate champ_front_base
}
```

where

- **velocity_profil** *champ_front_base* ([17.1](#)): `velocity_profil` 0 velocity field to define the profil of velocity.
- **flow_rate** *champ_front_base* ([17.1](#)): `flow_rate` 1 uniform field in space to define the flow rate. It could be, for example, `champ_front_uniforme`, `ch_front_input_uniform` or `champ_front_fonc_t`

17.32 champ_front_zoom

Description: Basic class for fields at boundaries of two problems (global problem and local problem).

See also: `champ_front_base` ([17.1](#))

Usage:

```
champ_front_zoom pbMg pb_1 pb_2 bord inco
where
```

- **pbMg** *str*: Name of multi-grid problem.
- **pb_1** *str*: Name of first problem.
- **pb_2** *str*: Name of second problem.
- **bord** *str*: Name of bord.
- **inco** *str*: Name of field.

18 loi_etat_base

Description: Basic class for state laws.

See also: `objet_u` ([36](#)) `gaz_parfait` ([18.3](#)) `gaz_reel_rhot` ([18.1](#)) `melange_gaz_parfait` ([18.2](#))

Usage:

18.1 gaz_reel_rhot

Description: Real gas.

See also: [loi_etat_base \(18\)](#)

Usage:

gaz_reel_rhot bloc

where

- **bloc** *bloc_lecture* ([3.6](#)): Description.

18.2 melange_gaz_parfait

Description: Mixing of perfect gas.

See also: [loi_etat_base \(18\)](#)

Usage:

melange_gaz_parfait obj Lire obj {

```
    sc float  
    [ cp float]  
    prandtl float  
    [ correction_fraction ]  
    [ ignore_check_fraction ]  
    [ dtol_fraction float]
```

}

where

- **sc** *float*: Schmidt number of the gas $Sc = \nu/D$ (D: diffusion coefficient of the mixing).
- **cp** *float*: Specific heat at constant pressure of the gas Cp.
- **prandtl** *float*: Prandtl number of the gas $Pr = \mu * Cp / \lambda$
- **correction_fraction** : To force mass fractions between 0. and 1.
- **ignore_check_fraction** : Not to check if mass fractions between 0. and 1.
- **dtol_fraction** *float*: Delta tolerance on mass fractions for check testing (default value 1.e-6).

18.3 gaz_parfait

Description: Perfect gas.

See also: [loi_etat_base \(18\)](#)

Usage:

gaz_parfait obj Lire obj {

```
    Cp float  
    [ Cv float]  
    [ gamma float]  
    Prandtl float  
    [ rho_constant_pour_debug champ_base]
```

}

where

- **Cp** *float*: Specific heat at constant pressure (J/kg/K).
- **Cv** *float*: Specific heat at constant volume (J/kg/K).
- **gamma** *float*: C_p/C_v
- **Prandtl** *float*: Prandtl number of the gas $Pr = \mu * C_p / \lambda$
- **rho_constant_pour_debug** *champ_base* ([16.1](#))

19 loi_fermeture_base

Description: Class for appends fermeture to problem

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

See also: objet_u ([36](#)) loi_fermeture_test ([19.1](#))

Usage:

19.1 loi_fermeture_test

Description: Loi for test only

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

See also: loi_fermeture_base ([19](#))

Usage:

```
loi_fermeture_test obj Lire obj {
    [ coef float ]
}
```

where

- **coef** *float*: coefficient

20 loi_horaire

Description: to define the movement with a time-dependant law for the solid interface.

See also: objet_u ([36](#))

Usage:

```
loi_horaire obj Lire obj {
    position n word1 word2 ... wordn
    vitesse n word1 word2 ... wordn
    [ rotation n word1 word2 ... wordn ]
    [ derivee_rotation n word1 word2 ... wordn ]
}
```

where

- **position** *n word1 word2 ... wordn*
- **vitesse** *n word1 word2 ... wordn*
- **rotation** *n word1 word2 ... wordn*
- **derivee_rotation** *n word1 word2 ... wordn*

21 milieu_base

Description: Basic class for medium (physics properties of medium).

See also: [objet_u \(36\)](#) [solide \(21.7\)](#) [constituant \(21.1\)](#) [fluide_incompressible \(21.3\)](#) [fluide_diphasique \(21.2\)](#)

Usage:

```
milieu_base obj Lire obj {  
    [ rho champ_base]  
    [ cp champ_base]  
    [ lambda champ_base]  
}  
where
```

- **rho** *champ_base* ([16.1](#)): Density (kg.m-3).
- **cp** *champ_base* ([16.1](#)): Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* ([16.1](#)): Conductivity (W.m-1.K-1).

21.1 constituant

Description: Constituent.

See also: [milieu_base \(21\)](#)

Usage:

```
constituant obj Lire obj {  
    [ coefficient_diffusion champ_base]  
    [ rho champ_base]  
    [ cp champ_base]  
    [ lambda champ_base]  
}  
where
```

- **coefficient_diffusion** *champ_base* ([16.1](#)): Constituent diffusion coefficient value (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* ([16.1](#)) for inheritance: Density (kg.m-3).
- **cp** *champ_base* ([16.1](#)) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* ([16.1](#)) for inheritance: Conductivity (W.m-1.K-1).

21.2 fluide_diphasique

Description: Two-phase fluid.

See also: [milieu_base \(21\)](#)

Usage:

```
fluide_diphasique obj Lire obj {  
    sigma champ_don_base
```

```

fluide0 str
fluide1 str
[ chaleur_latente champ_don_base]
[ formule_mu str]
[ rho champ_base]
[ cp champ_base]
[ lambda champ_base]
}
where

```

- **sigma** *champ_don_base* (16.5): surfacic tension (J/m2)
- **fluide0** *str*: first phase fluid
- **fluide1** *str*: second phase fluid
- **chaleur_latente** *champ_don_base* (16.5): phase changement enthalpy $h(\text{phase1}_-) - h(\text{phase0}_-)$ (J/kg/K)
- **formule_mu** *str*: (into=[standard,arithmetic,harmonic]) formula used to calculate average
- **rho** *champ_base* (16.1) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).

21.3 fluide_incompressible

Description: This is a uncompressible fluid.

See also: milieu_base (21) fluide_quasi_compressible (21.5) fluide_ostwald (21.4)

Usage:

fluide_incompressible obj Lire obj {

```

[ beta_th champ_base]
[ mu champ_base]
[ beta_co champ_base]
[ indice champ_base]
[ kappa champ_base]
[ rho champ_base]
[ cp champ_base]
[ lambda champ_base]
}
where

```

- **beta_th** *champ_base* (16.1): Thermal expansion (K-1).
- **mu** *champ_base* (16.1): Dynamic viscosity (kg.m-1.s-1).
- **beta_co** *champ_base* (16.1): Volume expansion coefficient values in concentration.
- **indice** *champ_base* (16.1): Refractivity of fluid.
- **kappa** *champ_base* (16.1): Absorptivity of fluid (m-1).
- **rho** *champ_base* (16.1) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).

21.4 fluide_ostwald

Description: Non-Newtonian fluids governed by Ostwald's law. The law applicable to stress tensor is:

$\tau = K(T) \cdot (D:D/2)^{((n-1)/2)} \cdot D$ Where:

D refers to the deformation tensor

K refers to fluid consistency (may be a function of the temperature T)

n refers to the fluid structure index $n=1$ for a Newtonian fluid, $n<1$ for a rheofluidifier fluid, $n>1$ for a rheothickening fluid.

See also: fluide_incompressible (21.3)

Usage:

fluide_ostwald obj Lire obj {

```
[ k champ_base]
[ n champ_base]
[ beta_th champ_base]
[ mu champ_base]
[ beta_co champ_base]
[ indice champ_base]
[ kappa champ_base]
[ rho champ_base]
[ cp champ_base]
[ lambda champ_base]
```

}

where

- **k** *champ_base* (16.1): Fluid consistency.
- **n** *champ_base* (16.1): Fluid structure index.
- **beta_th** *champ_base* (16.1) for inheritance: Thermal expansion (K-1).
- **mu** *champ_base* (16.1) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **beta_co** *champ_base* (16.1) for inheritance: Volume expansion coefficient values in concentration.
- **indice** *champ_base* (16.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (16.1) for inheritance: Absorptivity of fluid (m-1).
- **rho** *champ_base* (16.1) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).

21.5 fluide_quasi_compressible

Description: Compressible flow at low mach number.

See also: fluide_incompressible (21.3)

Usage:

fluide_quasi_compressible obj Lire obj {

```
[ sutherland bloc_sutherland]
[ pression float]
[ loi_etat loi_etat_base]
[ traitement_ptn str into ['edo', 'constant', 'conservation_masse']]
[ traitement_rho_gravite str into ['standard', 'moins_rho_moyen']]
[ temps_debut_prise_en_compte_drho_dt float]
[ omega_relaxation_drho_dt float]
```

```

[ mu champ_base]
[ indice champ_base]
[ kappa champ_base]
[ rho champ_base]
[ cp champ_base]
[ lambda champ_base]
}
where

```

- **sutherland** *bloc_sutherland* (21.6): Sutherland law for viscosity and for conductivity.
- **pression** *float*: Initial pressure.
- **loi_etat** *loi_etat_base* (18): State law.
- **traitement_pth** *str into ['edo', 'constant', 'conservation_masse']*: Particular treatment for the thermodynamic pressure Pth ; there are three possibilities:
 - 1) with the keyword 'edo' the code computes Pth solving an O.D.E. ; in this case, the mass is not strictly conserved (it is the default case for quasi compressible computation):
 - 2) the keyword 'conservation_masse' forces the conservation of the mass (closed geometry or with periodic boundaries condition)
 - 3) the keyword 'constant' makes it possible to have a constant Pth ; it's the good choice when the flow is open (e.g. with pressure boundary conditions).
 It is possible to monitor the volume averaged value for temperature and density, plus Pth evolution in the .evol_glob file.
- **traitement_rho_gravite** *str into ['standard', 'moins_rho_moyen']*: It may be :1) 'standard': the gravity term is evaluated with $\rho \cdot g$ (It is the default). 2) 'moins_rho_moyen': the gravity term is evaluated with $(\rho - \rho_{\text{moy}}) \cdot g$. Unknown pressure is then $P^* = P + \rho_{\text{moy}} \cdot g \cdot z$. It is useful when you apply uniform pressure boundary condition like $P^* = 0$.
- **temps_debut_prise_en_compte_drho_dt** *float*: While time < value, $d\rho/dt$ is set to zero (ρ , volumic mass). Useful for some calculation during the first time steps with big variation of temperature and volumic mass.
- **omega_relaxation_drho_dt** *float*: Optional option to have a relaxed algorithm to solve the mass equation. value is used (1 per default) to specify omega.
- **mu** *champ_base* (16.1) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **indice** *champ_base* (16.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (16.1) for inheritance: Absorptivity of fluid (m-1).
- **rho** *champ_base* (16.1) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).

21.6 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 $\lambda(T) = \mu_0 \cdot C_p / Prandtl \cdot ((T_0 + \lambda_{\text{lambda}})/(T + \lambda_{\text{lambda}})) \cdot (T/T_0)^{1.5}$

See also: objet_lecture (35)

Usage:

```

m mu0 t t0 [ ms ] [ s ] mc c
where

```

- **m** *str into ['mu0']*
- **mu0** *float*
- **t** *str into ['T0']*
- **t0** *float*
- **ms** *str into ['lambda']*

- **s** *float*
- **mc** *str into ['C']*
- **c** *float*

21.7 solide

Description: Solid.

See also: `milieu_base` ([21](#))

Usage:

```
solide obj Lire obj {
    [ rho champ_base]
    [ cp champ_base]
    [ lambda champ_base]
}
```

where

- **rho** *champ_base* ([16.1](#)) for inheritance: Density (kg.m-3).
- **cp** *champ_base* ([16.1](#)) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* ([16.1](#)) for inheritance: Conductivity (W.m-1.K-1).

22 milieu_v2_base

Description: Basic class for medium (physics properties of medium) composed of constituents (fluids and solids).

See also: `objet_u` ([36](#))

Usage:

23 modele_rayonnement_base

Description: Basic class for wall thermal radiation model.

See also: `objet_u` ([36](#)) `modele_rayonnement_milieu_transparent` ([23.1](#))

Usage:

23.1 modele_rayonnement_milieu_transparent

Description: Wall thermal radiation model for a transparent gas and resolving a radiation-conduction-thermohydraulics coupled problem in VDF or VEF.

`Modele_Rayonnement_Milieu_Transparent` mod

```
Read mod {
    nom_pb_rayonnant
    problem_name
    fichier_fij
    file_name
    fichier_face_rayo
    file_name
```

[fichier_matrice | fichier_matrice_binaire file_name]

}

nom_pb_rayonnant problem_name : problem_name is the name of the radiating fluid problem

fichier_fij file_name : file_name is the name of the file which contains the shape factor matrix between all the faces.

fichier_face_rayo file_name : file_name is the name of the file which contains the radiating faces characteristics (area, emission value ...)

fichier_matrice|fichier_matrice_binaire file_name : file_name is the name of the ASCII (or binary) file which contains the inverted shape factor matrix. It is an optional keyword, if not defined, the inverted shape factor matrix will be calculated and written in a file.

The two first files can be generated by a preprocessor, they allow the radiating face characteristics to be entered (set of faces considered to be uniform with respect to radiation for emission value, flux, etc.) and the form factors for these various faces. These files have the following format:

File on radiating faces:

N M -> N nombre de faces rayonnantes (=bords) et

(N is the number of radiating faces (=edges) and

-> M nombre de faces rayonnantes a emissivitee non nulle

M equals the number of non-zero emission radiating faces

Nom(i) S(i) E(i) -> Nom du bord i, surface du bord i, valeur de

(Name of the edge i, surface area of the edge i)

-> l'emissivite (comprise entre 0 et 1) (emission value (between 0 and 1))

Exemple:

13 4

Gauche 50.0 0.0

Droit1 50.0 0.5

Bas 10.0 0.0

Haut 10.0 0.0

Arriere 5.0 0.0

Avant 5.0 0.0

Droit2 30.0 0.5

Bas1 40.0 0.0

Haut1 20.0 0.0

Avant1 20.0 0.0

Arriere1 20.0 0.0

Entree 20.0 0.5

Sortie 20.0 0.5

File on form factors:

N -> Nombre de faces rayonnantes (Number of radiating faces)

Fij -> Matrice des facteurs de formes avec i,j entre 1 et N (Matrix of form factors where i, j between 1 and N)

Example:

13

```
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.24 0.20 0.10 0.10 0.10 0.10 0.16
0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.40 0.00 0.00 0.00 0.00 0.20 0.10 0.10 0.10 0.10 0.10 0.00
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.00 0.15 0.10 0.10 0.15 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.30 0.00 0.10 0.10 0.00 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.20 0.10 0.00 0.10 0.10 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.20 0.10 0.10 0.00 0.10 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.30 0.00 0.10 0.10 0.00 0.10
```

0.00 0.40 0.00 0.00 0.00 0.00 0.00 0.20 0.10 0.10 0.10 0.10 0.00

Caution:

- a) The radiation model's precision is decided by the user when he/she names the domain edges. In fact, a radiating face is recognised by the preprocessor as the set of domain edges faces bearing the same name. Thus, if the user subdivides the edge into two edges which are named differently, he/she thus creates two radiating faces instead of one.
- b) The form factors are entered by the user, the preprocessor carries out no calculations other than checking preservation relationships on form factors.
- c) The fluid is considered to be a transparent gas.

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

See also: `modele_rayonnement_base` (23)

Usage:

modele_rayonnement_milieu_transparent bloc

where

- **bloc** *bloc_lecture* (3.6): See description.

24 modele_turbulence_scal_base

Description: Basic class for turbulence model for energy equation.

See also: `objet_u` (36) `prandtl` (24.1) `schmidt` (24.2) `sous_maille_dyn` (24.3)

Usage:

modele_turbulence_scal_base obj Lire obj {

turbulence_paro *turbulence_paro_scalaire_base*
[**dt_impr_nusselt** *float*]

}

where

- **turbulence_paro** *turbulence_paro_scalaire_base* (33): Keyword to set the wall law.
- **dt_impr_nusselt** *float*: Keyword to print local values of Nusselt number and temperature near a wall during a turbulent calculation. The values will be printed in the `_Nusselt.face` file each `dt_impr_nusselt` time period. The local Nusselt expression is as follows : $Nu = ((\lambda + \lambda_t)/\lambda) * d_{wall}/d_{eq}$ where `d_wall` is the distance from the first mesh to the wall and `d_eq` is given by the wall law. This option also gives the value of `d_eq` and $h = (\lambda + \lambda_t)/d_{eq}$ and the fluid temperature of the first mesh near the wall.

For the Neumann boundary conditions (`flux_impose`), the «equivalent» wall temperature given by the wall law is also printed (`Tparoi equiv.`) preceded for VEF calculation by the edge temperature «T face de bord».

24.1 prandtl

Description: The Prandtl model. For the scalar equations, only the model based on Reynolds analogy is available. If `K_Epsilon` was selected in the hydraulic equation, Prandtl must be selected for the convection-diffusion temperature equation coupled to the hydraulic equation and Schmidt for the concentration equations.

See also: `modele_turbulence_scal_base` (24)

Usage:

```
prandtl obj Lire obj {  
    [ prdt str]  
    [ prandt_turbulent_fonction_nu_t_alpha str]  
    turbulence_paro turbulence_paro_scalaire_base  
    [ dt_impr_nusselt float]  
}
```

where

- **prdt** *str*: Keyword to modify the constant (Prdt) of Prandtl model : $\text{Alphat} = \text{Nut} / \text{Prdt}$ Default value is 0.9
- **prandt_turbulent_fonction_nu_t_alpha** *str*: Optional keyword to specify turbulent diffusivity (by default, $\alpha_t = \text{nu}_t / \text{Prt}$) with another formulae, for example: $\alpha_t = \text{nu}_t^2 / (0.7 * \alpha + 0.85 * \text{nu}_t)$ with the string $\text{nu}_t * \text{nu}_t / (0.7 * \alpha + 0.85 * \text{nu}_t)$ where α is the thermal diffusivity.
- **turbulence_paro** *turbulence_paro_scalaire_base* (33) for inheritance: Keyword to set the wall law.
- **dt_impr_nusselt** *float* for inheritance: Keyword to print local values of Nusselt number and temperature near a wall during a turbulent calculation. The values will be printed in the `_Nusselt.face` file each `dt_impr_nusselt` time period. The local Nusselt expression is as follows : $\text{Nu} = ((\lambda + \lambda_t) / \lambda) * d_{\text{wall}} / d_{\text{eq}}$ where d_{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 $h = (\lambda + \lambda_t) / d_{\text{eq}}$ and the fluid temperature of the first mesh near the wall.
For the Neumann boundary conditions (flux_impose), the «equivalent» wall temperature given by the wall law is also printed (Tparoi equiv.) preceded for VEF calculation by the edge temperature «T face de bord».

24.2 schmidt

Description: The Schmidt model. For the scalar equations, only the model based on Reynolds analogy is available. If K_Epsilon was selected in the hydraulic equation, Prandtl must be selected for the convection-diffusion temperature equation coupled to the hydraulic equation and Schmidt for the concentration equations.

See also: `modele_turbulence_scal_base` (24)

Usage:

```
schmidt obj Lire obj {  
    [ scturb float]  
    turbulence_paro turbulence_paro_scalaire_base  
    [ dt_impr_nusselt float]  
}
```

where

- **scturb** *float*: Keyword to modify the constant (Sct) of Schmlidt model : $\text{Dt} = \text{Nut} / \text{Sct}$ Default value is 0.7.
- **turbulence_paro** *turbulence_paro_scalaire_base* (33) for inheritance: Keyword to set the wall law.
- **dt_impr_nusselt** *float* for inheritance: Keyword to print local values of Nusselt number and temperature near a wall during a turbulent calculation. The values will be printed in the `_Nusselt.face` file

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».

Description: Dynamic sub-grid turbulence modele.
Warning : Available in VDF only. Not coded in VEF yet.

Usage:

```
[ stabilise str into ['6_points', 'moy_euler', 'plans_paralleles']]
[ nb_points int]
turbulence_paroit turbulence_paroit_scalaire_base
[ dt_impr_nusselt float]
```

where

- 25 nom**

Usage:

where

- 271

25.1 nom_anonyme

Description: not_set

See also: nom (25)

Usage:

[**mot**]

where

- **mot** *str*: Chain of characters.

26 partitionneur_deriv

Description: not_set

See also: objet_u (36) metis (26.2) sous_zones (26.5) tranche (26.6) partition (26.3) fichier_decoupage (26.1) union (26.7) sous_domaine (26.4)

Usage:

partitionneur_deriv obj Lire obj {

 [**nb_parts** *int*]

}

where

- **nb_parts** *int*: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.1 fichier_decoupage

Description: This algorithm reads an array of integer values on the disc, one value for each mesh element. Each value is interpreted as the target part number $n \geq 0$ for this element. The number of parts created is the highest value in the array plus one. Empty parts can be created if some values are not present in the array.

The file format is ASCII, and contains space, tab or carriage-return separated integer values. The first value is the number nb_elem of elements in the domain, followed by nb_elem integer values (positive or zero). This algorithm has been designed to work together with the 'ecrire_decoupage' option. You can generate a partition with any other algorithm, write it to disc, modify it, and read it again to generate the .Zone files. Contrary to other partitioning algorithms, no correction is applied by default to the partition (eg. element 0 on processor 0 and corrections for periodic boundaries). If 'corriger_partition' is specified, these corrections are applied.

See also: partitionneur_deriv (26)

Usage:

fichier_decoupage obj Lire obj {

fichier *str*

 [**corriger_partition**]

 [**nb_parts** *int*]

}

where

- **fichier** *str*: FILENAME
- **corriger_partition**
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.2 metis

Description: Metis is an external partitioning library. It is a general algorithm that will generate a partition of the domain.

See also: `partitionneur_deriv` (26)

Usage:

```
metis obj Lire obj {
    [ kmetis ]
    [ use_weights ]
    [ nb_parts int ]
}
```

where

- **kmetis** : The default values are pmetis, default parameters are automatically chosen by Metis. 'kmetis' is faster than pmetis option but the last option produces better partitioning quality. In both cases, the partitioning quality may be slightly improved by increasing the nb_essais option (by default N=1). It will compute N partitions and will keep the best one (smallest edge cut number). But this option is CPU expensive, taking N=10 will multiply the CPU cost of partitioning by 10. Experiments show that only marginal improvements can be obtained with non default parameters.
- **use_weights** : If use_weights is specified, weighting of the element-element links in the graph is used to force metis to keep opposite periodic elements on the same processor. This option can slightly improve the partitioning quality but it consumes more memory and takes more time. It is not mandatory since a correction algorithm is always applied afterwards to ensure a correct partitioning for periodic boundaries.
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.3 partition

Synonymous: **decouper**

Description: This algorithm re-use the partition of the domain named DOMAINE_NAME. It is useful to partition for example a post processing domain. The partition should match with the calculation domain.

See also: `partitionneur_deriv` (26)

Usage:

```
partition obj Lire obj {
    domaine str
    [ nb_parts int ]
}
```

where

- **domaine** *str*: domain name

- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.4 sous_domaine

Description: Given a global partition of a global domain, 'sous-domaine' allows to produce a conform partition of a sub-domain generated from the bigger one using the keyword `create_domain_from_sous_zone`. The sub-domain will be partitionned in a conform fashion with the global domain.

See also: `partitionneur_deriv` (26)

Usage:

sous_domaine obj Lire obj {

```

    fichier  str
    fichier_ssz str
    [ nb_parts int]

```

}

where

- **fichier** *str*: fichier domaine
- **fichier_ssz** *str*: fichier sous zone
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.5 sous_zones

Description: This algorithm will create one part for each specified subzone. All elements contained in the first subzone are put in the first part, all remaining elements contained in the second subzone in the second part, etc...

If all elements of the domain are contained in the specified subzones, then N parts are created, otherwise, a supplemental part is created with the remaining elements.

If no subzone is specified, all subzones defined in the domain are used to split the mesh.

See also: `partitionneur_deriv` (26)

Usage:

sous_zones obj Lire obj {

```

    sous_zones n word1 word2 ... wordn
    [ nb_parts int]

```

}

where

- **sous_zones** *n word1 word2 ... wordn*: N SUBZONE_NAME_1 SUBZONE_NAME_2 ...
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.6 tranche

Description: This algorithm will create a geometrical partitioning by slicing the mesh in the two or three axis directions, based on the geometric center of each mesh element. `nz` must be given if `dimension=3`. Each slice contains the same number of elements (slices don't have the same geometrical width, and for VDF meshes, slice boundaries are generally not flat except if the number of mesh elements in each direction is an exact multiple of the number of slices). First, `nx` slices in the X direction are created, then each slice is split in `ny` slices in the Y direction, and finally, each part is split in `nz` slices in the Z direction. The resulting number of parts is `nx*ny*nz`. If one particular direction has been declared periodic, the default slicing (0, 1, 2, ..., `n-1`) is replaced by (0, 1, 2, ... `n-1`, 0), each of the two '0' slices having twice less elements than the other slices.

See also: [partitionneur_deriv \(26\)](#)

Usage:

```
tranche obj Lire obj {  
    [ tranches n1 n2 (n3) ]  
    [ nb_parts int ]  
}  
where
```

- **tranches** *n1 n2 (n3)*: Partitioned by `nx` in the X direction, `ny` in the Y direction, `nz` in the Z direction. Works only for structured meshes. No warranty for unstructured meshes.
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.7 union

Description: Let several local domains be generated from a bigger one using the keyword `create_domain_from_sous_zone`, and let their partitions be generated in the usual way. Provided the list of partition files for each small domain, the keyword 'union' will partition the global domain in a conform fashion with the smaller domains.

See also: [partitionneur_deriv \(26\)](#)

Usage:

```
union liste [ nb_parts ]  
where
```

- **liste** *bloc_lecture (3.6)*: List of the partition files with the following syntaxe: {`sous_zone1 decoupage1 ... sous_zoneim decoupageim` } where `sous_zone1 ... sous_zoneim` are small domains names and `decoupage1 ... decoupageim` are partition files.
- **nb_parts** *int*: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

27 precondition_base

Description: Basic class for preconditioning.

See also: [objet_u \(36\)](#) [ssor \(27.3\)](#) [ssor_bloc \(27.4\)](#) [precondsolv \(27.2\)](#) [ilu \(27.1\)](#)

Usage:

27.1 **ilu**

Description: This preconditionner can be only used with the generic GEN solver.

See also: `precond_base` ([27](#))

Usage:

```
ilu obj Lire obj {  
    [ type int]  
    [ filling int]  
}
```

where

- **type** *int*: values can be 0|1|2|3 for null|left|right|left-and-right preconditionning (default value = 2)
- **filling** *int*: default value = 1.

27.2 **precondsolv**

Description: `not_set`

See also: `precond_base` ([27](#))

Usage:

```
precondsolv solveur  
where
```

- **solveur** *solveur_sys_base* ([10.16](#)): Solver type.

27.3 **ssor**

Description: Symmetric successive over-relaxation algorithm.

See also: `precond_base` ([27](#))

Usage:

```
ssor obj Lire obj {  
    omega float  
}
```

where

- **omega** *float*: Over-relaxation facteur (between 1 and 2, optimal value around 1.5-1.6).

27.4 **ssor_bloc**

Description: `not_set`

See also: `precond_base` ([27](#))

Usage:

```
ssor_bloc obj Lire obj {
```

```

[ alpha_0 float]
[ precond0 precond_base]
[ alpha_1 float]
[ precond1 precond_base]
[ alpha_a float]
[ preconda precond_base]
}
where

```

- **alpha_0** *float*
- **precond0** *precond_base* (27)
- **alpha_1** *float*
- **precond1** *precond_base* (27)
- **alpha_a** *float*
- **preconda** *precond_base* (27)

28 schema_temps_base

Description: Basic class for time schemes. This scheme will be associated with a problem and the equations of this problem.

See also: [objet_u \(36\)](#) [scheme_euler_explicit \(28.4\)](#) [schema_predictor_corrector \(28.19\)](#) [Sch_CN_iteratif \(28.3\)](#) [runge_kutta_ordre_3 \(28.7\)](#) [runge_kutta_ordre_4_d3p \(28.8\)](#) [leap_frog \(28.5\)](#) [runge_kutta_rationnel_ordre_2 \(28.9\)](#) [schema_implicite_base \(28.17\)](#) [schema_adams_bashforth_order_2 \(28.10\)](#) [schema_adams_bashforth_order_3 \(28.11\)](#) [schema_phase_field \(28.18\)](#) [schema_euler_explicite_ALE \(28.20\)](#)

Usage:

schema_temps_base obj Lire obj {

```

[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]

```

}
where

- **tinit** *float*: Value of initial calculation time (0 by default).
- **tmax** *float*: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float*: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float*: Minimum calculation time step (1e-16s by default).
- **dt_max** *str*: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float*: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float*: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float*: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float*: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int*
- **diffusion_implicit** *int*: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt = facsec * dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt = facsec * dt_{max}$.
- **seuil_diffusion_implicit** *float*: This keyword changes the default value (1e-6) of convergence criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int*: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int*
- **no_conv_subiteration_diffusion_implicit** *int*
- **dt_start** *dt_start* (10.9): dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int*: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int*: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int*: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int*: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** : To disable the check of the available amount of disk space during the calculation.
- **disable_progress** : To disable the writing of the .progress file.

- **disable_dt_ev** : To disable the writing of the .dt_ev file.

28.1 implicit_euler_steady_scheme

Synonymous: **schema_euler_implicite_stationnaire**

Description: This is the Implicit Euler scheme using a dual time step procedure (using local and global dt) for steady problems. Remark: the only possible solver choice for this scheme is the implicit_steady solver.

See also: **schema_implicite_base** ([28.17](#))

Usage:

```
implicit_euler_steady_scheme obj Lire obj {
    [ max_iter_implicite int]
    [ steady_security_facteur float]
    [ steady_global_dt float]
    solveur solveur_implicite_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max str]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int]
    [ no_error_if_not_converged_diffusion_implicite int]
    [ no_conv_subiteration_diffusion_implicite int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicite int]
    [ precision_impr int]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
    [ disable_progress ]
    [ disable_dt_ev ]
}
```

where

- **max_iter_implicite** *int*: Maximum number of iterations allowed for the solver (by default 200)
- **steady_security_facteur** *float*: Parameter used in the local time step calculation procedure in order to increase or decrease the local dt value (by default 0.5). We expect a strictly positive value
- **steady_global_dt** *float*: This is the global time step used in the dual time step algorithm (by default 100). We expect a strictly positive value
- **solveur** *solveur_implicite_base* ([29](#)) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *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** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for im-

plicit diffusion.

- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.2 Sch_CN_EX_iteratif

Description: This keyword also describes a Crank-Nicholson method of second order accuracy but here, for scalars, because of instabilities encountered when $dt > dt_{CFL}$, the Crank Nicholson scheme is not applied to scalar quantities. Scalars are treated according to Euler-Explicite scheme at the end of the CN treatment for velocity flow fields (by doing p Euler explicite under-iterations at $dt \leq dt_{CFL}$). Parameters are the same (but default values may change) compare to the Sch_CN_iterative scheme plus a relaxation keyword: niter_min (2 by default), niter_max (6 by default), niter_avg (3 by default), facsec_max (20 by default), seuil (0.05 by default)

See also: Sch_CN_iteratif ([28.3](#))

Usage:

Sch_CN_EX_iteratif obj Lire obj {

```
[ omega float]
[ niter_min int]
[ niter_max int]
[ niter_avg int]
[ facsec_max float]
[ seuil float]
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
```

```
[ disable_dt_ev ]
}
```

where

- **omega** *float*: relaxation factor (0.1 by default)
- **niter_min** *int* for inheritance: minimal number of p-iterations to satisfy convergence criteria (2 by default)
- **niter_max** *int* for inheritance: number of maximum p-iterations allowed to satisfy convergence criteria (6 by default)
- **niter_avg** *int* for inheritance: threshold of p-iterations (3 by default). If the number of p-iterations is greater than niter_avg, facsec is reduced, if lesser than niter_avg, facsec is increased (but limited by the facsec_max value).
- **facsec_max** *float* for inheritance: maximum ratio allowed between dynamical time step returned by iterative process and stability time returned by CFL condition (2 by default).
- **seuil** *float* for inheritance: criteria for ending iterative process ($\text{Max}(\|u(p) - u(p-1)\|/\text{Max}\|u(p)\|) < \text{seuil}$) (0.001 by default)
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt = \text{facsec} * dt_{\text{convection}}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt = \text{facsec} * dt_{\text{max}}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance

- **dt_start** *dt_start* (10.9) for inheritance: **dt_start dt_min** : the first iteration is based on **dt_min**.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on **dt_calc**.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.3 Sch_CN_iteratif

Description: The Crank-Nicholson method of second order accuracy. A mid-point rule formulation is used (Euler-centered scheme). The basic scheme is:

$$u(t + 1) = u(t) + du/dt(t + 1/2) * dt$$

The estimation of the time derivative du/dt at the level $(t+1/2)$ is obtained either by iterative process. The time derivative du/dt at the level $(t+1/2)$ is calculated iteratively with a simple under-relaxations method. Since the method is implicit, neither the *cfl* nor the fourier stability criteria must be respected. The time step is calculated in a way that the iterative procedure converges with the less iterations as possible.

Remark : for stationary or RANS calculations, no limitation can be given for time step through high value of *facsec_max* parameter (for instance : *facsec_max* 1000). In counterpart, for LES calculations, high values of *facsec_max* may engender numerical instabilities.

See also: *schema_temps_base* (28) *Sch_CN_EX_iteratif* (28.2)

Usage:

Sch_CN_iteratif obj Lire obj {

```
[ niter_min  int]
[ niter_max  int]
[ niter_avg  int]
[ facsec_max float]
[ seuil      float]
[ tinit      float]
[ tmax       float]
[ tcpumax    float]
[ dt_min     float]
[ dt_max     str]
[ dt_sauv    float]
[ dt Impr    float]
[ facsec     float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicit int]
```

```

[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
where

```

- **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).
- **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** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the **.sauv** file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the **.sauv** files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the **.out** file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the **facsec** to 0.5.
Warning: Some schemes needs a **facsec** lower than 1 (0.5 is a good start), for example **Schema_Adams_Bashforth_order_3**.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt = \text{facsec} * dt_{\text{convection}}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large **facsec** value. Start with a **facsec** value of 1 and then increase it gradually

if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt = facsec * dt_max$.

- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: dt_start dt_min : the first iteration is based on dt_min .
 dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
 dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.4 scheme_euler_explicit

Synonymous: **schema_euler_explicite**

Description: This is the Euler explicit scheme.

See also: **schema_temps_base** (28)

Usage:

```
scheme_euler_explicit obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max str]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int]
    [ no_error_if_not_converged_diffusion_implicite int]
    [ no_conv_subiteration_diffusion_implicite int]
    [ dt_start dt_start]
```

```

[ nb_pas_dt_max int]
[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
where

```

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt = facsec * dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt = facsec * dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).

- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.5 leap_frog

Description: This is the leap-frog scheme.

See also: [schema_temps_base \(28\)](#)

Usage:

```
leap_frog obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max str]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int]
    [ no_error_if_not_converged_diffusion_implicit int]
    [ no_conv_subiteration_diffusion_implicit int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicit int]
    [ precision_impr int]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
    [ disable_progress ]
    [ disable_dt_ev ]
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).

- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.6 rk3_ft

Description: Keyword for Runge Kutta time scheme for Front_Tracking calculation.

See also: `runge_kutta_ordre_3` (28.7)

Usage:

```
rk3_ft obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max str]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ facsec float]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int]  
    [ diffusion_implicit int]  
    [ seuil_diffusion_implicit float]  
    [ impr_diffusion_implicit int]  
    [ no_error_if_not_converged_diffusion_implicit int]  
    [ no_conv_subiteration_diffusion_implicit int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicit int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the `.sauv` files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the `.out` file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the **facsec** to 0.5.
Warning: Some schemes needs a **facsec** lower than 1 (0.5 is a good start), for example `Schema_Adams_Bashforth_order_3`.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance

- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_convection$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_max$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value ($1e-6$) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: $dt_start\ dt_min$: the first iteration is based on dt_min .
 $dt_start\ dt_calc$: the time step at first iteration is calculated in agreement with CFL condition.
 $dt_start\ dt_fixe$ value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps ($1e9$ by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.7 runge_kutta_ordre_3

Description: This is the Runge-Kutta scheme of third order.

See also: `schema_temps_base` (28) `rk3_ft` (28.6)

Usage:

```
runge_kutta_ordre_3 obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max str]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicit int]
    [ seuil_diffusion_implicit float]
```

```

[ impr_diffusion_implicit int]
[ no_error_if_not_converged_diffusion_implicit int]
[ no_conv_subiteration_diffusion_implicit int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
where

```

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.

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

By default, the first iteration is based on dt_calc.

- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.8 runge_kutta_ordre_4_d3p

Description: not_set

See also: schema_temps_base (28)

Usage:

runge_kutta_ordre_4_d3p obj Lire obj {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int]
[ no_error_if_not_converged_diffusion_implicit int]
[ no_conv_subiteration_diffusion_implicit int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
```

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).

- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: **dt_start dt_min** : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.9 runge_kutta_rationnel_ordre_2

Description: This is the Runge-Kutta rational scheme of second order. The method is described in the note: Wambeck - Rational Runge-Kutta methods for solving systems of ordinary differential equations, at the link: <https://link.springer.com/article/10.1007/BF02252381>. Although rational methods require more computational work than linear ones, they can have some other properties, such as a stable behaviour with explicitness, which make them preferable. The CFD application of this RRK2 scheme is described in the note: https://link.springer.com/content/pdf/10.1007%2F3-540-13917-6_112.pdf.

See also: `schema_temps_base` (28)

Usage:

```
runge_kutta_rationnel_ordre_2 obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max str]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ facsec float]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int]  
    [ diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int]  
    [ no_error_if_not_converged_diffusion_implicite int]  
    [ no_conv_subiteration_diffusion_implicite int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicite int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every `dt_sauv`, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the `.sauv` files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the `.out` file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does

not converge with an explicit time scheme is to reduce the facsec to 0.5.

Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams_Bashforth_order_3.

- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: $dt_{start} dt_{min}$: the first iteration is based on dt_{min} .
 $dt_{start} dt_{calc}$: the time step at first iteration is calculated in agreement with CFL condition.
 $dt_{start} dt_{fixe}$ value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_{calc} .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.10 schema_adams_bashforth_order_2

Description: not_set

See also: schema_temps_base (28)

Usage:

schema_adams_bashforth_order_2 obj Lire obj {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
```

```

[ dt_max  str]
[ dt_sauv  float]
[ dt_impr  float]
[ facsec  float]
[ seuil_statio  float]
[ seuil_statio_relatif_deconseille  int]
[ diffusion_implicite  int]
[ seuil_diffusion_implicite  float]
[ impr_diffusion_implicite  int]
[ no_error_if_not_converged_diffusion_implicite  int]
[ no_conv_subiteration_diffusion_implicite  int]
[ dt_start  dt_start]
[ nb_pas_dt_max  int]
[ niter_max_diffusion_implicite  int]
[ precision_impr  int]
[ periode_sauvegarde_securite_en_heures  int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
where

```

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.

- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: *dt_start dt_min* : the first iteration is based on *dt_min*.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on *dt_calc*.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.11 schema_adams_bashforth_order_3

Description: not_set

See also: schema_temps_base (28)

Usage:

```
schema_adams_bashforth_order_3 obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max str]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int]
    [ no_error_if_not_converged_diffusion_implicit int]
    [ no_conv_subiteration_diffusion_implicit int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicit int]
    [ precision_impr int]
    [ periode_sauvegarde_securite_en_heures int]
```

```

[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
where

```

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt = facsec * dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt = facsec * dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: **dt_start dt_min** : the first iteration is based on **dt_min**.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on **dt_calc**.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).

- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.12 schema_adams_moulton_order_2

Description: not_set

See also: schema_implicite_base (28.17)

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 str]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int]
    [ no_error_if_not_converged_diffusion_implicite int]
    [ no_conv_subiteration_diffusion_implicite int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicite int]
    [ precision_impr int]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
    [ disable_progress ]
    [ disable_dt_ev ]
}

```

where

- **facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.
Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.
Advice:
The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set

to facsec value then). Faster convergence has been seen and depends on the kind of calculation:

- Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
- Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
- Thermohydraulic with natural convection, facsec around 300
- Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.

- **max_iter_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (29) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *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** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every *dt_sauv*, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.

- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: *dt_start dt_min* : the first iteration is based on *dt_min*.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on *dt_calc*.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.13 schema_adams_moulton_order_3

Description: not_set

See also: schema_implicit_base (28.17)

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 str]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int]
    [ no_error_if_not_converged_diffusion_implicit int]
    [ no_conv_subiteration_diffusion_implicit int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
```

```

[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
where

```

- **facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.
Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.
Advice:
The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:
-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
-Thermohydraulic with natural convection, facsec around 300
-Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable
These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.
- **max_iter_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (29) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. solveur is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicit (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
Advice: Since the 1.6.0 version, we recommend to use first the Implicit or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicit and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicit scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.

- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: $dt_{start} dt_{min}$: the first iteration is based on dt_{min} .
 $dt_{start} dt_{calc}$: the time step at first iteration is calculated in agreement with CFL condition.
 $dt_{start} dt_{fixe}$ value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_{calc} .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.14 schema_backward_differentiation_order_2

Description: not_set

See also: schema_implicit_base (28.17)

Usage:

schema_backward_differentiation_order_2 obj Lire obj {

[**facsec_max** *float*]
[**max_iter_implicit** *int*]


```

solveur solveur_implicite_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}

```

where

- **facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by **facsec** keyword is changed during the calculation with the implicit scheme but it couldn't be higher than **facsec_max** value.
Warning: Some implicit schemes do not permit high **facsec_max**, example **Schema_Adams_Moulton_order_3** needs **facsec=facsec_max=1**.
Advice:
The calculation may start with a **facsec** specified by the user and increased by the algorithm up to the **facsec_max** limit. But the user can also choose to specify a constant **facsec** (**facsec_max** will be set to **facsec** value then). Faster convergence has been seen and depends on the kind of calculation:
-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), **facsec** between 20-30
-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), **facsec** between 90-100
-Thermohydraulic with natural convection, **facsec** around 300
-Conduction only, **facsec** can be set to a very high value (1e8) as if the scheme was unconditionally stable
These values can also be used as rule of thumb for initial **facsec** with a **facsec_max** limit higher.
- **max_iter_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (29) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. **solver** is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and

at least Simplr. Because the two first give a fastest convergence (several times) than Piso and the Simplr has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt = facsec * dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt = facsec * dt_{max}$.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: $dt_{start} dt_{min}$: the first iteration is based on dt_{min} .
 $dt_{start} dt_{calc}$: the time step at first iteration is calculated in agreement with CFL condition.
 $dt_{start} dt_{fixe}$ value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_{calc} .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.

- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.15 schema_backward_differentiation_order_3

Description: not_set

See also: `schema_implicite_base` ([28.17](#))

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 str]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
```

}

where

- **facsec_max float**: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.

Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.

Advice:

The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:

-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and

temperature (Boussinesq value beta low), facsec between 20-30

-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100

-Thermohydraulic with natural convection, facsec around 300

-Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.

- **max_iter_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (29) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solveur* is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every *dt_sauv*, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.

- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: *dt_start dt_min* : the first iteration is based on *dt_min*.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on *dt_calc*.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.16 **scheme_euler_implicit**

Synonymous: **schema_euler_implicit**

Description: This is the Euler implicit scheme.

See also: **schema_implicit_base** (28.17)

Usage:

```
scheme_euler_implicit obj Lire obj {
    [ facsec_max float]
    [ max_iter_implicit int]
    solveur solveur_implicit_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max str]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int]
    [ no_error_if_not_converged_diffusion_implicit int]
    [ no_conv_subiteration_diffusion_implicit int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
```

```

[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
where

```

- facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.
 Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.
 Advice:
 The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:
 - Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
 - Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
 - Thermohydraulic with natural convection, facsec around 300
 - Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable
 These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.
- max_iter_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- solveur** *solveur_implicit_base* (29) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. solveur is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicit (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
 Advice: Since the 1.6.0 version, we recommend to use first the Implicit or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicit and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicit scheme.
- tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- tcupmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.

- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: $dt_{start} dt_{min}$: the first iteration is based on dt_{min} .
 $dt_{start} dt_{calc}$: the time step at first iteration is calculated in agreement with CFL condition.
 $dt_{start} dt_{fixe}$ value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_{calc} .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.17 schema_implicit_base

Description: Basic class for implicit time scheme.

See also: [schema_temps_base \(28\)](#) [scheme_euler_implicit \(28.16\)](#) [schema_adams_moulton_order_2 \(28.12\)](#) [schema_adams_moulton_order_3 \(28.13\)](#) [schema_backward_differentiation_order_2 \(28.14\)](#) [schema_backward_differentiation_order_3 \(28.15\)](#) [implicit_euler_steady_scheme \(28.1\)](#)

Usage:

```
schema_implicit_base obj Lire obj {
    [ max_iter_implicit int]
```

```

solveur solveur_implicite_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}
where

```

- **max_iter_implicite** *int*: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (29): This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *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).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every *dt_sauv*, fields are saved in the *.sauv* file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the *.sauv* files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the *.out* file.

- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: $dt_{start} dt_{min}$: the first iteration is based on dt_{min} .
 $dt_{start} dt_{calc}$: the time step at first iteration is calculated in agreement with CFL condition.
 $dt_{start} dt_{fixe}$ value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_{calc} .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.18 schema_phase_field

Description: Keyword for the only available Scheme for time discretization of the Phase Field problem.

See also: `schema_temps_base` (28)

Usage:

```
schema_phase_field obj Lire obj {
    [ schema_ch schema_temps_base]
    [ schema_ns schema_temps_base]
```



```

[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ dt_impr float]
[ facsec float]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int]
[ diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int]
[ no_error_if_not_converged_diffusion_implicit int]
[ no_conv_subiteration_diffusion_implicit int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
}

```

where

- **schema_ch** *schema_temps_base* (28): Time scheme for the Cahn-Hilliard equation.
- **schema_ns** *schema_temps_base* (28): Time scheme for the Navier-Stokes equation.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important

gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_max$.

- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: $dt_start\ dt_min$: the first iteration is based on dt_min .
 $dt_start\ dt_calc$: the time step at first iteration is calculated in agreement with CFL condition.
 $dt_start\ dt_fixe$ value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.19 schema_predictor_corrector

Description: This is the predictor-corrector scheme (second order). It is more accurate and economic than MacCormack scheme. It gives best results with a second ordre convective scheme like quick, centre (VDF).

See also: [schema_temps_base](#) (28)

Usage:

```
schema_predictor_corrector obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max str]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int]
    [ no_error_if_not_converged_diffusion_implicite int]
```

```

[ no_conv_subiteration_diffusion_implicit int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]

```

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.

- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

28.20 schema_euler_explicite_ALE

Description: This is the Euler explicit scheme used for ALE problems.

See also: [schema_temps_base \(28\)](#)

Usage:

```

schema_euler_explicite_ALE obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max str]
    [ dt_sauv float]
    [ dt_impr float]
    [ facsec float]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int]
    [ diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int]
    [ no_error_if_not_converged_diffusion_implicite int]
    [ no_conv_subiteration_diffusion_implicite int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicite int]
    [ precision_impr int]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
    [ disable_progress ]
    [ disable_dt_ev ]
}
where

```

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).

- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* for inheritance
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt = facsec * dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt = facsec * dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (10.9) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.

29 solveur_implicite_base

Description: Class for solver in the situation where the time scheme is the implicit scheme. Solver allows equation diffusion and convection operators to be set as implicit terms.

See also: objet_u (36) solveur_lineaire_std (29.7) simplifier (29.6)

Usage:

29.1 implicit_steady

Description: this is the implicit solver using a dual time step. Remark: this solver can be used only with the Implicit_Euler_Steady_Scheme time scheme.

See also: implicite (29.2)

Usage:

```
implicit_steady obj Lire obj {  
    [ seuil_convergence_implicite float]  
    [ nb_corrections_max int]  
    [ seuil_convergence_solveur float]  
    [ seuil_generation_solveur float]  
    [ seuil_verification_solveur float]  
    [ seuil_test_preliminaire_solveur float]  
    [ solveur solveur_sys_base]  
    [ no_qdm ]  
    [ nb_it_max int]  
    [ controle_residu ]  
}
```

where

- **seuil_convergence_implicite** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections than nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error $\|Ax-B\|$ is lesser than vrel after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than vrel.
- **solveur** *solveur_sys_base* (10.16) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.

- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

29.2 implicite

Description: similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.

See also: [piso \(29.4\)](#) [implicite_ALE \(29.3\)](#) [implicit_steady \(29.1\)](#)

Usage:

```
implicite obj Lire obj {
    [ seuil_convergence_implicite float]
    [ nb_corrections_max int]
    [ seuil_convergence_solveur float]
    [ seuil_generation_solveur float]
    [ seuil_verification_solveur float]
    [ seuil_test_preliminaire_solveur float]
    [ solveur solveur_sys_base]
    [ no_qdm ]
    [ nb_it_max int]
    [ controle_residu ]
}
```

where

- **seuil_convergence_implicite** float for inheritance: Convergence criteria.
- **nb_corrections_max** int for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections than nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections_max is set to 21).
- **seuil_convergence_solveur** float for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** float for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than vrel).
- **seuil_verification_solveur** float for inheritance: Option to check if residual error $\|Ax-B\|$ is lesser than vrel after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** float for inheritance: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than vrel.
- **solveur** solveur_sys_base ([10.16](#)) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** int for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

29.3 **implicite_ALE**

Description: Implicite solver used for ALE problem

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

Usage:

```
implicite_ALE obj Lire obj {  
    [ seuil_convergence_implicite float]  
    [ nb_corrections_max int]  
    [ seuil_convergence_solveur float]  
    [ seuil_generation_solveur float]  
    [ seuil_verification_solveur float]  
    [ seuil_test_preliminaire_solveur float]  
    [ solveur solveur_sys_base]  
    [ no_qdm ]  
    [ nb_it_max int]  
    [ controle_residu ]  
}
```

where

- **seuil_convergence_implicite** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections than `nb_corrections_max` if the accuracy of the projection is sufficient. (By default `nb_corrections_max` is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier-Stokes equation and the scalar equations if any. This value **MUST** be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use `vrel` as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than `vrel`).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error $\|Ax-B\|$ is lesser than `vrel` after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than `vrel`.
- **solveur** *solveur_sys_base* ([10.16](#)) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the `residu` suddenly increases.

29.4 **piso**

Description: Piso (Pressure Implicit with Split Operator) - method to solve N_S.

See also: [simpler \(29.6\)](#) [implicite \(29.2\)](#) [simple \(29.5\)](#)

Usage:

```
piso obj Lire obj {
```



```

[ seuil_convergence_implicit float]
[ nb_corrections_max int]
[ seuil_convergence_solveur float]
[ seuil_generation_solveur float]
[ seuil_verification_solveur float]
[ seuil_test_preliminaire_solveur float]
[ solveur solveur_sys_base]
[ no_qdm ]
[ nb_it_max int]
[ controle_residu ]
}
where

```

- **seuil_convergence_implicit** *float*: Convergence criteria.
- **nb_corrections_max** *int*: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections than `nb_corrections_max` if the accuracy of the projection is sufficient. (By default `nb_corrections_max` is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier-Stokes equation and the scalar equations if any. This value **MUST** be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use `vrel` as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than `vrel`).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error $\|Ax-B\|$ is lesser than `vrel` after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than `vrel`.
- **solveur** *solveur_sys_base* (10.16) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the `residu` suddenly increases.

29.5 simple

Description: SIMPLE type algorithm

See also: `piso` (29.4) `solveur_u_p` (29.8)

Usage:

simple obj Lire obj {

```

[ relax_pression float]
[ seuil_convergence_implicit float]
[ nb_corrections_max int]
[ seuil_convergence_solveur float]
[ seuil_generation_solveur float]
[ seuil_verification_solveur float]
[ seuil_test_preliminaire_solveur float]

```

```

[ solveur solveur_sys_base]
[ no_qdm ]
[ nb_it_max int]
[ controle_residu ]
}
where

```

- **relax_pression** *float*: Value between 0 and 1 (by default 1), this keyword is used only by the SIMPLE algorithm for relaxing the increment of pressure.
- **seuil_convergence_implicit** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections than `nb_corrections_max` if the accuracy of the projection is sufficient. (By default `nb_corrections_max` is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier-Stokes equation and the scalar equations if any. This value **MUST** be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use `vrel` as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than `vrel`).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error $\|Ax-B\|$ is lesser than `vrel` after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than `vrel`.
- **solveur** *solveur_sys_base* (10.16) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the `residu` suddenly increases.

29.6 simplr

Description: Simplr method for incompressible systems.

See also: `solveur_implicit_base` (29) `piso` (29.4)

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_implicite** *float*: Keyword to set the value of the convergence criteria for the resolution of the implicit system build to solve either the Navier_Stokes equation (only for Simple and Simpler algorithms) or a scalar equation. It is advised to use the default value (1e6) to solve the implicit system only once by time step. This value must be decreased when a coupling between problems is considered.
- **seuil_convergence_solveur** *float*: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value **MUST** be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float*: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than vrel).
- **seuil_verification_solveur** *float*: Option to check if residual error $\|Ax-B\|$ is lesser than vrel after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float*: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than vrel.
- **solveur** *solveur_sys_base* (10.16): Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** : Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** : Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

29.7 solveur_lineaire_std

Description: not_set

See also: solveur_implicite_base (29)

Usage:

```
solveur_lineaire_std obj Lire obj {
    [ solveur solveur_sys_base ]
}
```

where

- **solveur** *solveur_sys_base* (10.16)

29.8 solveur_u_p

Description: similar to simple.

See also: simple (29.5)

Usage:

```
solveur_u_p 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* for inheritance: Value between 0 and 1 (by default 1), this keyword is used only by the SIMPLE algorithm for relaxing the increment of pressure.
- **seuil_convergence_implicit** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections than nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier-Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error $\|Ax-B\|$ is lesser than vrel after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than vrel.
- **solveur** *solveur_sys_base* (10.16) 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.

30 source_base

Description: Basic class of source terms introduced in the equation.

See also: objet_u (36) source_generique (30.21) boussinesq_temperature (30.4) boussinesq_concentration (30.3) dirac (30.8) puissance_thermique (30.17) source_qdm_lambdaup (30.24) source_th_tdivu (30.30) source_robin (30.27) source_robin_scalaire (30.28) canal_perio (30.5) source_constituant (30.19) acceleration (30.2) coriolis (30.6) source_qdm (30.23) perte_charge_singuliere (30.16) perte_charge_directionnelle (30.12) perte_charge_isotrope (30.13) perte_charge_anisotrope (30.10) perte_charge_circulaire (30.11) darcy (30.7) forchheimer (30.9) perte_charge_reguliere (30.14) trainee (30.31) flottabilite (30.20) masse_ajoutee (30.22) source_transport_k_eps (30.32) source_qdm_phase_field (30.25) source_con_phase_field (30.18) source_rayo_semi_transp (30.26)

Usage:

30.1 Source_Transport_K_Eps_anisotherme

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transport equation. By default, these constants are set to: C1_eps=1.44 C2_eps=1.92 C3_eps=1.0

See also: `source_transport_k_eps` (30.32)

Usage:

```
Source_Transport_K_Eps_anisotherme obj Lire obj {  
    [ c3_eps float]  
    [ c1_eps float]  
    [ c2_eps float]
```

```
}
```

where

- **c3_eps** float: Third constant.
- **c1_eps** float for inheritance: First constant.
- **c2_eps** float for inheritance: Second constant.

30.2 acceleration

Description: Momentum source term to take in account the forces due to rotation or translation of a non Galilean referential R' (centre 0') into the Galilean referential R (centre 0).

See also: `source_base` (30)

Usage:

```
acceleration obj Lire obj {  
    [ vitesse champ_base]  
    [ acceleration champ_base]  
    [ omega champ_base]  
    [ domegadt champ_base]  
    [ centre_rotation champ_base]  
    [ option str into ['terme_complet', 'coriolis_seul', 'entrainement_seul']]
```

```
}
```

where

- **vitesse** champ_base (16.1): Keyword for the velocity of the referential R' into the R referential ($d\mathbf{OO}'/dt$ term [m.s-1]). The velocity is mandatory when you want to print the total cinetic energy into the non-mobile Galilean referential R (see `Ec_dans_repere_fixe` keyword).
- **acceleration** champ_base (16.1): Keyword for the acceleration of the referential R' into the R referential ($d^2\mathbf{OO}'/dt^2$ term [m.s-2]). `field_base` is a time dependant field (eg: `Champ_Fonc_t`).
- **omega** champ_base (16.1): Keyword for a rotation of the referential R' into the R referential [rad.s-1]. `field_base` is a 3D time dependant field specified for example by a `Champ_Fonc_t` keyword. The `time_field` field should have 3 components even in 2D (In 2D: 0 0 omega).
- **domegadt** champ_base (16.1): Keyword to define the time derivative of the previous rotation [rad.s-2]. Should be zero if the rotation is constant. The `time_field` field should have 3 components even in 2D (In 2D: 0 0 domegadt).
- **centre_rotation** champ_base (16.1): Keyword to specify the centre of rotation (expressed in R' coordinates) of R' into R (if the domain rotates with the R' referential, the centre of rotation is $\mathbf{0}'=(0,0,0)$). 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.

30.3 boussinesq_concentration

Description: Class to describe a source term that couples the movement quantity equation and constituent transport equation with the Boussinesq hypothesis.

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

Usage:

```
boussinesq_concentration obj Lire obj {
```

```
    c0 n x1 x2 ... xn  
    [ verif_boussinesq int]
```

```
}
```

where

- **c0** *n x1 x2 ... xn*: Reference concentration field type. The only field type currently available is Champ_Uniforme (Uniform field).
- **verif_boussinesq** *int*: Keyword to check (1) or not (0) the reference concentration in comparison with the mean concentration value in the domain. It is set to 1 by default.

30.4 boussinesq_temperature

Description: Class to describe a source term that couples the movement quantity equation and energy equation with the Boussinesq hypothesis.

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

Usage:

```
boussinesq_temperature obj Lire obj {
```

```
    t0 str  
    [ verif_boussinesq int]
```

```
}
```

where

- **t0** *str*: Reference temperature value (oC or K). It can also be a time dependant function since the 1.6.6 version.
- **verif_boussinesq** *int*: Keyword to check (1) or not (0) the reference temperature in comparison with the mean temperature value in the domain. It is set to 1 by default.

30.5 canal_perio

Description: Momentum source term to maintain flow rate. The expression of the source term is:

$$S(t) = (2*(Q(0) - Q(t)) - (Q(0) - Q(t-dt)))/(coeff*dt*area)$$

Where:

coeff=damping coefficient

area=area of the periodic boundary

Q(t)=flow rate at time t

dt=time step

Three files will be created during calculation on a datafile named DataFile.data. The first file contains

the flow rate evolution. The second file is useful for resuming a calculation with the flow rate of the previous stopped calculation, and the last one contains the pressure gradient evolution:

```
-DataFile_Channel_Flow_Rate_ProblemName_BoundaryName
-DataFile_Channel_Flow_Rate_repr_ProblemName_BoundaryName
-DataFile_Pressure_Gradient_ProblemName_BoundaryName
```

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

Usage:

canal_perio obj Lire obj {

```
    bord str
    [ h float]
    [ coeff float]
    [ debit_impose float]
```

}

where

- **bord** *str*: The name of the (periodic) boundary normal to the flow direction.
- **h** *float*: Half height of the channel.
- **coeff** *float*: Damping coefficient (optional, default value is 10).
- **debit_impose** *float*: Optional option to specify the aimed flow rate $Q(0)$. If not used, $Q(0)$ is computed by the code after the projection phase, where velocity initial conditions are slightly changed to verify incompressibility.

30.6 coriolis

Description: Keyword for a Coriolis term in hydraulic equation. Warning: Only available in VDF.

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

Usage:

coriolis **omega**

where

- **omega** *str*: Value of omega.

30.7 darcy

Description: Class for calculation in a porous media with source term of Darcy $-\nu/K \cdot V$. This keyword must be used with a permeability model. For the moment there are two models : permeability constant or Ergun's law. Darcy source term is available for quasi compressible calculation. A new keyword is added for porosity (porosite).

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

Usage:

darcy **bloc**

where

- **bloc** *bloc_lecture (3.6)*: Description.

30.8 **dirac**

Description: Class to define a source term corresponding to a volume power release in the energy equation.

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

Usage:

dirac position ch

where

- **position** *n x1 x2 ... xn*
- **ch** *champ_base (16.1)*: Thermal power field type. To impose a volume power on a domain sub-area, the *Champ_Uniforme_Morceaux* (*partly_uniform_field*) type must be used.
Warning : The volume thermal power is expressed in W.m-3.

30.9 **forchheimer**

Description: Class to add the source term of Forchheimer $-C_f/\sqrt{K} \cdot V^2$ in the Navier-Stokes equations. We must precise a permeability model : constant or Ergun's law. Moreover we can give the constant C_f : by default its value is 1. Forchheimer source term is available also for quasi compressible calculation. A new keyword is added for porosity (*porosite*).

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

Usage:

forchheimer bloc

where

- **bloc** *bloc_lecture (3.6)*: Description.

30.10 **perte_charge_anisotrope**

Description: Anisotropic pressure loss.

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

Usage:

perte_charge_anisotrope obj Lire obj {

```
    lambda str
    lambda_ortho str
    diam_hydr champ_don_base
    direction champ_don_base
    [ sous_zone str ]
```

}

where

- **lambda** *str*: Function for loss coefficient which may be Reynolds dependant (Ex: $64/Re$).
- **lambda_ortho** *str*: Function for loss coefficient in transverse direction which may be Reynolds dependant (Ex: $64/Re$).
- **diam_hydr** *champ_don_base (16.5)*: Hydraulic diameter value.
- **direction** *champ_don_base (16.5)*: Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

30.11 perte_charge_circulaire

Description: New pressure loss.

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

Usage:

```
perte_charge_circulaire obj Lire obj {  
    lambda str  
    lambda_ortho str  
    diam_hydr champ_don_base  
    diam_hydr_ortho champ_don_base  
    direction champ_don_base  
    [ sous_zone str ]  
}
```

where

- **lambda** *str*: Function $f(\text{Re}_{\text{tot}}, \text{Re}_{\text{long}}, t, x, y, z)$ for loss coefficient in the longitudinal direction
- **lambda_ortho** *str*: function: Function $f(\text{Re}_{\text{tot}}, \text{Re}_{\text{ortho}}, t, x, y, z)$ for loss coefficient in transverse direction
- **diam_hydr** *champ_don_base* (16.5): Hydraulic diameter value.
- **diam_hydr_ortho** *champ_don_base* (16.5): Transverse hydraulic diameter value.
- **direction** *champ_don_base* (16.5): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

30.12 perte_charge_directionnelle

Description: Directional pressure loss.

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

Usage:

```
perte_charge_directionnelle obj Lire obj {  
    lambda str  
    diam_hydr champ_don_base  
    direction champ_don_base  
    [ sous_zone str ]  
}
```

where

- **lambda** *str*: Function for loss coefficient which may be Reynolds dependant (Ex: $64/\text{Re}$).
- **diam_hydr** *champ_don_base* (16.5): Hydraulic diameter value.
- **direction** *champ_don_base* (16.5): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

30.13 perte_charge_isotrope

Description: Isotropic pressure loss.

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

Usage:

```
perte_charge_isotrope obj Lire obj {
```

```

    lambda str
    diam_hydr champ_don_base
    [ sous_zone str]
}
where

```

- **lambda** *str*: Function for loss coefficient which may be Reynolds dependant (Ex: 64/Re).
- **diam_hydr** *champ_don_base* (16.5): Hydraulic diameter value.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

30.14 perte_charge_reguliere

Description: Source term modelling the presence of a bundle of tubes in a flow.

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

Usage:

```

perte_charge_reguliere spec zone_name
where

```

- **spec** *spec_pdcr_base* (30.15): Description of longitudinale or transversale type.
- **zone_name** *str*: Name of the sub-area occupied by the tube bundle. A Sous_Zone (Sub-area) type object called *zone_name* should have been previously created.

30.15 spec_pdcr_base

Description: Class to read the source term modelling the presence of a bundle of tubes in a flow. $C_f = A \text{Re} - B$.

See also: [objet_lecture \(35\)](#) [longitudinale \(30.15.1\)](#) [transversale \(30.15.2\)](#)

Usage:

```

spec_pdcr_base ch_a a [ ch_b ] [ b ]
where

```

- **ch_a** *str into ['a', 'cf']*: Keyword to be used to set law coefficient values for the coefficient of regular pressure losses.
- **a** *float*: Value of a law coefficient for regular pressure losses.
- **ch_b** *str into ['b']*: Keyword to be used to set law coefficient values for regular pressure losses.
- **b** *float*: Value of a law coefficient for regular pressure losses.

30.15.1 longitudinale

Description: Class to define the pressure loss in the direction of the tube bundle.

See also: [spec_pdcr_base \(30.15\)](#)

Usage:

```

longitudinale dir dd ch_a a [ ch_b ] [ b ]
where

```

- **dir** *str into ['x', 'y', 'z']*: Direction.

- **dd** *float*: Tube bundle hydraulic diameter value. This value is expressed in m.
- **ch_a** *str into ['a', 'cf']*: Keyword to be used to set law coefficient values for the coefficient of regular pressure losses.
- **a** *float*: Value of a law coefficient for regular pressure losses.
- **ch_b** *str into ['b']*: Keyword to be used to set law coefficient values for regular pressure losses.
- **b** *float*: Value of a law coefficient for regular pressure losses.

30.15.2 transversale

Description: Class to define the pressure loss in the direction perpendicular to the tube bundle.

See also: [spec_pdcr_base \(30.15\)](#)

Usage:

transversale dir dd chaine_d d ch_a a [ch_b][b]

where

- **dir** *str into ['x', 'y', 'z']*: Direction.
- **dd** *float*: Value of the tube bundle step.
- **chaine_d** *str into ['d']*: Keyword to be used to set the value of the tube external diameter.
- **d** *float*: Value of the tube external diameter.
- **ch_a** *str into ['a', 'cf']*: Keyword to be used to set law coefficient values for the coefficient of regular pressure losses.
- **a** *float*: Value of a law coefficient for regular pressure losses.
- **ch_b** *str into ['b']*: Keyword to be used to set law coefficient values for regular pressure losses.
- **b** *float*: Value of a law coefficient for regular pressure losses.

30.16 perte_charge_singuliere

Description: Source term that is used to model a pressure loss over a surface area (transition through a grid, sudden enlargement) defined by the faces of elements located on the intersection of a subzone named *subzone_name* and a X,Y, or Z plane located at X,Y or Z = location.

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

Usage:

perte_charge_singuliere dir coeff bloc_definition_surface

where

- **dir** *str into ['kx', 'ky', 'kz']*: KX, KY or KZ designate directional pressure loss coefficients for respectively X, Y or Z direction.
- **coeff** *float*: Value of friction coefficient (KX, KY, KZ).
- **bloc_definition_surface** *bloc_lecture (3.6)*: Two syntaxes are possible for the surface definition block:
For VDF and VEF: { X|Y|Z = location subzone_name }
Only for VEF: { Surface surface_name }.

30.17 puissance_thermique

Description: Class to define a source term corresponding to a volume power release in the energy equation.

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

Usage:

puissance_thermique **ch**

where

- **ch** *champ_base* (16.1): Thermal power field type. To impose a volume power on a domain sub-area, the *Champ_Uniforme_Morceaux* (*partly_uniform_field*) type must be used.
Warning : The volume thermal power is expressed in W.m-3 in 3D. It is a power per volume unit (in a porous media, it is a power per fluid volume unit).

30.18 source_con_phase_field

Description: Keyword to define the source term of the Cahn-Hilliard equation.

See also: *source_base* (30)

Usage:

source_con_phase_field obj Lire obj {

```
    temps_d_affichage  int
    alpha  float
    beta  float
    kappa  float
    kappa_variable  str into ['oui', 'non']
    moyenne_de_kappa  str
    multiplicateur_de_kappa  float
    couplage_NS_CH  str
    implicitation_CH  str into ['oui', 'non']
    gmres_non_lineaire  str into ['oui', 'non']
    seuil_cv_iterations_ptfixe  float
    seuil_residu_ptfixe  float
    seuil_residu_gmresnl  float
    dimension_espace_de_krylov  int
    nb_iterations_gmresnl  int
    residu_min_gmresnl  float
    residu_max_gmresnl  float
```

}

where

- **temps_d_affichage** *int*: Time during the characteristics of the problem are shown before calculation.
- **alpha** *float*: Internal capillary coefficient α .
- **beta** *float*: Parameter β of the model.
- **kappa** *float*: Mobility coefficient κ_0 .
- **kappa_variable** *str* into ['oui', 'non']: To define a mobility which depends on concentration C .
- **moyenne_de_kappa** *str*: To define how mobility κ is calculated on faces of the mesh according to cell-centered values (*chaîne* is *arithmetique/harmonique/geometrique*).
- **multiplicateur_de_kappa** *float*: To define the parameter of the mobility expression when mobility depends on C .
- **couplage_NS_CH** *str*: Evaluating time chosen for the term source calculation into the Navier Stokes equation (*chaîne* is $\text{mutilde}(n+1/2)/\text{mutilde}(n)$, in order to be conservative, the first choice seems better).
- **implicitation_CH** *str* into ['oui', 'non']: To define if the Cahn-Hilliard will be solved using a implicit algorithm or not.

- **gmres_non_lineaire** *str* into ['oui', 'non']: To define the algorithm to solve Cahn-Hilliard equation (oui: Newton-Krylov method, non: fixed point method).
- **seuil_cv_iterations_ptfixe** *float*: Convergence threshold (an option of the fixed point method).
- **seuil_residu_ptfixe** *float*: Threshold for the matrix inversion used in the method (an option of the fixed point method).
- **seuil_residu_gmresnl** *float*: Convergence threshold (an option of the Newton-Krylov method).
- **dimension_espace_de_krylov** *int*: Vector numbers used in the method (an option of the Newton-Krylov method).
- **nb_iterations_gmresnl** *int*: Maximal iteration (an option of the Newton-Krylov method).
- **residu_min_gmresnl** *float*: Minimal convergence threshold (an option of the Newton-Krylov method).
- **residu_max_gmresnl** *float*: Maximal convergence threshold (an option of the Newton-Krylov method).

30.19 source_constituant

Description: Keyword to specify source rates, in $[[C]/s]$, for each one of the nb constituents. $[C]$ is the concentration unit.

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

Usage:

source_constituant ch
where

- **ch** *champ_base* ([16.1](#)): Field type.

30.20 flottabilite

Description: buoyancy effect

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

Usage:

flottabilite

30.21 source_generique

Description: to define a source term depending on some discrete fields of the problem and (or) analytic expression. It is expressed by the way of a generic field usually used for post-processing.

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

Usage:

source_generique champ
where

- **champ** *champ_generique_base* ([8](#)): the source field

30.22 masse_ajoutee

Description: weight added effect

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

Usage:

masse_ajoutee

30.23 source_qdm

Description: Momentum source term in the Navier-Stokes equations.

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

Usage:

source_qdm ch

where

- **ch** *champ_base* (16.1): Field type.

30.24 source_qdm_lambdaup

Description: This source term is a dissipative term which is intended to minimise the energy associated to non-conformscales u' (responsible for spurious oscillations in some cases). The equation for these scales can be seen as: $du'/dt = -\lambda u' + \text{grad } P'$ where $-\lambda u'$ represents the dissipative term, with $\lambda = a/\Delta t$. For Crank-Nicholson temporal scheme, recommended value for a is 2.

Remark : This method requires to define a filtering operator.

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

Usage:

source_qdm_lambdaup obj Lire obj {

```
    lambda float  
    [ lambda_min float]  
    [ lambda_max float]  
    [ ubar_umprim_cible float]
```

}

where

- **lambda** *float*: value of λ
- **lambda_min** *float*: value of λ_{\min}
- **lambda_max** *float*: value of λ_{\max}
- **ubar_umprim_cible** *float*: value of $\bar{u}_{\text{umprim_cible}}$

30.25 source_qdm_phase_field

Description: Keyword to define the capillary force into the Navier Stokes equation for the Phase Field problem.

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

Usage:

```
source_qdm_phase_field obj Lire obj {  
    forme_du_terme_source int  
}
```

where

- **forme_du_terme_source** *int*: Kind of the source term (1, 2, 3 or 4).

30.26 **source_rayo_semi_transp**

Description: Radiative term source in energy equation.

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

Usage:

```
source_rayo_semi_transp
```

30.27 **source_robin**

Description: This source term should be used when a `Paroi_decalee_Robin` boundary condition is set in a hydraulic equation. The source term will be applied on the `N` specified boundaries. To post-process the values of `tauw`, `u_tau` and `Reynolds_tau` into the files `tauw_robin.dat`, `reynolds_tau_robin.dat` and `u_tau_robin.dat`, you must add a block `Traitement_particulier { canal { } }`

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

Usage:

```
source_robin bords
```

where

- **bords** *vect_nom* ([3.110](#))

30.28 **source_robin_scalaire**

Description: This source term should be used when a `Paroi_decalee_Robin` boundary condition is set in an energy equation. The source term will be applied on the `N` specified boundaries. The values `temp_wall_valueI` are the temperature specified on the `I`th boundary. The last value `dt_impr` is a printing period which is mandatory to specify in the data file but has no effect yet.

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

Usage:

```
source_robin_scalaire bords
```

where

- **bords** *listdeuxmots_sacc* ([30.29](#))

30.29 listdeuxmots_sacc

Description: List of groups of two words (without curly brackets).

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

Usage:

n object1 object2

list of *deuxmots* ([5.27](#))

30.30 source_th_tdivu

Description: This term source is dedicated for any scalar (called T) transport. Coupled with upwind (amont) or muscl scheme, this term gives for final expression of convection : $\text{div}(\mathbf{U}.T)-T.\text{div}(\mathbf{U})=\mathbf{U}.\text{grad}(T)$ This ensures, in incompressible flow when divergence free is badly resolved, to stay in a better way in the physical boundaries.

Warning: Only available in VEF discretization.

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

Usage:

source_th_tdivu

30.31 trainee

Description: drag effect

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

Usage:

trainee

30.32 source_transport_k_eps

Description: Keyword to alter the source term constants in the standard k-eps model epsilon transport equation. By default, these constants are set to: C1_eps=1.44 C2_eps=1.92

See also: source_base ([30](#)) Source_Transport_K_Eps_anisotherme ([30.1](#)) source_transport_k_eps_aniso_concen ([30.33](#)) source_transport_k_eps_aniso_therm_concen ([30.34](#))

Usage:

source_transport_k_eps obj Lire obj {

 [**c1_eps** float]

 [**c2_eps** float]

}

where

- **c1_eps** float: First constant.
- **c2_eps** float: Second constant.

30.33 source_transport_k_eps_aniso_concen

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transport equation. By default, these constants are set to: C1_eps=1.44 C2_eps=1.92 C3_eps=1.0

See also: source_transport_k_eps ([30.32](#))

Usage:

source_transport_k_eps_aniso_concen obj Lire obj {

[**c3_eps** *float*]

[**c1_eps** *float*]

[**c2_eps** *float*]

}

where

- **c3_eps** *float*: Third constant.
- **c1_eps** *float* for inheritance: First constant.
- **c2_eps** *float* for inheritance: Second constant.

30.34 source_transport_k_eps_aniso_therm_concen

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transport equation. By default, these constants are set to: C1_eps=1.44 C2_eps=1.92 C3_eps=1.0

See also: source_transport_k_eps ([30.32](#))

Usage:

source_transport_k_eps_aniso_therm_concen obj Lire obj {

[**c3_eps** *float*]

[**c1_eps** *float*]

[**c2_eps** *float*]

}

where

- **c3_eps** *float*: Third constant.
- **c1_eps** *float* for inheritance: First constant.
- **c2_eps** *float* for inheritance: Second constant.

31 sous_zone

Description: It is an object type describing a domain sub-set.

A Sous_Zone (Sub-area) type object must be associated with a Domaine type object. The Read (Lire) interpreter is used to define the items comprising the sub-area.

Caution: The Domain type object nom_domaine must have been meshed (and triangulated or tetrahedralised in VEF) prior to carrying out the Associate (Associer) nom_sous_zone nom_domaine instruction; this instruction must always be preceded by the read instruction.

See also: objet_u ([36](#))

Usage:

sous_zone obj Lire obj {

```

[ restriction str]
[ rectangle bloc_origine_cotes]
[ segment bloc_origine_cotes]
[ boite bloc_origine_cotes]
[ liste n n1 n2 ... nn]
[ fichier str]
[ intervalle deuxentiers]
[ polynomes bloc_lecture]
[ couronne bloc_couronne]
[ tube bloc_tube]
[ fonction_sous_zone str]
[ union str]
}
where

```

- **restriction** *str*: The elements of the sub-area *nom_sous_zone* must be included into the other sub-area named *nom_sous_zone2*. This keyword should be used first in the Read keyword.
- **rectangle** *bloc_origine_cotes* (31.1): The sub-area will include all the domain elements whose centre of gravity is within the Rectangle (in dimension 2).
- **segment** *bloc_origine_cotes* (31.1)
- **boite** *bloc_origine_cotes* (31.1): The sub-area will include all the domain elements whose centre of gravity is within the Box (in dimension 3).
- **liste** *n n1 n2 ... nn*: The sub-area will include *n* domain items, numbers No. 1 No. *i* No. *n*.
- **fichier** *str*: The sub-area is read into the file filename.
- **intervalle** *deuxentiers* (5.26.11): The sub-area will include domain items whose number is between *n1* and *n2* (where $n1 \leq n2$).
- **polynomes** *bloc_lecture* (3.6): A REPENDRE
- **couronne** *bloc_couronne* (31.2): In 2D case, to create a couronne.
- **tube** *bloc_tube* (31.3): In 3D case, to create a tube.
- **fonction_sous_zone** *str*: Keyword to build a sub-area with the the elements included into the area defined by *fonction*>0.
- **union** *str*: The elements of the sub-area *nom_sous_zone3* will be added to the sub-area *nom_sous_zone*. This keyword should be used last in the Read keyword.

31.1 bloc_origine_cotes

Description: Class to create a rectangle (or a box).

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

Usage:

name origin name2 cotes

where

- **name** *str* into [*'Origine'*]: Keyword to define the origin of the rectangle (or the box).
- **origin** *x1 x2 (x3)*: Coordinates of the origin of the rectangle (or the box).
- **name2** *str* into [*'Cotes'*]: Keyword to define the length along the axes.
- **cotes** *x1 x2 (x3)*: Length along the axes.

31.2 bloc_couronne

Description: Class to create a couronne (2D).

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

Usage:

name origin name3 ri name4 re

where

- **name** *str* into [*'Origine'*]: Keyword to define the center of the circle.
- **origin** *x1 x2 (x3)*: Center of the circle.
- **name3** *str* into [*'ri'*]: Keyword to define the interior radius.
- **ri** *float*: Interior radius.
- **name4** *str* into [*'re'*]: Keyword to define the exterior radius.
- **re** *float*: Exterior radius.

31.3 bloc_tube

Description: Class to create a tube (3D).

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

Usage:

name origin name2 direction name3 ri name4 re name5 h

where

- **name** *str* into [*'Origine'*]: Keyword to define the center of the tube.
- **origin** *x1 x2 (x3)*: Center of the tube.
- **name2** *str* into [*'dir'*]: Keyword to define the direction of the main axis.
- **direction** *str* into [*'X', 'Y', 'Z'*]: direction of the main axis X, Y or Z
- **name3** *str* into [*'ri'*]: Keyword to define the interior radius.
- **ri** *float*: Interior radius.
- **name4** *str* into [*'re'*]: Keyword to define the exterior radius.
- **re** *float*: Exterior radius.
- **name5** *str* into [*'hauteur'*]: Keyword to define the height of the tube.
- **h** *float*: Height of the tube.

32 turbulence_paro_base

Description: Basic class for wall laws for Navier-Stokes equations.

See also: [objet_u \(36\)](#) [loi_standard_hydr_old \(32.5\)](#) [loi_standard_hydr \(32.4\)](#) [paroi_tble \(32.8\)](#) [negligeable \(32.7\)](#) [utau_imp \(32.12\)](#) [loi_puissance_hydr \(32.3\)](#)

Usage:

32.1 loi_ciofalo_hydr

Description: A Loi_ciofalo_hydr law for wall turbulence for NAVIER STOKES equations.

See also: [loi_standard_hydr \(32.4\)](#)

Usage:

loi_ciofalo_hydr

32.2 loi_expert_hydr

Description: This keyword is similar to the previous keyword `Loi_standard_hydr` but has several additional options into brackets.

See also: `loi_standard_hydr` ([32.4](#))

Usage:

```
loi_expert_hydr obj Lire obj {  
    [ u_star_impose float]  
    [ methode_calcul_face_keps_impose str into ['toutes_les_faces_accrochees', 'que_les_faces_des-  
_elts_dirichlet']]  
    [ kappa float]  
    [ Erugu float]  
    [ A_plus float]  
}
```

where

- **u_star_impose** float: The value of the friction velocity (u^*) is not calculated but given by the user.
- **methode_calcul_face_keps_impose** str into ['toutes_les_faces_accrochees', 'que_les_faces_des-
_elts_dirichlet']: The available options select the algorithm to apply K and Eps boundaries condition (the algorithms differ according to the faces).
toutes_les_faces_accrochees : Default option in 2D (the algorithm is the same than the algorithm used in `Loi_standard_hydr`)
que_les_faces_des_elts_dirichlet : Default option in 3D (another algorithm where less faces are concerned when applying K-Eps boundary condition).
- **kappa** float: The value can be changed from the default one (0.415)
- **Erugu** float: The value of E can be changed from the default one for a smooth wall (9.11). It is also possible to change the value for one boundary wall only with `paroi_rugueuse` keyword/
- **A_plus** float: The value can be changed from the default one (26.0)

32.3 loi_puissance_hydr

Description: A `Loi_puissance_hydr` law for wall turbulence for NAVIER STOKES equations.

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

Usage:

32.4 loi_standard_hydr

Description: Keyword for the logarithmic wall law for a hydraulic problem. `Loi_standard_hydr` refers to first cell rank eddy-viscosity defined from continuous analytical functions, whereas `Loi_standard_hydr_3couches` from functions separately defined for each sub-layer

See also: `turbulence_paro_base` ([32](#)) `loi_expert_hydr` ([32.2](#)) `loi_ww_hydr` ([32.6](#)) `loi_ciofalo_hydr` ([32.1](#))

Usage:

```
loi_standard_hydr
```

32.5 loi_standard_hydr_old

Description: not_set

See also: turbulence_paro_base (32)

Usage:

loi_standard_hydr_old

32.6 loi_ww_hydr

Description: laws have been qualified on channel calculation

See also: loi_standard_hydr (32.4)

Usage:

32.7 negligeable

Description: Keyword to suppress the calculation of a law of the wall with a turbulence model. The wall stress is directly calculated with the derivative of the velocity, in the direction perpendicular to the wall ($\tau_{\text{tan}}/\rho = \nu \, dU/dy$).

Warning: This keyword is not available for k-epsilon models. In that case you must choose a wall law.

See also: turbulence_paro_base (32)

Usage:

negligeable

32.8 paroi_tble

Description: Keyword for the Thin Boundary Layer Equation wall-model (a more complete description of the model can be found into this PDF file). The wall shear stress is evaluated thanks to boundary layer equations applied in a one-dimensional fine grid in the near-wall region.

See also: turbulence_paro_base (32)

Usage:

paroi_tble obj Lire obj {

```
[ n int]
[ facteur float]
[ modele_visco str]
[ stats twofloat]
[ sonde_tble liste_sonde_tble]
[ restart ]
[ stationnaire entierfloat]
[ lambda str]
[ mu str]
[ sans_source_boussinesq ]
[ alpha float]
[ kappa float]
```

}

where

- **n** *int*: Number of nodes in the TBLE grid (mandatory option).
- **facteur** *float*: Stretching ratio for the TBLE grid (to refine, the TBLE facteur must be greater than 1).
- **modele_visco** *str*: File name containing the description of the eddy viscosity model.
- **stats** *twofloat* (32.9): Statistics of the TBLE velocity and turbulent viscosity profiles. 2 values are required : the starting time and ending time of the statistics computation.
- **sonde_tble** *liste_sonde_tble* (32.10)
- **restart**
- **stationnaire** *entierfloat* (32.11)
- **lambda** *str*
- **mu** *str*
- **sans_source_boussinesq**
- **alpha** *float*
- **kappa** *float*

32.9 twofloat

Description: two reals.

See also: *objet_lecture* (35)

Usage:

a b

where

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

32.10 liste_sonde_tble

Description: *not_set*

See also: *listobj* (34.3)

Usage:

n *object1 object2*

list of *sonde_tble* (32.10.1)

32.10.1 sonde_tble

Description: *not_set*

See also: *objet_lecture* (35)

Usage:

name point

where

- **name** *str*
- **point** *un_point* (3.15.3)

32.11 entierfloat

Description: An integer and a real.

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

Usage:

the_int the_float

where

- **the_int** *int*: Integer.
- **the_float** *float*: Real.

32.12 utau_imp

Description: Keyword to impose the friction velocity on the wall with a turbulence model for thermohydraulic problems. There are two possibilities to use this keyword :

1 - we can impose directly the value of the friction velocity u_{star} .

2 - we can also give the friction coefficient and hydraulic diameter. So, TRUST determines the friction velocity by : $u_{\text{star}} = U \cdot \sqrt{\lambda_c / 8}$.

See also: [turbulence_paro_base \(32\)](#)

Usage:

utau_imp obj Lire obj {

 [**u_tau** *champ_base*]

 [**lambda_c** *str*]

 [**diam_hydr** *champ_base*]

}

where

- **u_tau** *champ_base* (16.1): Field type.
- **lambda_c** *str*: The friction coefficient. It can be function of the spatial coordinates x,y,z, the Reynolds number Re , and the hydraulic diameter.
- **diam_hydr** *champ_base* (16.1): The hydraulic diameter.

33 turbulence_paro_scalaire_base

Description: Basic class for wall laws for energy equation.

See also: [objet_u \(36\)](#) [loi_standard_hydr_scalaire \(33.6\)](#) [loi_analytique_scalaire \(33.2\)](#) [paroi_tble_scal \(33.8\)](#) [loi_paro_nu_impose \(33.5\)](#) [negligeable_scalaire \(33.7\)](#) [loi_odvm \(33.4\)](#) [loi_WW_scalaire \(33.1\)](#)

Usage:

33.1 loi_WW_scalaire

Description: not_set

See also: [turbulence_paro_scalaire_base \(33\)](#)

Usage:

loi_WW_scalaire

33.2 loi_analytique_scalaire

Description: not_set

See also: turbulence_paro_scalaire_base (33)

Usage:

loi_analytique_scalaire

33.3 loi_expert_scalaire

Description: Keyword similar to keyword Loi_standard_hydr_scalaire but with additional option.

See also: loi_standard_hydr_scalaire (33.6)

Usage:

```
loi_expert_scalaire obj Lire obj {  
    [ prdt_sur_kappa float ]  
    [ calcul_ldp_en_flux_impose int into [0, 1] ]  
}
```

where

- **prdt_sur_kappa** *float*: This option is to change the default value of 2.12 in the scalable wall function.
- **calcul_ldp_en_flux_impose** *int into [0, 1]*: By default (value set to 0), the law of the wall is not applied for a wall with a Neumann condition. With value set to 1, the law is applied even on a wall with Neumann condition.

33.4 loi_odvm

Description: Thermal wall-function based on the simultaneous 1D resolution of a turbulent thermal boundary-layer and a variance transport equation, adapted to conjugate heat-transfer problems with fluid/solid thermal interaction (where a specific boundary condition should be used : Paroi_Echange_Contact_OVDM_VDF). This law is also available with isothermal walls.

See also: turbulence_paro_scalaire_base (33)

Usage:

```
loi_odvm obj Lire obj {  
    n int  
    gamma float  
    [ stats floatfloat ]  
    [ check_files ]  
}
```

where

- **n** *int*: Number of points per face in the 1D uniform meshes. n should be choosen in order to have the first point situated near $\Delta y=1/3$.
- **gamma** *float*: Smoothing parameter of the signal between 10e-5 (no smoothing) and 10e-1 (high averaging).

- **stats** *floatfloat* (5.28): *value_t0 value_dt* : Only for plane channel flow, it gives mean and root mean square profiles in the fine meshes, since *value_t0* and every *value_dt* seconds. The values are printed into files named *ODVM_fields*.dat*.
- **check_files** : It gives for one boundary face a historical view of local instantaneous and filtered values, as well as the calculated variance profiles from the resolution of the equation. The printed values are into the file *Suivi_ndeb.dat*.

33.5 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* (33)

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* (16.1): The hydraulic diameter.

33.6 loi_standard_hydr_scalaire

Description: Keyword for the law of the wall.

See also: *turbulence_paroι_scalaire_base* (33) *loi_expert_scalaire* (33.3)

Usage:

loi_standard_hydr_scalaire

33.7 negligeable_scalaire

Description: Keyword to suppress the calculation of a law of the wall with a turbulence model for thermohydraulic problems. The wall stress is directly calculated with the derivative of the velocity, in the direction perpendicular to the wall.

See also: *turbulence_paroι_scalaire_base* (33)

Usage:

negligeable_scalaire

33.8 paroi_tble_scal

Description: Keyword for the Thin Boundary Layer Equation thermal wall-model.

See also: *turbulence_paroι_scalaire_base* (33)

Usage:

paroi_tble_scal obj Lire obj {

```
[ n int]  
[ facteur float]  
[ modele_visco str]  
[ nb_comp int]  
[ stats fourfloat]  
[ sonde_tble liste_sonde_tble]  
[ prandtl float]
```

}

where

- **n** *int*: Number of nodes in the TBLE grid (mandatory option).
- **facteur** *float*: Stretching ratio for the TBLE grid (to refine, the TBLE facteur must be greater than 1).
- **modele_visco** *str*: File name containing the description of the eddy viscosity model.
- **nb_comp** *int*: Number of component to solve in the fine grid (1 if 2D simulation (2D not available yet), 2 if 3D simulation).
- **stats** *fourfloat* (33.9): Statistics of the TBLE velocity and turbulent viscosity profiles. 4 values are required : the starting time of velocity averaging, the starting time of the RMS fluctuations, the ending time of the statistics computation and finally the print time period for the statistics.
- **sonde_tble** *liste_sonde_tble* (32.10)
- **prandtl** *float*

33.9 fourfloat

Description: Four reals.

See also: objet_lecture (35)

Usage:

a b c d

where

- **a** *float*: First real.
- **b** *float*: Second real.
- **c** *float*: Third real.
- **d** *float*: Fourth real.

34 listobj_impl

Description: not_set

See also: objet_u (36) listobj (34.3)

Usage:

34.1 list_un_pb

Description: pour les groupes

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

Usage:

{ object1 , object2 }
list of [un_pb \(34.2\)](#) separated with ,

34.2 un_pb

Description: pour les groupes

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

Usage:

mot

where

- **mot** *str*: the string

34.3 listobj

Description: List of objects.

See also: [listobj_impl \(34\)](#) [champs_a_post \(4.2.21\)](#) [list_stat_post \(4.2.24\)](#) [listpoints \(4.2.7\)](#) [sondes \(4.2.3\)](#) [listchamp_generique \(8.3\)](#) [list_nom_virgule \(8.2\)](#) [definition_champs \(4.2.1\)](#) [post_processings \(4.3\)](#) [liste_post \(4.5\)](#) [liste_post_ok \(4.4\)](#) [condlims \(4.10.1\)](#) [sources \(5.5\)](#) [vect_nom \(3.110\)](#) [list_nom \(3.95\)](#) [list_bord \(3.56.4\)](#) [list_bloc_mailler \(3.56\)](#) [list_un_pb \(34.1\)](#) [list_list_nom \(4.8\)](#) [ecrire_fichier_xyz_valeur_param \(5.6\)](#) [pp \(5.19\)](#) [listdeuxmots_sacc \(30.29\)](#) [liste_sonde_tble \(32.10\)](#) [listeqn \(4.12\)](#) [list_info_med \(4.39\)](#) [listsous_zone_valeur \(5.2.12\)](#) [reactions \(9.1\)](#)

Usage:

35 objet_lecture

Description: Auxiliary class for reading.

See also: [objet_u \(36\)](#) [bloc_lecture \(3.6\)](#) [deuxmots \(5.27\)](#) [format_file \(4.6\)](#) [deuxentiers \(5.26.11\)](#) [floatfloat \(5.28\)](#) [entierfloat \(32.11\)](#) [champ_a_post \(4.2.22\)](#) [champs_posts \(4.2.20\)](#) [stat_post_deriv \(4.2.25\)](#) [stats_posts \(4.2.23\)](#) [stats_serie_posts \(4.2.31\)](#) [sonde_base \(4.2.5\)](#) [un_point \(3.15.3\)](#) [sonde \(4.2.4\)](#) [definition_champ \(4.2.2\)](#) [postraitement_base \(4.4.2\)](#) [un_postraitement \(4.3.1\)](#) [type_un_post \(4.5.2\)](#) [type_postraitement_ft_lata \(4.5.3\)](#) [un_postraitement_spec \(4.5.1\)](#) [nom_postraitement \(4.4.1\)](#) [condinit \(5.4.1\)](#) [condinits \(5.4\)](#) [condlimlu \(4.10.2\)](#) [mailler_base \(3.56.1\)](#) [defbord \(3.56.7\)](#) [bord_base \(3.56.5\)](#) [bloc_pave \(3.56.3\)](#) [parametre_equation_base \(5.7\)](#) [un_pb \(34.2\)](#) [bords_ecrire \(5.6.2\)](#) [ecrire_fichier_xyz_valeur_item \(5.6.1\)](#) [convection_deriv \(5.2.1\)](#) [bloc_convection \(5.2\)](#) [diffusion_deriv \(5.3.1\)](#) [op_implicite \(5.3.9\)](#) [bloc_diffusion \(5.3\)](#) [traitement_particulier_base \(5.29.1\)](#) [traitement_particulier \(5.29\)](#) [penalisation_l2_ftd_lec \(5.19.1\)](#) [dt_impr_ustar_mean_only \(5.26.1\)](#) [modele_turbulence_hyd_deriv \(5.26\)](#) [paroi_ft_disc_deriv \(12.61\)](#) [bloc_sutherland \(21.6\)](#) [form_a_nb_points \(5.26.4\)](#) [fourfloat \(33.9\)](#) [twofloat \(32.9\)](#) [sonde_tble \(32.10.1\)](#) [remove_elem_bloc \(3.83\)](#) [lecture_bloc_moment_base \(3.15\)](#) [bloc_origine_cotes \(31.1\)](#) [bloc_couronne \(31.2\)](#) [bloc_tube \(31.3\)](#) [verifiercoin_bloc \(3.113\)](#) [bloc_lecture_poro \(3.67\)](#) [bloc_lec_champ_init_canal_sinal \(16.15\)](#) [fonction_champ_reprise \(16.11\)](#) [bloc_decouper \(3.64\)](#) [troisf \(3.41\)](#) [spec_pdc_r_base \(30.15\)](#) [format_lata_to_med \(3.52\)](#) [info_med \(4.39.1\)](#) [methode_transport_deriv \(5.36\)](#) [bloc_ef \(5.2.9\)](#) [sous_zone_valeur \(5.2.13\)](#) [bloc_diffusion_standard \(5.3.7\)](#) [reaction \(9.1.1\)](#) [bloc_lecture_remaillage \(5.37\)](#) [objet_lecture_maintien_temperature \(5.21\)](#) [interpolation_champ_face_deriv \(5.39\)](#) [parcours_interface \(5.38\)](#) [injection_marqueur \(5.42\)](#) [penalisation_forcage \(5.25\)](#) [modele_fonction_bas_reynolds_base \(5.26.21\)](#) [floatentier \(5.26.12\)](#) [eq_rayo_semi_transp \(4.10\)](#) [ceg_cea_jaea \(5.29.12\)](#)

ceg_areva ([5.29.11](#))

Usage:

36 index

Index

/*, 189
#, 210

, 110, 117, 120, 164
associer, 18
champ_post_statistiques_correlation, 73, 192
champ_post_statistiques_ecart_type, 73, 193
champ_post_statistiques_moyenne, 73, 196
champ_uniforme, 244
decouper, 43, 269
discretiser, 24
divergence, 192
ecrire_fichier, 62
extraction, 193
fin, 31
gradient, 194
interpolation, 195
lire, 48
lire_fichier, 48
lire_fichier_bin, 48
lire_med, 16
morceau_equation, 195
operateur_eqn, 191
postraitement, 75
postraitements, 74
raffiner_simplexes, 47
rectify_mesh, 49
reduction_0d, 197
refchamp, 198
resoudre, 54
schema_euler_explicite, 281
schema_euler_implicite, 304
schema_euler_implicite_stationnaire, 274
tparoi_vef, 198
transformation, 199
6_points, 158, 267
≤, 37, 38
=, 37, 38
A, 214
a, 326, 327
amont, 114
analytique, 179, 181
ancien, 124–126
antisym, 112
arrete, 144–151, 153–158
avec_energie_cinetique, 132, 133
avec_les_cl, 139, 140, 170, 172–175, 177
avec_sources, 139, 140, 170, 172–175, 177
avec_sources_et_operateurs, 139, 140, 170, 172–175, 177
average, 197

b, 326, 327
binaire, 25, 70, 77, 238
bords, 121
C, 262
C_ext, 215, 217, 218
centre, 114
cf, 326, 327
chakravarthy, 114
champ_frontiere, 194
chsom, 66
composante, 199
conservation_masse, 261, 262
constant, 261, 262
coriolis_seul, 321
Cotes, 334
d, 327
debit_total, 33
default, 195
default_bar, 112, 118
dir, 335
distant, 38
divrhout_moins_Tdivrhout, 124–126
divut_moins_Tdivu, 124–126
dt_integr, 74
dt_post, 70, 72
edo, 261, 262
elem, 41, 71, 73, 235, 237
emissivite, 214
entrainement_seul, 321
euclidian_norm, 197
faces, 71, 73
family_names_from_group_names, 16, 17
filtrer_resu, 112, 119
Fluctu_Temperature_ext, 215, 217, 218
flux_bords, 195, 196
Flux_Chaleur_Turb_ext, 215, 217, 218
flux_surfacique_bords, 195, 196
fonction, 238
format_post_sup, 34
formatte, 25, 70, 77, 238
formule, 199
grad_i, 139, 140
grad_Ubar, 119
grav, 66
gravcl, 66
hauteur, 335
homogene, 38
implicite, 119
initiale, 179, 181
integrale_en_z, 33

k , 230
 K_Eps_ext , 215, 217, 218
 kx , 327
 ky , 327
 kz , 327
 L1_norm , 197
 L2_norm , 197
 last_time , 235, 237
 lata , 34, 45, 46, 64, 65, 75, 76
 lata_v1 , 34, 45, 46, 64, 65, 75, 76
 lata_v2 , 34, 45, 46, 64, 65, 75, 76
 left_value , 197
 lml , 34, 45, 46, 64, 65, 75, 76
 local , 38
 max , 197
 med , 34, 45, 46, 64, 65, 75, 76
 med_major , 64, 65, 75, 76
 min , 197
 minmod , 114
 modifiee , 179, 181
 moins_rho_moyen , 261, 262
 moy_euler , 158, 267
 moyenne , 197
 moyenne_ponderee , 197
 mu0 , 262
 muscl , 114
 nb_pas_dt_post , 70, 72
 no , 186, 195
 nodes , 66
 non , 42, 170, 328
 normalized_euclidian_norm , 197
 norme , 199
 nu , 119
 nu_transp , 119
 nut , 119
 nut_transp , 119
 one_way_coupling , 187
 Origine , 334, 335
 oui , 42, 170, 328
 periode , 66
 plans_paralleles , 158, 267
 post_processing , 77
 postraitement , 77
 postraitement_ft_lata , 77
 postraitement_lata , 77
 produit_scalaire , 199
 que_les_faces_des_elts_dirichlet , 335, 336
 re , 334, 335
 rho_g , 139, 140
 ri , 334, 335
 sans_energie_cinetique , 132, 133
 sans_rien , 139, 140, 170, 172–175, 177
 scotti , 144–151, 153–158
 short_family_names , 16, 17
 simplifiee , 179, 181
 Slambda , 262
 solveur , 119
 som , 41, 66, 71, 73, 235, 237
 somme , 197
 somme_ponderee , 197
 somme_ponderee_porosite , 197
 stabilite , 195, 196
 standard , 261, 262
 suivi , 187
 sum , 197
 superbee , 114
 T0 , 262
 T_ext , 215, 217, 218
 terme_complet , 321
 toutes_les_faces_accrochees , 335, 336
 trace , 194
 transportant_bar , 112
 transporte_bar , 112
 two_way_coupling , 187
 uniforme , 179, 181
 use_existing_domain , 235, 237
 V2_ext , 215, 217, 218
 valeur_a_elem , 179, 180
 valeur_a_gauche , 197
 valeur_normale , 252
 vanalbada , 114
 vanleer , 114
 vdf_lineaire , 179, 180
 vecteur , 199
 vef , 16, 17
 vitesse_interpoele , 187
 vitesse_parois , 230
 vitesse_particules , 187
 vitesse_tangentielle , 255
 volume , 144–151, 153–158
 volume_sans_lissage , 144–151, 153–158
 weighted_average , 197
 weighted_sum , 197
 weighted_sum_porosity , 197
 X , 37, 38, 53, 335
 x , 326, 327
 xyz , 77, 238
 Y , 37, 38, 53, 335
 y , 326, 327
 yes , 186, 195
 Z , 38, 53, 335
 z , 326, 327
 , 110, 117, 120, 164
champs , 65, 76
conditions_initiales , 110, 123–128, 130–134, 136–138, 141, 171, 173, 174, 176, 178, 179, 186, 187

conditions_limites , 79, 110, 123–128, 130–134, 136–138, 141, 171, 173, 174, 176, 178, 181, 186, 188
fichier , 46
nom_zones , 43
partitionneur , 43
postraitement , 64, 78, 80–82, 84–96, 98–105, 107, 108
postraitements , 64, 78, 80–82, 84–95, 97–104, 106, 107, 109
Read_file , 62
save_matrice , 204, 205, 210
sondes , 65, 76
1D , 167
3D , 167
a0 , 201, 202
A_plus , 336
acceleration , 321
alias , 127–129, 133
alpha , 15, 113, 328, 337
alpha_0 , 272
alpha_1 , 273
alpha_a , 273
alpha_sous_zone , 113
amont_sous_zone , 113
ampli_bruit , 239
ampli_sin , 239
approximation_de_boussinesq , 170
areva , 169
ascii , 16, 55
autre_bord , 213
autre_champ_indicatrice , 213
autre_champ_temperature , 213
autre_champ_temperature_indic0 , 213
autre_champ_temperature_indic1 , 213
autre_probleme , 213
avec_certains_bords , 28
avec_certains_bords_pour_extraire_surface , 28
avec_les_bords , 28
beta , 328
beta_co , 260, 261
beta_th , 260, 261
binaire , 23, 46
boite , 334
bord , 21, 165, 322
bords_a_decouper , 23
boundaries , 143
boundary_conditions , 79, 110, 123–128, 130–134, 136–138, 141, 171, 173, 174, 176, 178, 181, 186, 188
boundary_xmax , 40
boundary_xmin , 40
boundary_ymax , 40
boundary_ymin , 40
boundary_zmax , 40
boundary_zmin , 40
btd , 116
c , 169
c0 , 321
c1_eps , 320, 332, 333
c2_eps , 320, 332, 333
c3_eps , 320, 332, 333
calc_spectre , 166, 167
calcul_ldp_en_flux_impose , 340
canal , 151
canalx , 149
cea_jaea , 169
centre_rotation , 321
chaleur_latente , 260
champ_med , 33
changement_de_base_p1bulle , 234
check_files , 340
cl_pression_sommet_faible , 234
clipping_courbure_interface , 140
cmu , 160
coef , 258
coeff , 322
coefficient_diffusion , 259
coefficients_activites , 201
collisions , 180
compo , 196
condition_elements , 27, 28
condition_faces , 28
condition_geometrique , 23
conduction , 81
conduction_milieu_variable , 82
conservation_Ec , 167
constante_modele_micro_melange , 200
constante_taux_reaction , 201
contre_energie_activation , 201
contre_reaction , 201
contribution_one_way , 187
controle_residu , 205, 314–318, 320
convection , 110, 124–130, 132–134, 136–138, 141, 171, 173, 174, 176, 178, 181, 186, 187
convection_diffusion_chaleur_qc , 99, 100
convection_diffusion_chaleur_turbulent_qc , 103, 104
convection_diffusion_concentration , 86, 87, 94, 95
convection_diffusion_concentration_turbulent , 88, 89, 96, 98
convection_diffusion_phase_field , 91
convection_diffusion_temperature , 93–95, 101
convection_diffusion_temperature_turbulent , 96, 98, 102, 105
correction_fraction , 257
correction_parcours_thomas , 185

correction_visco_turb_pour_controle_pas_de_temps , 142, 144, 146–150, 152–159, 161, 163
 correction_visco_turb_pour_controle_pas_de_temps_parametre , 142, 144, 146–148, 150–157, 159, 161, 163
 corriger_partition , 268
 couplage_NS_CH , 328
 couronne , 334
 Cp , 257
 cp , 222, 223, 235, 257, 259–263
 crank , 121
 critere_absolu , 30
 critere_arete , 184
 critere_longueur_fixe , 184
 critere_remaillage , 184
 cs , 146
 Cv , 258
 cw , 145
 d , 243, 246
 debit , 222, 223
 debit_impose , 322
 debug , 169
 debut_stat , 165
 definition_champs , 64, 75
 delta , 221
 derivee_rotation , 258
 dh , 222, 223
 diag , 205
 diam_hydr , 324, 325, 339, 341
 diam_hydr_ortho , 325
 diffusion , 110, 123–130, 132–134, 136–138, 141, 171, 173, 174, 176, 178, 181, 186, 187
 diffusion_implicite , 274, 276, 278, 280, 282, 284, 285, 287, 289, 290, 292, 294, 296, 298, 301, 303, 305, 307, 309, 311, 313
 dim_espace_krilov , 205
 dimension_espace_de_krylov , 328
 dir , 222, 223
 dir_flow , 239
 dir_wall , 240
 direction , 21, 29–31, 165, 324, 325
 disable_dt_ev , 274, 276, 279, 281, 282, 284, 286, 287, 289, 291, 293, 294, 297, 299, 301, 304, 306, 308, 310, 311, 313
 disable_progress , 274, 276, 279, 281, 282, 284, 286, 287, 289, 291, 293, 294, 297, 299, 301, 304, 306, 308, 310, 311, 313
 distance_projete_faces , 181
 dmax , 149
 domain , 40
 domaine , 21, 22, 27–31, 46, 65, 76, 194, 195, 269
 domaine_final , 21, 29
 domaine_flottant_fluide , 142
 domaine_grossier , 22
 domaine_init , 21, 29
 domaines , 46
 omega , 321
 dt_impr , 143, 222, 223, 274, 276, 278, 280, 282, 283, 285, 287, 288, 290, 292, 293, 296, 298, 300, 303, 305, 307, 309, 311, 312
 dt_impr_moy_spat , 165
 dt_impr_moy_temp , 165
 dt_impr_nusselt , 265–267
 dt_impr_ustar , 142, 144, 146–148, 150–155, 157–161, 163
 dt_impr_ustar_mean_only , 142, 144, 146–148, 150–154, 156–161, 163
 dt_injection , 188
 dt_max , 273, 276, 278, 280, 282, 283, 285, 287, 288, 290, 292, 293, 296, 298, 300, 303, 305, 307, 309, 310, 312
 dt_min , 273, 276, 278, 280, 282, 283, 285, 287, 288, 290, 292, 293, 296, 298, 300, 303, 305, 307, 309, 310, 312
 dt_post , 169
 dt_projection , 141, 170, 172, 174, 176, 177
 dt_sauv , 273, 276, 278, 280, 282, 283, 285, 287, 288, 290, 292, 293, 296, 298, 300, 303, 305, 307, 309, 311, 312
 dt_start , 274, 276, 278, 280, 282, 284, 285, 287, 289, 291, 292, 294, 296, 299, 301, 303, 306, 308, 309, 311, 313
 dt_uniforme , 189
 dtol_fraction , 257
 Ec , 166
 Ec_dans_repere_fixe , 166
 ecrire_decoupage , 43
 ecrire_fichier_xyz_valeur , 110, 123–128, 130–134, 136–138, 141, 171, 173, 174, 176, 178, 181, 186, 188
 ecrire_fichier_xyz_valeur_bin , 110, 123–127, 129–134, 136–138, 141, 171, 173, 174, 176, 178, 182, 186, 188
 ecrire_frontiere , 46
 ecrire_lata , 44
 emissivite_pour_rayonnement_entre_deux_plaques-quasi_infinies , 224
 energie_activation , 201
 ensemble_points , 188
 enthalpie_reaction , 201
 epaisseur , 28, 30
 eps_max , 159, 160, 163
 eps_min , 159, 160, 163
 eq_rayo_semi_transp , 78
 equation_frequence_resolue , 122
 equation_interface , 128, 135
 equation_interfaces_proprietes_fluide , 139
 equation_interfaces_vitesse_imposee , 139

equation_navier_stokes , 136
 equation_non_resolue , 110, 122–127, 129–134,
 136, 138, 139, 141, 171, 173, 175, 176,
 178, 182, 186, 188
 equation_temperature_mpoint , 140
 equation_temperature_mpoint_vapeur , 140
 equations_interfaces_vitesse_imposee , 140
 equations_scalaires_passifs , 80, 87, 89, 95, 98,
 100, 101, 104, 105
 Erugu , 336
 erugu , 230
 espece , 130, 131
 espece_en_competition_micro_melange , 200
 expert_only , 62
 exposant_beta , 201
 expression , 199
 facon_init , 166, 167
 facsec , 274, 276, 278, 280, 282, 283, 285, 287,
 288, 290, 292, 294, 296, 298, 300, 303,
 305, 307, 309, 311, 312
 facsec_max , 277, 280, 295, 297, 300, 302, 304
 facteur , 116, 337, 341
 facteur_longueur_ideale , 184
 facteurs , 36
 fichier , 65, 76, 149, 268, 270, 334
 fichier_distance_paroie , 162
 fichier_ecriture_K_Eps , 149
 fichier_matrice , 55
 fichier_post , 21
 fichier_secmem , 55
 fichier_solution , 55
 fichier_solveur , 55
 fichier_solveur_non_recree , 205
 fichier_sortie , 33
 fichier_ssz , 270
 fields , 65, 76
 file , 46
 file_coord_x , 40
 file_coord_y , 40
 file_coord_z , 40
 filling , 272
 fin_stat , 165
 flow_rate , 256
 fluide0 , 260
 fluide1 , 260
 fonction , 51, 148
 fonction_filtre , 41
 fonction_sous_zone , 334
 force , 204
 format , 46, 65, 76
 format_post , 41
 formate , 44
 forme_du_terme_source , 330
 formulation_a_nb_points , 144–146, 148–151, 153–
 158
 formule_mu , 260
 frequence_recalc , 205
 frontiere , 169
 function_coord_x , 40
 function_coord_y , 40
 function_coord_z , 40
 gamma , 258, 340
 genere_fichier_solveur , 55
 ghost_thickness , 40
 gmres_non_lineaire , 328
 gravite , 170
 groupes , 78, 83, 108
 h , 239, 322
 haspi , 169
 hexa_old , 29
 ignore_check_fraction , 257
 implicitation_CH , 328
 implicite , 187
 impr , 55, 184, 202, 204, 205, 210
 impr_diffusion_implicite , 274, 276, 278, 280, 282,
 284, 285, 287, 289, 291, 292, 294, 296,
 299, 301, 303, 306, 308, 309, 311, 313
 indic_faces_modifiee , 181
 indice , 260–262
 info , 118
 init_Ec , 166, 167
 initial_conditions , 110, 123–128, 130–134, 136–
 138, 141, 171, 173, 174, 176, 178, 179,
 186, 187
 initial_value , 240, 241, 247
 injecteur_interfaces , 181
 injection , 187
 interfaces , 65, 76
 interpolation_champ_face , 181
 interpolation_repere_local , 181
 intervalle , 334
 inverse_condition_element , 28
 iterations_correction_volume , 180
 joints_non_postraites , 46
 k , 261
 k_min , 159, 160, 163
 kappa , 260–262, 328, 336, 337
 kappa_variable , 328
 kmetis , 269
 lambda , 222, 223, 259–263, 324, 325, 330, 337
 lambda_c , 339
 lambda_max , 330
 lambda_min , 330
 lambda_ortho , 324, 325
 larg_joint , 43
 Lire_fichier , 62
 lissage_courbure_coeff , 184

lissage_courbure_iterations , 184
 lissage_courbure_iterations_si_remaillage , 184
 lissage_courbure_iterations_systematique , 184
 liste , 51, 334
 liste_cas , 26
 liste_de_postraitements , 64, 78, 80–82, 84–95, 97–104, 106, 107, 109
 liste_postraitements , 64, 79–82, 84–95, 97–104, 106, 107, 109
 localisation , 41, 195, 199
 loi_etat , 262
 longueur_boite , 167
 longueur_maille , 144, 145, 147–151, 153–158
 longueurs , 36
 maillage , 180
 main , 44
 maintien_temperature , 136
 masse_molaire , 127–129, 133, 235
 matrice_pression_invariante , 140
 max_iter_implicit , 275, 295, 298, 300, 302, 305, 307
 methode , 33, 194, 195, 197, 199
 methode_calcul_face_keps_impose , 336
 methode_calcul_pression_initiale , 140, 170, 172, 174, 175, 177
 methode_couplage , 187
 methode_interpolation_v , 180
 methode_transport , 180, 187
 min_critere_q_sur_max_critere_q , 169
 min_dir_flow , 240
 min_dir_wall , 240
 mode_calcul_convection , 125, 126
 modele_fonc_bas_reynolds , 160
 modele_fonc_realisable , 163
 modele_micro_melange , 200
 modele_turbulence , 126, 129, 131, 137, 140, 175, 177
 modele_visco , 337, 341
 modif_div_face_dirichlet , 234
 moyenne_convergee , 196
 moyenne_de_kappa , 328
 mpoint_inactif_sur_qdm , 140
 mpoint_vapeur_inactif_sur_qdm , 140
 mu , 222, 223, 235, 260–262, 337
 mu_1 , 133
 mu_2 , 133
 multiplicateur_de_kappa , 328
 n , 223, 261, 337, 340, 341
 n_iterations_distance , 180
 n_iterations_interpolation_ibc , 181
 name_of_initial_zones , 16
 name_of_new_zones , 16
 navier_stokes_phase_field , 91
 navier_stokes_qc , 99, 100
 navier_stokes_standard , 84, 86, 87, 93–95, 101
 navier_stokes_standard_ALE , 85
 navier_stokes_turbulent , 88–90, 96, 97, 102, 105
 navier_stokes_turbulent_qc , 103, 104
 nb_comp , 240, 241, 247, 341
 nb_corrections_max , 314–317, 319
 nb_it_max , 204, 205, 210, 314–318, 320
 nb_iter_barycentrage , 183
 nb_iter_correction_volume , 184
 nb_iter_remaillage , 183
 nb_iteration_max_uzawa , 181
 nb_iterations , 187
 nb_iterations_gmresnl , 328
 nb_mailles_mini , 169
 nb_nodes , 40
 nb_parts , 268–271
 nb_parts_geom , 22
 nb_parts_naif , 22
 nb_parts_tot , 44
 nb_pas_dt_max , 274, 276, 278, 281, 282, 284, 286, 287, 289, 291, 292, 294, 296, 299, 301, 303, 306, 308, 309, 311, 313
 nb_points , 158, 267
 nb_points_par_phase , 165
 nb_procs , 26
 nb_test , 55
 nb_tranche , 33
 nb_tranches , 29–31
 nb_var , 148
 new_jacobian , 118
 niter_avg , 277, 280
 niter_max , 277, 280
 niter_max_diffusion_implicit , 122, 274, 276, 278, 281, 282, 284, 286, 287, 289, 291, 292, 294, 296, 299, 301, 303, 306, 308, 309, 311, 313
 niter_min , 277, 280
 no_check_disk_space , 274, 276, 278, 281, 282, 284, 286, 287, 289, 291, 293, 294, 297, 299, 301, 303, 306, 308, 310, 311, 313
 no_conv_subiteration_diffusion_implicit , 274, 276, 278, 280, 282, 284, 285, 287, 289, 291, 292, 294, 296, 299, 301, 303, 306, 308, 309, 311, 313
 no_error_if_not_converged_diffusion_implicit , 274, 276, 278, 280, 282, 284, 285, 287, 289, 291, 292, 294, 296, 299, 301, 303, 306, 308, 309, 311, 313
 no_qdm , 314–318, 320
 nom , 240, 241, 247
 nom_bord , 29, 30
 nom_cl_derriere , 31
 nom_cl_devant , 31
 nom_domaine , 41

nom_fichier_post , 41
 nom_fichier_solveur , 205
 nom_fichier_sortie , 23
 nom_frontiere , 194
 nom_inconnue , 127–129, 133
 nom_mon_indicatrice , 213
 nom_pb , 41
 nom_source , 190–196, 198, 199
 nombre_de_noeuds , 36
 nombre_facettes_retenues_par_cellule , 181
 noms_champs , 41
 normal_value , 246
 normalise , 169
 nu , 118, 222, 223
 nu_transp , 118
 numero , 196, 199
 numero_op , 191
 numero_source , 191
 nusselt , 341
 nut , 118
 nut_max , 142, 145–148, 150–152, 154–161, 164
 nut_transp , 118
 old , 113
 omega , 239, 272, 277, 321
 omega_relaxation_drho_dt , 262
 optimisation_sous_maillage , 195
 optimized , 204, 210
 option , 128, 196, 321
 Origine , 36
 origine , 28
 p0 , 234
 p1 , 234
 p_imposee_aux_faces , 42
 pa , 234
 par_sous_zone , 21
 parametre_equation , 110, 123–127, 129–134, 136–138, 141, 171, 173, 175, 176, 178, 182, 186, 188
 parcours_interface , 181
 Partition_tool , 43
 pas , 183
 pas_de_solution_initiale , 55
 pas_lissage , 183
 pb_champ , 197, 198
 pb_name , 44
 penalisation_forcage , 140
 penalisation_l2_ftd , 134, 136
 perio_x , 40
 perio_y , 40
 perio_z , 40
 periode , 166
 periode_calc_spectre , 167
 periode_sauvegarde_securite_en_heures , 274, 276, 278, 281, 282, 284, 286, 287, 289, 291, 292, 294, 296, 299, 301, 303, 306, 308, 310, 311, 313
 periodique , 44
 phase , 128, 135, 213
 phase_marquee , 187
 point1 , 28
 point2 , 28
 point3 , 28
 polynomes , 334
 position , 258
 Post_processing , 64, 78, 80–82, 84–96, 98–105, 107, 108
 Post_processings , 64, 78, 80–82, 84–95, 97–104, 106, 107, 109
 potentiel_chimique_generalise , 133
 prandtl_turbulent_fonction_nu_t_alpha , 266
 Prandtl , 258
 prandtl , 257, 342
 prandtl_eps , 160, 163
 prandtl_k , 160, 163
 prdt , 266
 prdt_sur_kappa , 340
 precision_impr , 274, 276, 278, 281, 282, 284, 286, 287, 289, 291, 292, 294, 296, 299, 301, 303, 306, 308, 310, 311, 313
 precondition , 203, 204, 210
 precondition0 , 272
 precondition1 , 273
 precondition_nul , 203, 210
 preconda , 273
 preconditionnement_diag , 121
 pression , 262
 pression_reference , 142
 Probes , 65, 76
 probleme , 27, 28, 240, 241, 247
 produits , 201
 projection_initiale , 140, 170, 172, 174, 176, 177
 projection_normale_bord , 30
 proprietes_particules , 188
 pulsation_w , 165
 quiet , 159, 161, 163, 202, 204, 205, 210
 reactifs , 201
 reactions , 200
 rectangle , 334
 relax_barycentrage , 183
 relax_pression , 317, 319
 remaillage , 180
 reorder , 44
 reprise , 64, 79, 81–94, 96–103, 105–107, 109, 165
 reprise_correlation , 223, 224
 residu_max_gmresnl , 328
 residu_min_gmresnl , 328
 resolution_explicite , 122
 restart , 337

restriction , 333
resume_last_time , 64, 79, 81–90, 92–101, 103–106, 108, 109
reynolds_stress_isotrope , 162
rho , 222, 223, 259–263
rho_1 , 133
rho_2 , 133
rho_constant_pour_debug , 258
rotation , 258
rt , 234
sans_passer_par_le2d , 29
sans_solveur_masse , 191
sans_source_boussinesq , 337
sauvegarde , 64, 79, 80, 82–94, 96–104, 106, 107, 109
sauvegarde_simple , 64, 79, 81–94, 96–103, 105–107, 109
save_matrix , 204, 205, 210
sc , 257
schema_ch , 309
schema_ns , 309
scturb , 266
segment , 334
seuil , 203–205, 210, 278, 280
seuil_convergence_implicite , 122, 314–319
seuil_convergence_solveur , 122, 314–319
seuil_convergence_uzawa , 181
seuil_cv_iterations_ptfixe , 328
seuil_diffusion_implicite , 122, 274, 276, 278, 280, 282, 284, 285, 287, 289, 290, 292, 294, 296, 298, 301, 303, 305, 307, 309, 311, 313
seuil_divU , 141, 171, 172, 174, 176, 178
seuil_dvolume_residuel , 184
seuil_generation_solveur , 314–319
seuil_residu_gmresnl , 328
seuil_residu_ptfixe , 328
seuil_statio , 274, 276, 278, 280, 282, 283, 285, 287, 288, 290, 292, 294, 296, 298, 301, 303, 305, 307, 309, 311, 312
seuil_statio_relatif_deconseille , 274, 276, 278, 280, 282, 284, 285, 287, 289, 290, 292, 294, 296, 298, 301, 303, 305, 307, 309, 311, 313
seuil_test_preliminaire_solveur , 314–319
seuil_verification , 55
seuil_verification_solveur , 314–319
sigma , 260
solv_elem , 204
solveur , 55, 79, 122, 275, 295, 298, 300, 302, 305, 307, 314–320
solveur0 , 203
solveur1 , 203
solveur_bar , 141, 170, 172, 174, 176, 177
solveur_pression , 140, 170, 172, 174, 176, 177
sonde_tble , 337, 342
source , 190–196, 198, 199
source_reference , 190–196, 198, 199
sources , 110, 123–128, 130–134, 136–138, 141, 171, 173, 174, 176, 178, 181, 186, 188, 190–196, 198, 199
sources_reference , 190–196, 198, 199
sous_zone , 27, 240, 241, 247, 324, 325
sous_zones , 270
splitting , 40
stabilise , 158, 267
standard , 118
stationnaire , 337
statistiques , 65, 76
statistiques_enSerie , 65, 76
stats , 337, 340, 341
steady_global_dt , 275
steady_security_facteur , 275
stencil_width , 136
surface , 223
surfacique , 45
sutherland , 262
symx , 36
symy , 36
symz , 36
t0 , 322
t_deb , 169, 192, 193, 196
t_debut_injection , 188
t_fin , 169, 192, 193, 196
tcpumax , 273, 275, 278, 280, 281, 283, 285, 287, 288, 290, 292, 293, 296, 298, 300, 303, 305, 307, 309, 310, 312
tdivu , 113
temps_d_affichage , 328
temps_debut_prise_en_compte_drho_dt , 262
terme_gravite , 140
test , 113
thi , 151
tinf , 222, 223
tinit , 273, 275, 278, 280, 281, 283, 285, 286, 288, 290, 292, 293, 296, 298, 300, 303, 305, 307, 309, 310, 312
tmax , 273, 275, 278, 280, 281, 283, 285, 287, 288, 290, 292, 293, 296, 298, 300, 303, 305, 307, 309, 310, 312
traitement_coins , 42
traitement_particulier , 141, 171, 172, 174, 176, 178
traitement_pth , 262
traitement_rho_gravite , 262
tranches , 271
transformation_bulles , 187
transport_k_epsilon , 160

transport_k_epsilon_realisable , 163
 triangle , 28
 trois_tetra , 29
 tsup , 222, 223
 tube , 334
 turbulence_paroï , 142, 144, 146–148, 150–155, 157–161, 163, 265–267
 tuyauz , 149
 type , 196, 272
 type_vitesse_imposee , 181
 u , 243, 246
 u_star_impose , 335
 u_tau , 339
 ubar_umprim_cible , 330
 ucent , 239
 union , 334
 use_weights , 269
 val_Ec , 166, 167
 velocity_profil , 256
 verif_boussinesq , 321, 322
 verif_dparoi , 149
 via_extraire_surface , 28
 vingt_tetra , 29
 viscosite_dynamique_constante , 170
 vitesse , 258, 321
 vitesse_fluide_explicite , 185
 vitesse_imposee_regularisee , 181
 volume , 222
 volume_impose_phase_1 , 181
 volumes_etendus , 113
 volumes_non_etendus , 113
 volumique , 45
 with_nu , 186
 xinf , 223
 xsup , 223
 xtanh , 36
 xtanh_dilatation , 36
 xtanh_taille_premiere_maille , 36
 ytanh , 36
 ytanh_dilatation , 36
 ytanh_taille_premiere_maille , 36
 zmax , 33
 zmin , 33
 zones_name , 43
 ztanh , 36
 ztanh_dilatation , 36
 ztanh_taille_premiere_maille , 36
 acceleration, 320
 ale, 115
 algo_base, 188
 algo_couple_1, 189
 amort, 111
 amort_old, 111
 analyse_angle, 17
 associer, 18
 associer_algo, 18
 associer_pbm_g_pbf, 18
 associer_pbm_g_pbgglobal, 19
 axi, 19
 base, 185
 bidim_axi, 19
 bord, 37
 bord_base, 37
 boundary_field_inward, 246
 boundary_field_uniform_keps_from_ud, 246
 boussinesq_concentration, 321
 boussinesq_temperature, 321
 brech, 168
 btd, 116
 calcul, 20
 calculer_moments, 19
 canal, 165
 canal_perio, 322
 ceg, 168
 centre, 111
 centre4, 111
 centre_de_gravite, 20
 centre_old, 111
 ch_front_input, 246
 Ch_front_input_ALE, 245
 ch_front_input_uniforme, 247
 champ_base, 235
 champ_don_base, 236
 champ_don_lu, 236
 champ_fonc_fonction, 236
 champ_fonc_fonction_txyz, 237
 champ_fonc_med, 237
 Champ_Fonc_MED_Tabule, 235
 Champ_Fonc_MEDfile, 235
 champ_fonc_reprise, 237
 champ_fonc_t, 238
 champ_fonc_tabule, 238
 champ_fonc_txyz, 243
 champ_fonc_xyz, 243
 Champ_front_ale, 245
 champ_front_base, 245
 champ_front_bruite, 248
 champ_front_calc, 248
 champ_front_contact_rayo_semi_transp_vef, 248
 champ_front_contact_rayo_transp_vef, 249
 champ_front_contact_vef, 249
 champ_front_debit, 249
 champ_front_debit_massique, 250
 Champ_front_debit_QC_VDF, 245
 champ_front_fonc_pois_ipsn, 250

champ_front_fonc_pois_tube, 250
 champ_front_fonc_t, 251
 champ_front_fonc_txyz, 251
 champ_front_fonc_xyz, 251
 champ_front_fonction, 251
 champ_front_lu, 252
 champ_front_MED, 247
 champ_front_normal_vef, 252
 champ_front_pression_from_u, 252
 champ_front_recyclage, 252
 champ_front_tabule, 254
 champ_front_tangentiel_vef, 255
 champ_front_uniforme, 255
 champ_front_vortex, 255
 champ_front_xyz_debit, 256
 champ_front_zoom, 256
 champ_generique_base, 189
 champ_init_canal_sinal, 239
 champ_input_base, 240
 champ_input_p0, 240
 champ_ostwald, 241
 champ_post_de_champs_post, 189
 champ_post_extraction, 193
 champ_post_interpolation, 194
 champ_post_morceau_equation, 195
 champ_post_operateur_base, 190
 champ_post_operateur_divergence, 192
 champ_post_operateur_eqn, 191
 champ_post_operateur_gradient, 194
 champ_post_reduction_0d, 197
 champ_post_refchamp, 198
 champ_post_statistiques_base, 191
 champ_post_tparoi_vef, 198
 champ_post_transformation, 199
 champ_som_lu_vdf, 241
 champ_som_lu_vef, 241
 Champ_Tabule_Morceaux, 235
 champ_tabule_temps, 242
 champ_uniforme_morceaux, 242
 champ_uniforme_morceaux_tabule_temps, 242
 Champ_front_fonc_txyz, 13
 chimie, 200
 chmoy_faceperio, 167
 Cholesky, 206–208
 cholesky, 202
 circle, 68
 circle_3, 69
 class_generic, 201
 combinaison, 147
 Concentration, 71, 74
 condlim_base, 211
 condlims, 79
 conduction, 122
 conduction_milieu_variable, 123
 constant, 229
 constituant, 259
 contact_vdf_vef, 211
 contact_vef_vdf, 212
 convection_deriv, 110
 convection_diffusion_chaleur_qc, 124
 convection_diffusion_chaleur_turbulent_qc, 125
 convection_diffusion_concentration, 126
 convection_diffusion_concentration_ft_disc, 127
 convection_diffusion_concentration_turbulent, 129
 convection_diffusion_fraction_massique_qc, 130
 convection_diffusion_fraction_massique_turbulent_qc, 131
 convection_diffusion_phase_field, 132
 convection_diffusion_temperature, 133
 convection_diffusion_temperature_ft_disc, 135
 convection_diffusion_temperature_turbulent, 137
 coriolis, 323
 Correlation, 71
 correlation, 73, 192
 corriger_frontiere_periodique, 20
 create_domain_from_sous_zone, 21
 darcy, 323
 debug, 21
 decoupebord_pour_rayonnement, 22
 decouper_bord_coincident, 23
 di_l2, 111
 diffusion_deriv, 117
 dilate, 23
 dimension, 23
 dirac, 323
 dirichlet, 212
 disable_TU, 23
 discretisation_base, 233
 discretiser_domaine, 24
 discretize, 24
 distance_parois, 24
 domain, 39
 domaine, 234
 domaine_ale, 234
 dt_calc, 202
 dt_fixe, 202
 dt_min, 202
 dt_start, 203
 Dt_post, 71
 EASM_Baglietto, 162
 ec, 165
 ecart_type, 73, 193
 Ecart_type, 71, 74
 echange_contact_rayo_transp_vdf, 212
 echange_contact_vdf_ft_disc, 212
 echange_contact_vdf_ft_disc_solid, 213

ecrire, 62
 ecrire_champ_med, 25
 ecrire_fichier_bin, 62
 ecrire_fichier_formatte, 25
 ecrire_med, 62
 ecrire_medfile, 63
 ecriturelecturespecial, 25
 ef, 112, 233
 ef_stab, 113
 end, 31
 entree_temperature_imposee_h, 213
 epsilon, 39
 eqn_base, 138
 execute_parallel, 25
 export, 26
 extract_2d_from_3d, 26
 extract_2daxi_from_3d, 26
 extraire_domaine, 26
 extraire_plan, 27
 extraire_surface, 28
 extrudebord, 28
 extrudeparoi, 29
 extruder, 30
 extruder_en20, 30
 extruder_en3, 31

 fichier_decoupage, 268
 field_uniform_keps_from_ud, 243
 flottabilite, 329
 fluide_diphasique, 259
 fluide_incompressible, 260
 fluide_ostwald, 260
 fluide_quasi_compressible, 261
 flux_radiatif, 214
 flux_radiatif_vdf, 214
 flux_radiatif_vef, 214
 forchheimer, 323
 frontiere_ouverte, 214
 frontiere_ouverte_concentration_imposee, 215
 frontiere_ouverte_fraction_massique_imposee, 215
 frontiere_ouverte_gradient_pression_impose, 215
 frontiere_ouverte_gradient_pression_impose_vefprep1b, 215
 frontiere_ouverte_gradient_pression_libre_vef, 216
 frontiere_ouverte_gradient_pression_libre_vefprep1b, 216
 frontiere_ouverte_k_eps_impose, 216
 frontiere_ouverte_pression_imposee, 216
 frontiere_ouverte_pression_imposee_orlansky, 217
 frontiere_ouverte_pression_moyenne_imposee, 217
 frontiere_ouverte_rayo_semi_transp, 217
 frontiere_ouverte_rayo_transp, 217
 frontiere_ouverte_rayo_transp_vdf, 218
 frontiere_ouverte_rayo_transp_vef, 218

 frontiere_ouverte_rho_u_impose, 218
 frontiere_ouverte_temperature_imposee, 218
 frontiere_ouverte_temperature_imposee_rayo_semi-
 _transp, 219
 frontiere_ouverte_temperature_imposee_rayo_transp,
 219
 frontiere_ouverte_vitesse_imposee, 219
 frontiere_ouverte_vitesse_imposee_sortie, 219

 gaz_parfait, 257
 gaz_reel_rhot, 256
 GCP, 206, 209
 gcp, 209
 gcp_ns, 203
 gen, 204
 generic, 114
 gmres, 204
 Gradient, 206

 IBICGSTAB, 206
 ilu, 271
 implicit_euler_steady_scheme, 274
 implicit_steady, 313
 implicite, 314
 implicite_ALE, 315
 imposer_vit_bords_ale, 32
 imprimer_flux, 32
 imprimer_flux_sum, 32
 init_par_partie, 243
 integrer_champ_med, 32
 Interface, 207
 internes, 38
 interpolation_champ_face_deriv, 185
 interpreter, 15
 interpreter_geometrique_base, 33

 Jones_Launders, 162

 k_epsilon, 160
 K_Epsilon_Realisable, 162
 kquick, 114

 Lam_Bremhorst, 161
 lata_to_med, 33
 lata_to_other, 34
 Launder_Sharma, 161
 leap_frog, 282
 lineaire, 185
 lire_ideas, 34
 lire_medfile, 16
 lire_tgrid, 49
 list_bloc_mailler, 35
 list_bord, 36
 list_nom, 54
 list_nom_virgule, 190

liste_post, 76
 liste_post_ok, 74
 listobj, 342
 listobj_impl, 342
 local, 208
 loi_analytique_scalaire, 339
 loi_ciofalo_hydr, 335
 loi_etat_base, 256
 loi_expert_hydr, 335
 loi_expert_scalaire, 339
 loi_fermeture_base, 258
 loi_fermeture_test, 258
 loi_horaire, 182, 258
 loi_odvm, 340
 loi_paroι_nu_impose, 340
 loi_puissance_hydr, 336
 loi_standard_hydr, 336
 loi_standard_hydr_old, 336
 loi_standard_hydr_scalaire, 341
 loi_ww_hydr, 336
 loi_WW_scalaire, 339
 longitudinale, 326
 longueur_melange, 148

 mailler, 34
 mailler_base, 35
 maillerparallel, 39
 masse_ajoutee, 329
 melange_gaz_parfait, 257
 methode_transport_deriv, 182
 metis, 268
 milieu_base, 259
 milieu_v2_base, 263
 mod_turb_hyd_rans, 159
 mod_turb_hyd_ss_maille, 143
 Modele_Fonc_Realisable, 201
 Modele_Fonc_Realisable_base, 201
 modele_fonction_bas_reynolds_base, 161
 modele_rayo_semi_transp, 78
 modele_rayonnement_base, 263
 modele_rayonnement_milieu_transparent, 263
 Modele_Shih_Zhu_Lumley_VDF, 201
 modele_turbulence_hyd_deriv, 142
 modele_turbulence_scal_base, 265
 modif_bord_to_raccord, 40
 mor_eqn, 109
 Moyenne, 71, 74
 moyenne, 72, 196
 moyenne_volumique, 41
 muscl, 115
 muscl3, 112
 muscl_new, 115
 muscl_old, 115

 N, 207
 navier_stokes_ft_disc, 139
 navier_stokes_phase_field, 169
 navier_stokes_qc, 171
 navier_stokes_standard, 173
 navier_stokes_turbulent, 175
 navier_stokes_turbulent_qc, 177
 negligeable, 115, 117, 336
 negligeable_scalaire, 341
 nettoiepasnoeuds, 42
 neumann, 220
 Neumann_homogene, 211
 Neumann_paroι_adiabatique, 211
 nom, 267
 NUL, 143
 NULL, 208
 numero_elem_sur_maitre, 67

 objet_lecture, 343
 Op_Conv_EF_Stab_PolyMAC_Face, 15
 optimal, 205
 option, 119
 option_vdf, 42
 orientefacesbord, 42
 orienter_simplexes, 49

 p1b, 117
 p1ncp1b, 117
 parametre_diffusion_implicite, 121
 parametre_equation_base, 121
 parametre_implicite, 122
 Paroi, 211
 paroi_adiabatique, 220
 paroi_contact, 220
 paroi_contact_fictif, 221
 paroi_decalee_robin, 221
 paroi_defilante, 221
 paroi_echange_contact_correlation_vdf, 222
 paroi_echange_contact_correlation_vdf, 223
 paroi_echange_contact_odvm_vdf, 224
 paroi_echange_contact_rayo_semi_transp_vdf, 224
 paroi_echange_contact_vdf, 224
 paroi_echange_contact_vdf_ft, 225
 paroi_echange_contact_vdf_zoom_fin, 225
 paroi_echange_contact_vdf_zoom_grossier, 225
 paroi_echange_externe_impose, 226
 paroi_echange_externe_impose_h, 226
 paroi_echange_externe_impose_rayo_semi_transp, 226
 paroi_echange_externe_impose_rayo_transp, 227
 paroi_echange_global_impose, 227
 paroi_fixe, 227
 paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses-
 _sommets, 228
 paroi_flux_impose, 228

paroi_flux_impose_rayo_semi_transp_vdf, 228
 paroi_flux_impose_rayo_semi_transp_vef, 228
 paroi_flux_impose_rayo_transp, 228
 paroi_ft_disc, 229
 paroi_ft_disc_deriv, 229
 paroi_knudsen_non_negligeable, 229
 paroi_rugueuse, 230
 paroi_tble, 337
 paroi_tble_scal, 341
 paroi_temperature_imposee, 230
 paroi_temperature_imposee_rayo_semi_transp, 230
 paroi_temperature_imposee_rayo_transp, 231
 partition, 42, 269
 partitionneur_deriv, 268
 pave, 35
 pb_avec_passif, 80
 Pb_base, 63
 pb_conduction, 81
 pb_conduction_milieu_variable, 82
 pb_couple_rayo_semi_transp, 83
 pb_couple_rayonnement, 108
 pb_gen_base, 63
 pb_hydraulique, 83
 pb_hydraulique_ALE, 84
 pb_hydraulique_concentration, 85
 pb_hydraulique_concentration_scalaires_passifs, 86
 pb_hydraulique_concentration_turbulent, 87
 pb_hydraulique_concentration_turbulent_scalaires_passifs, 88
 pb_hydraulique_turbulent, 90
 pb_mg, 90
 pb_phase_field, 91
 pb_thermohydraulique, 93
 pb_thermohydraulique_concentration, 94
 pb_thermohydraulique_concentration_scalaires_passifs, 95
 pb_thermohydraulique_concentration_turbulent, 96
 pb_thermohydraulique_concentration_turbulent_scalaires_passifs, 97
 pb_thermohydraulique_qc, 98
 pb_thermohydraulique_qc_fraction_massique, 99
 pb_thermohydraulique_scalaires_passifs, 100
 pb_thermohydraulique_turbulent, 102
 pb_thermohydraulique_turbulent_qc, 103
 pb_thermohydraulique_turbulent_qc_fraction_massique, 104
 pb_thermohydraulique_turbulent_scalaires_passifs, 105
 pbc_med, 106
 periodique, 231
 perte_charge_anisotrope, 324
 perte_charge_circulaire, 324
 perte_charge_directionnelle, 325
 perte_charge_isotrope, 325
 perte_charge_reguliere, 325
 perte_charge_singuliere, 327
 Petsc, 206, 208
 petsc, 205
 pilote_icoco, 44
 piso, 316
 plan, 68
 point, 67
 points, 66
 polymac, 233
 porosites, 44
 porosites_champ, 45
 position_like, 67
 post_processing, 75
 post_processings, 74
 postraitement_base, 75
 postraitement_ft_lata, 76
 postraiter_domaine, 45
 pp, 134
 prandtl, 265
 precisiongeom, 46
 Precond, 206, 208
 precondition_base, 271
 precondsolv, 272
 predefini, 196
 Pression, 71, 74
 Print, 207
 problem_read_generic, 107
 probleme_couple, 77
 probleme_ft_disc_gen, 108
 profils_thermo, 168
 puissance_thermique, 327
 quick, 115
 raccord, 38
 raffiner_anisotrope, 46
 raffiner_isotrope, 47
 Raffiner_isotrope_parallele, 15
 read, 48
 read_file, 48
 read_file_binary, 48
 read_med, 16
 read_unsupported_ascii_file_from_icem, 49
 redresser_hexaedres_vdf, 49
 refine_mesh, 50
 regroupebord, 50
 remove_elem, 50
 remove_invalid_internal_boundaries, 51
 reordonner, 52
 reordonner_faces_periodiques, 51
 reorienter_tetraedres, 52
 reorienter_triangles, 52
 rk3_ft, 284
 rotation, 52

RT, 116
 runge_kutta_ordre_3, 286
 runge_kutta_ordre_4_d3p, 288
 runge_kutta_rationnel_ordre_2, 289

 scalaire_impose_parois, 231
 scatter, 53
 scatterformatte, 53
 scattermed, 53
 Sch_CN_EX_iteratif, 276
 Sch_CN_iteratif, 279
 schema_adams_bashforth_order_2, 291
 schema_adams_bashforth_order_3, 293
 schema_adams_moulton_order_2, 294
 schema_adams_moulton_order_3, 297
 schema_backward_differentiation_order_2, 299
 schema_backward_differentiation_order_3, 301
 schema_euler_explicite_ALE, 311
 schema_implicite_base, 306
 schema_phase_field, 308
 schema_predictor_corrector, 310
 schema_temps_base, 273
 scheme_euler_explicit, 281
 scheme_euler_implicit, 304
 schmidt, 266
 segment, 68
 segmentfacesx, 69
 segmentfacesy, 69
 segmentfacesz, 70
 segmentpoints, 67
 Shih_Zhu_Lumley, 201
 simple, 317
 simpler, 318
 solide, 262
 solve, 54
 Solver, 206, 209
 Solver_moving_mesh_ALE, 17
 Solveur, 206, 208
 solveur_implicite_base, 313
 solveur_lineaire_std, 318
 solveur_sys_base, 210
 solveur_u_p, 319
 Solveur_pression, 206, 208
 sonde_base, 66
 sortie_libre_rho_variable, 231
 sortie_libre_temperature_imposee_h, 232
 source_base, 320
 source_con_phase_field, 327
 source_constituant, 329
 source_generique, 329
 source_qdm, 329
 source_qdm_lambdaup, 330
 source_qdm_phase_field, 330
 source_rayo_semi_transp, 330

 source_robin, 331
 source_robin_scalaire, 331
 source_th_tdivu, 331
 source_transport_k_eps, 332
 source_transport_k_eps_aniso_concen, 332
 source_transport_k_eps_aniso_therm_concen, 333
 Source_Transport_K_Eps_anisotherme, 320
 sources, 120
 sous_domaine, 269
 sous_maille, 150
 sous_maille_1elt, 154
 sous_maille_1elt_selectif_mod, 155
 sous_maille_axi, 156
 sous_maille_dyn, 266
 sous_maille_selectif, 153
 sous_maille_selectif_mod, 151
 sous_maille_smago, 146
 sous_maille_smago_dyn, 158
 sous_maille_smago_filtre, 157
 sous_maille_wale, 145
 sous_zone, 333
 sous_zones, 270
 Spai, 208
 spec_pdc_base, 326
 SSOR, 208, 209
 ssor, 272
 ssor_bloc, 272
 stab, 117
 standard, 118
 standard_KEps, 162
 stat_post_deriv, 72
 Statistiques, 71, 73, 74
 Statistiques_en_serie, 73, 74
 supg, 116
 supprime_bord, 54
 symetrie, 229, 232
 system, 54

 t_deb, 72
 t_fin, 72
 tayl_green, 244
 Temperature, 71, 74
 temperature, 164
 temperature_imposee_parois, 232
 test_solveur, 55
 testeur, 55
 testeur_medcoupling, 56
 tetraedriser, 56
 tetraedriser_homogene, 56
 tetraedriser_homogene_compact, 57
 tetraedriser_homogene_fin, 58
 tetraedriser_par_prisme, 58
 thi, 166
 thi_thermo, 167

trainee, [332](#)
traitement_particulier_base, [164](#)
tranche, [270](#)
transformer, [59](#)
transport_interfaces_ft_disc, [178](#)
Transport_K_Eps_Realisable, [109](#)
transport_k_epsilon, [185](#)
transport_marqueur_ft, [186](#)
transversale, [326](#)
triangler, [59](#)
triangler_fin, [59](#)
triangler_h, [60](#)
turbulence_paro_base, [335](#)
turbulence_paro_scalaire_base, [339](#)
type, [71](#), [74](#), [207](#), [208](#)

uniform_field, [244](#)
union, [271](#)
utau_imp, [338](#)

valeur_totale_sur_volume, [244](#)
vdf, [233](#)
vect_nom, [61](#)
vef, [233](#)
vefprep1b, [233](#)
verifier_qualite_raffinements, [60](#)
verifier_simplexes, [61](#)
verifiercoin, [61](#)
Vitesse, [71](#), [74](#)
vitesse_imposee, [182](#)
vitesse_interpolee, [182](#)
volume, [68](#)

xyz, [13](#)