

TrioCFD Reference Manual V1.7.4

Support team: triou@cea.fr

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

December 12, 2016

Contents

1	Syntax to define a mathematical function	14
2	interprete	16
2.1	Raffiner_isotrope_parallele	16
2.2	analyse_angle	16
2.3	associate	17
2.4	associer_algo	17
2.5	associer_pbmgb_pbfm	17
2.6	associer_pbmgb_pbgglobal	18
2.7	axi	18
2.8	bidim_axi	18
2.9	calculer_moments	18
2.10	lecture_bloc_moment_base	19
2.10.1	calcul	19
2.10.2	centre_de_gravite	19
2.10.3	un_point	19
2.11	corriger_frontiere_periodique	19
2.12	create_domain_from_sous_zone	20
2.13	debog	20
2.14	{	21
2.15	decoupebord_pour_rayonnement	21
2.16	decouper_bord_coincident	22
2.17	dilate	22
2.18	dimension	22
2.19	discretiser_domaine	23
2.20	discretize	23
2.21	distance_pari	23
2.22	ecrire_champ_med	24
2.23	ecrire_fichier_formatte	24
2.24	ecriturelecturespecial	24
2.25	execute_parallel	24
2.26	export	25
2.27	extract_2d_from_3d	25
2.28	extract_2daxi_from_3d	25
2.29	extraire_domaine	26
2.30	extraire_plan	26
2.31	extraire_surface	27
2.32	extrudebord	27
2.33	extrudeparoi	28
2.34	extruder	29
2.35	troisf	29
2.36	extruder_en20	29
2.37	extruder_en3	30
2.38	end	30
2.39	}	31
2.40	imposer_vit_bords_ale	31
2.41	bloc_lecture	31
2.42	imprimer_flux	31
2.43	imprimer_flux_sum	32
2.44	integrer_champ_med	32
2.45	lata_to_med	32
2.46	format_lata_to_med	33

2.47	lata_to_other	33
2.48	lire_ideas	33
2.49	mailler	34
2.50	list_bloc_mailler	34
2.50.1	mailler_base	34
2.50.2	pave	34
2.50.3	bloc_pave	34
2.50.4	list_bord	35
2.50.5	bord_base	35
2.50.6	bord	36
2.50.7	defbord	36
2.50.8	defbord_2	36
2.50.9	defbord_3	36
2.50.10	raccord	37
2.50.11	internes	37
2.50.12	epsilon	37
2.50.13	domain	38
2.51	maillerparallel	38
2.52	modif_bord_to_raccord	39
2.53	moyenne_volumique	39
2.54	nettoiepasnoeuds	40
2.55	option_vdf	41
2.56	orientefacesbord	41
2.57	partition	41
2.58	bloc_decouper	41
2.59	pilote_icoco	43
2.60	porosites	43
2.61	bloc_lecture_poro	43
2.62	porosites_champ	44
2.63	postraiter_domaine	44
2.64	precisiongeom	44
2.65	raffiner_anisotrope	45
2.66	raffiner_isotrope	45
2.67	read	45
2.68	read_file	46
2.69	read_file_binary	46
2.70	lire_tgrid	46
2.71	read_unsupported_ascii_file_from_icem	46
2.72	read_med	47
2.73	orienter_simplexes	47
2.74	redresser_hexaedres_vdf	48
2.75	regroupebord	48
2.76	remove_elem	48
2.77	remove_elem_bloc	49
2.78	remove_invalid_internal_boundaries	49
2.79	reordonner_faces_periodiques	50
2.80	reorienter_tetraedres	50
2.81	reorienter_triangles	50
2.82	reordonner	50
2.83	rotation	51
2.84	scatter	51
2.85	scatterformatte	51
2.86	scattermed	52
2.87	solve	52

2.88	supprime_bord	52
2.89	list_nom	52
2.90	system	53
2.91	test_solveur	53
2.92	testeur	53
2.93	testeur_medcoupling	54
2.94	tetraedriser	54
2.95	tetraedriser_homogene	54
2.96	tetraedriser_homogene_compact	55
2.97	tetraedriser_homogene_fin	55
2.98	tetraedriser_par_prisme	55
2.99	transformer	56
2.100	triangler	56
2.101	triangler_fin	56
2.102	triangler_h	56
2.103	verifier_qualite_raffinements	57
2.104	vect_nom	57
2.105	verifier_simplexes	57
2.106	verifiercoin	57
2.107	ecrire	58
2.108	ecrire_fichier_bin	58
2.109	ecrire_med	58
3	pb_gen_base	59
3.1	Pb_base	59
3.2	corps_postraitement	60
3.2.1	definition_champs	60
3.2.2	definition_champ	61
3.2.3	sondes	61
3.2.4	sonde	61
3.2.5	sonde_base	62
3.2.6	points	62
3.2.7	listpoints	62
3.2.8	point	62
3.2.9	segmentpoints	62
3.2.10	numero_elem_sur_maitre	63
3.2.11	position_like	63
3.2.12	segment	63
3.2.13	plan	63
3.2.14	volume	64
3.2.15	circle	64
3.2.16	circle_3	64
3.2.17	champs_posts	65
3.2.18	champs_a_post	65
3.2.19	champ_a_post	65
3.2.20	stats_posts	66
3.2.21	list_stat_post	67
3.2.22	stat_post_deriv	67
3.2.23	t_deb	67
3.2.24	t_fin	67
3.2.25	moyenne	67
3.2.26	ecart_type	68
3.2.27	correlation	68
3.2.28	stats_serie_posts	68

3.3	post_processings	69
3.3.1	un_postraitement	69
3.4	liste_post_ok	69
3.4.1	nom_postraitement	70
3.4.2	postraitement_base	70
3.4.3	post_processing	70
3.4.4	postraitement_ft_lata	71
3.5	liste_post	71
3.5.1	un_postraitement_spec	71
3.5.2	type_un_post	71
3.5.3	type_postraitement_ft_lata	72
3.6	format_file	72
3.7	probleme_couple	72
3.8	list_list_nom	73
3.9	modele_rayo_semi_transp	73
3.10	eq_rayo_semi_transp	74
3.10.1	condlims	74
3.10.2	condlimlu	74
3.11	pb_avec_passif	75
3.12	listeqn	76
3.13	pb_conduction	76
3.14	pb_couple_rayo_semi_transp	77
3.15	pb_hydraulique	77
3.16	pb_hydraulique_concentration	78
3.17	pb_hydraulique_concentration_scalaires_passifs	79
3.18	pb_hydraulique_concentration_turbulent	80
3.19	pb_hydraulique_concentration_turbulent_scalaires_passifs	81
3.20	pb_hydraulique_turbulent	83
3.21	pb_mg	84
3.22	pb_phase_field	84
3.23	pb_post	85
3.24	pb_thermohydraulique	86
3.25	pb_thermohydraulique_concentration	87
3.26	pb_thermohydraulique_concentration_scalaires_passifs	88
3.27	pb_thermohydraulique_concentration_turbulent	89
3.28	pb_thermohydraulique_concentration_turbulent_scalaires_passifs	90
3.29	pb_thermohydraulique_qc	91
3.30	pb_thermohydraulique_qc_fraction_massique	92
3.31	pb_thermohydraulique_scalaires_passifs	94
3.32	pb_thermohydraulique_turbulent	95
3.33	pb_thermohydraulique_turbulent_qc	96
3.34	pb_thermohydraulique_turbulent_qc_fraction_massique	97
3.35	pb_thermohydraulique_turbulent_scalaires_passifs	98
3.36	pb_med	99
3.37	list_info_med	99
3.37.1	info_med	99
3.38	problem_read_generic	100
3.39	pb_couple_rayonnement	101
3.40	probleme_ft_disc_gen	101

4	mor_eqn	102
4.1	conduction	102
4.2	bloc_diffusion	103
4.2.1	diffusion_deriv	103
4.2.2	negligeable	104
4.2.3	p1b	104
4.2.4	p1ncp1b	104
4.2.5	stab	104
4.2.6	standard	105
4.2.7	bloc_diffusion_standard	105
4.2.8	option	106
4.2.9	op_implicite	106
4.3	condinits	106
4.3.1	condinit	106
4.4	sources	107
4.5	ecrire_fichier_xyz_valeur_param	107
4.5.1	ecrire_fichier_xyz_valeur_item	107
4.5.2	bords_ecrire	107
4.6	parametre_equation_base	108
4.6.1	parametre_diffusion_implicite	108
4.6.2	parametre_implicite	108
4.7	convection_diffusion_chaleur_qc	109
4.8	bloc_convection	110
4.8.1	convection_deriv	110
4.8.2	amont	110
4.8.3	amont_old	111
4.8.4	centre	111
4.8.5	centre4	111
4.8.6	centre_old	111
4.8.7	di_l2	111
4.8.8	ef	111
4.8.9	bloc_ef	112
4.8.10	muscl3	112
4.8.11	ef_stab	113
4.8.12	listsous_zone_valeur	113
4.8.13	sous_zone_valeur	113
4.8.14	generic	114
4.8.15	kquick	114
4.8.16	muscl	114
4.8.17	muscl_old	114
4.8.18	muscl_new	115
4.8.19	negligeable	115
4.8.20	quick	115
4.8.21	supg	115
4.8.22	btd	115
4.8.23	ale	116
4.9	convection_diffusion_chaleur_turbulent_qc	116
4.10	convection_diffusion_concentration	117
4.11	convection_diffusion_concentration_ft_disc	118
4.12	convection_diffusion_concentration_turbulent	120
4.13	convection_diffusion_fraction_massique_qc	121
4.14	convection_diffusion_fraction_massique_turbulent_qc	122
4.15	convection_diffusion_phase_field	123
4.16	convection_diffusion_temperature	124

4.17	pp	125
4.17.1	penalisation_l2_ftd_lec	125
4.18	convection_diffusion_temperature_ft_disc	126
4.19	objet_lecture_maintien_temperature	127
4.20	convection_diffusion_temperature_turbulent	127
4.21	eqn_base	128
4.22	navier_stokes_ft_disc	129
4.23	penalisation_forage	132
4.24	modele_turbulence_hyd_deriv	132
4.24.1	dt_impr_ustar_mean_only	133
4.24.2	NUL	134
4.24.3	mod_turb_hyd_ss_maille	134
4.24.4	form_a_nb_points	136
4.24.5	sous_maille_wale	136
4.24.6	sous_maille_smago	137
4.24.7	combinaison	138
4.24.8	longueur_melange	140
4.24.9	sous_maille	141
4.24.10	sous_maille_selectif_mod	143
4.24.11	deuxentiers	144
4.24.12	floatentier	144
4.24.13	sous_maille_selectif	144
4.24.14	sous_maille_1elt	146
4.24.15	sous_maille_1elt_selectif_mod	147
4.24.16	sous_maille_axi	148
4.24.17	sous_maille_smago_filtre	149
4.24.18	sous_maille_smago_dyn	150
4.24.19	k_epsilon	152
4.24.20	modele_fonction_bas_reynolds_base	153
4.24.21	Lam_Bremhorst	153
4.24.22	standard_KEps	153
4.24.23	Launder_Sharma	154
4.24.24	Jones_Launder	154
4.24.25	k_epsilon_bas_reynolds	154
4.24.26	deuxmots	155
4.24.27	k_epsilon_v2	155
4.24.28	k_epsilon_2_couches	156
4.25	floatfloat	157
4.26	traitement_particulier	158
4.26.1	traitement_particulier_base	158
4.26.2	temperature	158
4.26.3	canal	158
4.26.4	ec	159
4.26.5	thi	159
4.26.6	thi_thermo	160
4.26.7	chmoy_faceperio	161
4.26.8	concmoy	161
4.26.9	profils_thermo	162
4.26.10	brech	162
4.26.11	ceg	162
4.26.12	ceg_areva	163
4.26.13	ceg_cea_jaea	163
4.27	navier_stokes_phase_field	163
4.28	navier_stokes_qc	165

4.29	navier_stokes_standard	167
4.30	navier_stokes_turbulent	169
4.31	navier_stokes_turbulent_qc	170
4.32	transport_interfaces_ft_disc	172
4.33	methode_transport_deriv	176
4.33.1	loi_horaire	176
4.33.2	vitesse_imposee	176
4.33.3	vitesse_interpolee	177
4.34	bloc_lecture_remaillage	177
4.35	parcours_interface	178
4.36	interpolation_champ_face_deriv	179
4.36.1	base	179
4.36.2	lineaire	179
4.37	transport_k_epsilon	179
4.38	transport_marqueur_ft	180
4.39	injection_marqueur	182
5	algo_base	182
5.1	algo_couple_1	183
6	/*	183
6.1	/*	183
7	champ_generique_base	183
7.1	champ_post_de_champs_post	183
7.2	list_nom_virgule	184
7.3	listchamp_generique	184
7.4	champ_post_operateur_base	184
7.5	champ_post_operateur_eqn	185
7.6	champ_post_statistiques_base	185
7.7	correlation	186
7.8	champ_post_operateur_divergence	186
7.9	ecart_type	187
7.10	champ_post_extraction	187
7.11	champ_post_operateur_gradient	188
7.12	champ_post_interpolation	189
7.13	champ_post_morceau_equation	189
7.14	moyenne	190
7.15	predefini	191
7.16	champ_post_reduction_0d	191
7.17	champ_post_refchamp	192
7.18	champ_post_tparoi_vef	192
7.19	champ_post_transformation	193
8	chimie	193
8.1	reactions	194
8.1.1	reaction	194
9	class_generic	195
9.1	cholesky	195
9.2	dt_calc	195
9.3	dt_fixe	195
9.4	dt_min	195
9.5	dt_start	196

9.6	gcp_ns	196
9.7	gen	197
9.8	gmres	197
9.9	optimal	197
9.10	petsc	198
9.11	gcp	202
9.12	solveur_sys_base	202
10	coeur	202
11	#	203
11.1	#	203
12	condlim_base	203
12.1	Paroi	204
12.2	contact_vdf_vef	204
12.3	contact_vef_vdf	204
12.4	dirichlet	204
12.5	echange_contact_rayo_transp_vdf	204
12.6	entree_temperature_imposee_h	205
12.7	flux_radiatif	205
12.8	flux_radiatif_vdf	205
12.9	flux_radiatif_vef	206
12.10	frontiere_ouverte	206
12.11	frontiere_ouverte_concentration_imposee	206
12.12	frontiere_ouverte_fraction_massique_imposee	207
12.13	frontiere_ouverte_gradient_pression_impose	207
12.14	frontiere_ouverte_gradient_pression_impose_vef	207
12.15	frontiere_ouverte_gradient_pression_impose_vefprep1b	207
12.16	frontiere_ouverte_gradient_pression_libre_vef	208
12.17	frontiere_ouverte_gradient_pression_libre_vefprep1b	208
12.18	frontiere_ouverte_k_eps_impose	208
12.19	frontiere_ouverte_pression_imposee	208
12.20	frontiere_ouverte_pression_imposee_orlansky	208
12.21	frontiere_ouverte_pression_moyenne_imposee	209
12.22	frontiere_ouverte_rayo_semi_transp	209
12.23	frontiere_ouverte_rayo_transp	209
12.24	frontiere_ouverte_rayo_transp_vdf	210
12.25	frontiere_ouverte_rayo_transp_vef	210
12.26	frontiere_ouverte_rho_u_impose	210
12.27	frontiere_ouverte_temperature_imposee	210
12.28	frontiere_ouverte_temperature_imposee_rayo_semi_transp	211
12.29	frontiere_ouverte_temperature_imposee_rayo_transp	211
12.30	frontiere_ouverte_vitesse_imposee	211
12.31	frontiere_ouverte_vitesse_imposee_sortie	211
12.32	neumann	212
12.33	paroi_adiabatique	212
12.34	paroi_contact	212
12.35	paroi_contact_fictif	213
12.36	paroi_couple	213
12.37	paroi_decalee_robin	213
12.38	paroi_defilante	214
12.39	paroi_echange_contact_correlation_vdf	214
12.40	paroi_echange_contact_correlation_vef	215

12.41	paroi_echange_contact_odvm_vdf	216
12.42	paroi_echange_contact_rayo_semi_transp_vdf	216
12.43	paroi_echange_contact_vdf	217
12.44	paroi_echange_contact_vdf_ft	217
12.45	paroi_echange_contact_vdf_zoom_fin	217
12.46	paroi_echange_contact_vdf_zoom_grossier	218
12.47	paroi_echange_externer_impose	218
12.48	paroi_echange_externer_impose_h	218
12.49	paroi_echange_externer_impose_rayo_semi_transp	219
12.50	paroi_echange_externer_impose_rayo_transp	219
12.51	paroi_echange_global_impose	219
12.52	paroi_fixe	219
12.53	paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets	220
12.54	paroi_flux_impose	220
12.55	paroi_flux_impose_rayo_semi_transp_vdf	220
12.56	paroi_flux_impose_rayo_semi_transp_vef	220
12.57	paroi_flux_impose_rayo_transp	221
12.58	paroi_ft_disc	221
12.59	paroi_ft_disc_deriv	221
12.59.1	symetrie	221
12.59.2	constant	221
12.60	paroi_knudsen_non_negligeable	222
12.61	paroi_rugueuse	222
12.62	paroi_temperature_imposee	222
12.63	paroi_temperature_imposee_rayo_semi_transp	223
12.64	paroi_temperature_imposee_rayo_transp	223
12.65	periodique	223
12.66	sortie_libre_rho_variable	223
12.67	sortie_libre_temperature_imposee_h	224
12.68	symetrie	224
12.69	temperature_imposee_paro	224
13	discretisation_base	224
13.1	ef	224
13.2	vdf	225
13.3	vef	225
13.4	vefprep1b	225
14	domaine	226
14.1	domaine_ale	226
15	espece	226
16	champ_base	226
16.1	champ_base	226
16.2	champ_don_base	226
16.3	champ_don_lu	227
16.4	champ_fonc_fonction	227
16.5	champ_fonc_fonction_txyz	227
16.6	champ_fonc_med	228
16.7	champ_fonc_reprise	228
16.8	fonction_champ_reprise	229
16.9	champ_fonc_t	229
16.10	champ_fonc_tabule	229

16.11	champ_init_canal_sinal	229
16.12	bloc_lec_champ_init_canal_sinal	230
16.13	champ_input_base	230
16.14	champ_input_p0	231
16.15	champ_ostwald	231
16.16	champ_som_lu_vdf	232
16.17	champ_som_lu_vef	232
16.18	champ_tabule_temps	232
16.19	champ_uniforme_morceaux	233
16.20	champ_uniforme_morceaux_tabule_temps	233
16.21	champ_fonc_txyz	233
16.22	champ_fonc_xyz	234
16.23	field_uniform_keps_from_ud	234
16.24	init_par_partie	234
16.25	tayl_green	234
16.26	uniform_field	235
16.27	valeur_totale_sur_volume	235
17	champ_front_base	235
17.1	champ_front_base	235
17.2	boundary_field_inward	236
17.3	boundary_field_uniform_keps_from_ud	236
17.4	ch_front_input	236
17.5	ch_front_input_uniforme	237
17.6	champ_front_ale	237
17.7	champ_front_bruite	237
17.8	champ_front_calc	238
17.9	champ_front_contact_rayo_semi_transp_vef	238
17.10	champ_front_contact_rayo_transp_vef	239
17.11	champ_front_contact_vef	239
17.12	champ_front_debit	239
17.13	champ_front_fonc_pois_ipsn	239
17.14	champ_front_fonc_pois_tube	240
17.15	champ_front_fonc_txyz	240
17.16	champ_front_fonc_xyz	240
17.17	champ_front_fonction	241
17.18	champ_front_lu	241
17.19	champ_front_normal_vef	241
17.20	champ_front_pression_from_u	241
17.21	champ_front_recyclage	242
17.22	champ_front_tabule	243
17.23	champ_front_tangentiel_vef	244
17.24	champ_front_uniforme	244
17.25	champ_front_vortex	244
17.26	champ_front_zoom	244
18	loi_etat_base	245
18.1	gaz_reel_rhot	245
18.2	melange_gaz_parfait	245
18.3	gaz_parfait	245
19	loi_horaire	246

20	milieu_base	246
20.1	constituant	247
20.2	fluide_incompressible	247
20.3	fluide_ostwald	248
20.4	fluide_quasi_compressible	248
20.5	bloc_sutherland	249
20.6	solide	250
21	milieu_v2_base	250
21.1	fluide_diphasique	250
22	modele_rayonnement_base	250
22.1	modele_rayonnement_milieu_transparent	251
23	modele_turbulence_scal_base	252
23.1	fluctuation_temperature_w_bas_re	253
23.2	prandtl	253
23.3	schmidt	254
23.4	sous_maille_dyn	255
24	nom	255
24.1	nom_anonyme	255
25	partitionneur_deriv	256
25.1	fichier_decoupage	256
25.2	metis	256
25.3	partition	257
25.4	sous_zones	257
25.5	tranche	258
26	precond_base	258
26.1	precond_local	259
26.2	precondsolv	259
26.3	ssor	259
26.4	ssor_bloc	259
27	schema_temps_base	260
27.1	Sch_CN_EX_iteratif	261
27.2	Sch_CN_iteratif	263
27.3	scheme_euler_explicit	265
27.4	leap_frog	267
27.5	rk3_ft	269
27.6	runge_kutta_ordre_3	270
27.7	runge_kutta_ordre_4_d3p	272
27.8	runge_kutta_rationnel_ordre_2	274
27.9	schema_adams_bashforth_order_2	275
27.10	schema_adams_bashforth_order_3	277
27.11	schema_adams_moulton_order_2	278
27.12	schema_adams_moulton_order_3	281
27.13	schema_backward_differentiation_order_2	283
27.14	schema_backward_differentiation_order_3	285
27.15	scheme_euler_implicit	287
27.16	schema_implicite_base	290
27.17	schema_phase_field	292
27.18	schema_predictor_corrector	294

28	solveur_implicite_base	295
28.1	implicite	295
28.2	piso	296
28.3	simple	297
28.4	simpler	298
28.5	solveur_lineaire_std	299
29	source_base	299
29.1	Source_Transport_K_Eps_anisotherme	299
29.2	acceleration	300
29.3	boussinesq_concentration	300
29.4	boussinesq_temperature	301
29.5	canal_perio	301
29.6	coriolis	302
29.7	darcy	302
29.8	dirac	303
29.9	forchheimer	303
29.10	perte_charge_anisotrope	303
29.11	perte_charge_circulaire	304
29.12	perte_charge_directionnelle	304
29.13	perte_charge_isotrope	304
29.14	perte_charge_reguliere	305
29.15	spec_pdc_base	305
29.15.1	longitudinale	305
29.15.2	transversale	306
29.16	perte_charge_singuliere	306
29.17	puissance_thermique	307
29.18	source_con_phase_field	307
29.19	source_constituant	308
29.20	flottabilite	308
29.21	source_generique	308
29.22	masse_ajoutee	309
29.23	source_qdm	309
29.24	source_qdm_lambdaup	309
29.25	source_qdm_phase_field	310
29.26	source_rayo_semi_transp	310
29.27	source_robin	310
29.28	source_robin_scalaire	310
29.29	listdeuxmots_sacc	311
29.30	source_th_tdivu	311
29.31	trainee	311
29.32	source_transport_k_eps	311
29.33	source_transport_k_eps_aniso_concen	312
29.34	source_transport_k_eps_aniso_therm_concen	312
29.35	source_transport_k_eps_bas_reynolds	312
30	sous_zone	313
30.1	bloc_origine_cotes	314
30.2	bloc_couronne	314
30.3	bloc_tube	314

31	turbulence_paroι_base	315
31.1	loi_ciofalo_hydr	315
31.2	loi_expert_hydr	315
31.3	loi_paroι_2_couches	315
31.4	loi_puissance_hydr	316
31.5	loi_standard_hydr	316
31.6	loi_standard_hydr_old	316
31.7	loi_ww_hydr	316
31.8	negligeable	316
31.9	paroι_tble	317
31.10	twofloat	317
31.11	liste_sonde_tble	318
31.11.1	sonde_tble	318
31.12	entierfloat	318
31.13	utau_imp	318
32	turbulence_paroι_scalaire_base	319
32.1	loi_WW_scalaire	319
32.2	loi_analytique_scalaire	319
32.3	loi_expert_scalaire	319
32.4	loi_odvm	320
32.5	loi_paroι_2_couches_scalaire	320
32.6	loi_paroι_nu_impose	320
32.7	loi_standard_hydr_scalaire	321
32.8	negligeable_scalaire	321
32.9	paroι_tble_scal	321
32.10	fourfloat	322
33	listobj_impl	322
33.1	list_un_pb	322
33.2	un_pb	322
33.3	listdeuxmots	323
33.4	listobj	323
34	objet_lecture	323
34.1	floattantchaïne	323
34.2	threefloat	324
35	index	324

1 Syntax to define a mathematical function

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

ABS : absolute value function
COS : cosinus function
SIN : sinus function
TAN : tan function
ATAN : arctan function
EXP : exponential function
LN : neperian logaithm function
SQRT : root mean square function
INT : integer function
ERF : erf function

RND(x) : random function (values between 0 and x)
 COSH : hyperbolic cosinus function
 SINH : hyperbolic sinus function
 TANH : hyperbolic tangent function
 ACOS : inverse cosinus function
 ATANH : inverse hyperbolic tangent function
 NOT(x) : not equal to x
 x_AND_y : and function (returns 1 if x and y true else 0)
 x_OR_y : or function (returns 1 if x or y true else 0)
 x_GT_y : greater to (returns 1 if x>y else 0)
 x_GE_y : greater or equal to (returns 1 if x>=y else 0)
 x_LT_y : lesser to (returns 1 if x<y else 0)
 x_LE_y : lesser or equal to (returns 1 if x<=y else 0)
 x_MIN_y : minimum of x and y
 x_MAX_y : maximum of x and y
 x_MOD_y : modular division of x per y
 x_EQ_y : equal to (returns 1 if x=y else 0)
 x_NEQ_y : not equal to (returns 1 if x!=y else 0)

You can also use the following operations:

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

You can also use the following constants:

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

The variables which can be used are:

x,y,z : coordinates
 t : time

Examples:

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

Possible error:

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

2 interprete

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

See also: [objet_u \(35\)](#) [read \(2.66\)](#) [associate \(2.2\)](#) [discretize \(2.19\)](#) [mailler \(2.48\)](#) [maillerparallel \(2.50.13\)](#) [ecrire_fichier_bin \(2.107\)](#) [ecrire \(2.106\)](#) [read_file \(2.67\)](#) [lire_tgrid \(2.69\)](#) [solve \(2.86\)](#) [execute_parallel \(2.24\)](#) [end \(2.37\)](#) [dimension \(2.17\)](#) [bidim_axi \(2.7\)](#) [axi \(2.6\)](#) [transformer \(2.98\)](#) [rotation \(2.82\)](#) [dilate \(2.16\)](#) [testeur \(2.91\)](#) [test_solveur \(2.90\)](#) [postraiter_domaine \(2.62\)](#) [modif_bord_to_raccord \(2.51\)](#) [remove_elem \(2.75\)](#) [regroupebord \(2.74\)](#) [supprime_bord \(2.87\)](#) [calculer_moments \(2.8\)](#) [imprimer_flux \(2.41\)](#) [decouper_bord_coincident \(2.15\)](#) [raffiner_anisotrope \(2.64\)](#) [raffiner_isotrope \(2.65\)](#) [triangler \(2.99\)](#) [tetraedriser \(2.93\)](#) [orientefacesbord \(2.55\)](#) [reorienter_tetraedres \(2.79\)](#) [reorienter_triangles \(2.80\)](#) [verifiercoin \(2.105\)](#) [porosites \(2.59\)](#) [porosites_champ \(2.61\)](#) [discretiser_domaine \(2.18\)](#) [{ \(2.13\) } \(2.38\)](#) [export \(2.25\)](#) [debug \(2.12\)](#) [pilote_icoco \(2.58\)](#) [moyenne_volumique \(2.52\)](#) [ecrire_champ_med \(2.21\)](#) [read_med \(2.71\)](#) [lire_ideas \(2.47\)](#) [ecrire_med \(2.108\)](#) [system \(2.89\)](#) [redresser_hexaedres_vdf \(2.73\)](#) [analyse_angle \(2.1\)](#) [remove_invalid_internal_boundaries \(2.77\)](#) [reordonner \(2.81\)](#) [option_vdf \(2.54\)](#) [precisiongeom \(2.63\)](#) [scatter \(2.83\)](#) [partition \(2.56\)](#) [reordonner_faces_periodiques \(2.78\)](#) [corriger_frontiere_periodique \(2.10.3\)](#) [distance_paroi \(2.20\)](#) [extrudebord \(2.31\)](#) [extruder \(2.33\)](#) [extract_2d_from_3d \(2.26\)](#) [extruder_en20 \(2.35\)](#) [extrudeparoi \(2.32\)](#) [ecriturelecturespecial \(2.23\)](#) [lata_to_med \(2.44\)](#) [lata_to_other \(2.46\)](#) [decoupebord_pour_rayonnement \(2.14\)](#) [extraire_plan \(2.29\)](#) [create_domain_from_sous_zone \(2.11\)](#) [extraire_domaine \(2.28\)](#) [extraire_surface \(2.30\)](#) [integrer_champ_med \(2.43\)](#) [orienter_simplexes \(2.72\)](#) [verifier_simplexes \(2.104\)](#) [verifier_qualite_raffinements \(2.102\)](#) [testeur_medcoupling \(2.92\)](#) [Raffiner_isotrope_parallele \(2\)](#) [imposer_vit_bords_ale \(2.39\)](#) [nettoiepasnoeuds \(2.53\)](#)

Usage:

interprete

2.1 Raffiner_isotrope_parallele

Description: Refine parallel mesh in parallel

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

Usage:

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

where

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

2.2 analyse_angle

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

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

Usage:

analyse_angle domain_name nb_histo
where

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

2.3 associate

Synonymous: **associer**

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

See also: [interprete \(2\)](#) [associer_pbm_glb \(2.5\)](#) [associer_pbm_glb \(2.4\)](#) [associer_algo \(2.3\)](#)

Usage:

associate objet_1 objet_2
where

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

2.4 associer_algo

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

See also: [associate \(2.2\)](#)

Usage:

associer_algo objet_1 objet_2
where

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

2.5 associer_pbm_glb

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

See also: [associate \(2.2\)](#)

Usage:

associer_pbm_glb objet_1 objet_2
where

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

2.6 associer_pbmng_pbgglobal

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

See also: [associate \(2.2\)](#)

Usage:

associer_pbmng_pbgglobal **objet_1** **objet_2**

where

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

2.7 axi

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

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

Usage:

axi

2.8 bidim_axi

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

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

Usage:

bidim_axi

2.9 calculer_moments

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

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

Usage:

calculer_moments **nom_dom** **mot**

where

- **nom_dom** *str*: Name of domain.
- **mot** *lecture_bloc_moment_base (2.9)*: Keyword.

2.10 lecture_bloc_moment_base

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

See also: objet_lecture (34) calcul (2.10) centre_de_gravite (2.10.1)

Usage:

2.10.1 calcul

Description: The centre of gravity will be calculated.

See also: (2.9)

Usage:

calcul

2.10.2 centre_de_gravite

Description: To specify a specific centre of gravity.

See also: (2.9)

Usage:

centre_de_gravite point

where

- **point** *un_point* (2.10.2): A centre of gravity.

2.10.3 un_point

Description: A point.

See also: objet_lecture (34)

Usage:

pos

where

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

2.11 corriger_frontiere_periodique

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

See also: interpret (2)

Usage:

corriger_frontiere_periodique {

domaine *str*

```

    bord str
    [ direction n x1 x2 ... xn]
    [ fichier_post str]
}
where

```

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

2.12 create_domain_from_sous_zone

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

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

Usage:

```

create_domain_from_sous_zone {
    domaine_final str
    par_sous_zone str
    domaine_init str
}
where

```

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

2.13 debug

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

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

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

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

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

Usage:

debog pb fichier1 fichier2 seuil mode

where

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

2.14 {

Description: Block's beginning.

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

Usage:

{

2.15 decoupebord_pour_rayonnement

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

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

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

Usage:

decoupebord_pour_rayonnement {

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

}

where

- **domaine** *str*
- **domaine_grossier** *str*

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

2.16 decouper_bord_coincident

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

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

Usage:

decouper_bord_coincident domain_name bord
where

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

2.17 dilate

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

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

Usage:

dilate domain_name alpha
where

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

2.18 dimension

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

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

Usage:

dimension dim
where

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

2.19 discretiser_domaine

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

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

Usage:

discretiser_domaine **domain_name**

where

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

2.20 discretize

Synonymous: **discretiser**

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

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

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

Usage:

discretize **problem_name** **dis**

where

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

2.21 distance_pari

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

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

Usage:

distance_pari **dom** **bords** **format**

where

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

2.22 ecrire_champ_med

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

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

Usage:

ecrire_champ_med **nom_dom** **nom_chp** **file**
where

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

2.23 ecrire_fichier_formatte

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

See also: [ecrire_fichier_bin \(2.107\)](#)

Usage:

ecrire_fichier_formatte **name_obj** **filename**
where

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

2.24 ecrirelecturespecial

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

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

Usage:

ecrirelecturespecial **type**
where

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

2.25 execute_parallel

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

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

Usage:

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


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

where

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

2.26 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](#) (2)

Usage:

export

2.27 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](#) (2) [extract_2daxi_from_3d](#) (2.27)

Usage:

extract_2d_from_3d dom3D bord dom2D

where

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

2.28 extract_2daxi_from_3d

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

See also: [extract_2d_from_3d](#) (2.26)

Usage:

extract_2daxi_from_3d dom3D bord dom2D

where

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

2.29 extraire_domaine

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

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

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

Usage:

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

where

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

2.30 extraire_plan

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

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

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

Usage:

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

}
where

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

2.31 extraire_surface

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

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

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

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

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

Usage:

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

where

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

2.32 extrudebord

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

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

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

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

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

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

Usage:

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

where

- **domaine_init** *str*: Initial domain with hexaedras or tetrahedras.
- **direction** *x1 x2 (x3)*: Directions for the extrusion.
- **nb_tranches** *int*: Number of elements in the extrusion direction.
- **domaine_final** *str*: Extruded domain.
- **nom_bord** *str*: Name of the boundary of the initial domain where extrusion will be applied.
- **non_perio** : Extruded domain will not have periodic boundaries. So, the boundaries will be named DEVANT and DERRIERE instead of PERIO.
- **hexa_old** : Old algorithm for boundary extrusion from a hexahedral mesh.
- **trois_tetra** : To extrude in 3 tetrahedras instead of 14 tetrahedras.
- **vingt_tetra** : To extrude in 20 tetrahedras instead of 14 tetrahedras.
- **sans_passer_par_le2D** *int*: Only for non regression

2.33 extrudeparoi

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

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

Usage:

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

}
where

- **domaine** *str*: Name of the domain.
- **nom_bord** *str*: Name of the (no slide) boundary for creation of prismatic layers.
- **epaisseur** *n x l 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

2.34 extruder

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

See also: [interprete \(2\)](#) [extruder_en3 \(2.36\)](#)

Usage:

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

- **domaine** *str*: Name of the domain.
- **direction** *troisf* [\(2.34\)](#): Direction of the extrude operation.
- **nb_tranches** *int*: Number of elements in the extrusion direction.

2.35 troisf

Description: Auxiliary class to extrude.

See also: [objet_lecture \(34\)](#)

Usage:

```
lx ly lz  
where
```

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

2.36 extruder_en20

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

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

Usage:

```
extruder_en20 {
```

```

    domaine str
    [ direction troisf]
    nb_tranches int
}
where

```

- **domaine** *str*: Name of the domain.
- **direction** *troisf* (2.34): 0 Direction of the extrude operation.
- **nb_tranches** *int*: Number of elements in the extrusion direction.

2.37 extruder_en3

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

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

See also: *extruder* (2.33)

Usage:

```

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

```

- **domaine** *n word1 word2 ... wordn*: List of the domains
- **nom_cl_devant** *str*: New name of the first boundary.
- **nom_cl_derriere** *str*: New name of the second boundary.
- **direction** *troisf* (2.34) for inheritance: Direction of the extrude operation.
- **nb_tranches** *int* for inheritance: Number of elements in the extrusion direction.

2.38 end

Synonymous: **fin**

Description: Keyword which must complete the data file.

See also: *interpret* (2)

Usage:

end

2.39 }

Description: Block's end.

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

Usage:
}

2.40 **imposer_vit_bords_ale**

Description: not_set

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

Usage:
imposer_vit_bords_ale **dom** **bloc**
where

- **dom** *str*: Name of domain.
- **bloc** *bloc_lecture* ([2.40](#)): Description.

2.41 **bloc_lecture**

Description: pour lire entre deux accolades

See also: [objet_lecture \(34\)](#)

Usage:
bloc_lecture
where

- **bloc_lecture** *str*

2.42 **imprimer_flux**

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

See also: [interpret \(2\)](#) [imprimer_flux_sum \(2.42\)](#)

Usage:
imprimer_flux **domain_name** **noms_bord**
where

- **domain_name** *str*: Name of the domain.
- **noms_bord** *bloc_lecture* ([2.40](#)): Liste des noms des bords ex: { Bord1 Bord2 }

2.43 imprimer_flux_sum

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

See also: imprimer_flux ([2.41](#))

Usage:

imprimer_flux_sum **domain_name** **noms_bord**

where

- **domain_name** *str*: Name of the domain.
- **noms_bord** *bloc_lecture* ([2.40](#)): Liste des noms des bords ex: { Bord1 Bord2 }

2.44 integrer_champ_med

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

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

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

Usage:

integrer_champ_med {

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

}

where

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

2.45 lata_to_med

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

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

Usage:

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

where

- **format** *format_lata_to_med* (2.45): generated file post_med.data use format (MED or MESHTV or LML keyword).
- **file** *str*: LATA file to convert to the new format.
- **file_med** *str*: Name of file med.

2.46 format_lata_to_med

Description: not_set

See also: objet_lecture (34)

Usage:

mot [**format**]

where

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

2.47 lata_to_other

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

See also: interpret (2)

Usage:

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

where

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

2.48 lire_ideas

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

See also: interpret (2)

Usage:

lire_ideas **nom_dom** **file**

where

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

2.49 mailler

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

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

Usage:

mailler domaine bloc

where

- **domaine** *str*: Name of domain.
- **bloc** *list_bloc_mailler* ([2.49](#)): Instructions to mesh.

2.50 list_bloc_mailler

Description: List of block mesh.

See also: [listobj \(33.3\)](#)

Usage:

{ *object1* , *object2* }

list of *mailler_base* ([2.50](#)) separated with ,

2.50.1 mailler_base

Description: Basic class to mesh.

See also: [objet_lecture \(34\)](#) [pave \(2.50.1\)](#) [epsilon \(2.50.11\)](#) [domain \(2.50.12\)](#)

Usage:

2.50.2 pave

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

See also: [mailler_base \(2.50\)](#)

Usage:

pave name bloc list_bord

where

- **name** *str*: Name of the pave (block).
- **bloc** *bloc_pave* ([2.50.2](#)): Definition of the pave (block).
- **list_bord** *list_bord* ([2.50.3](#)): Definition of boundaries of domain.

2.50.3 bloc_pave

Description: Class to create a pave.

See also: [objet_lecture \(34\)](#)

Usage:

{

```

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

```

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

2.50.4 list_bord

Description: The block sides.

See also: listobj ([33.3](#))

Usage:

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

list of *bord_base* ([2.50.4](#))

2.50.5 bord_base

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

See also: objet_lecture ([34](#)) bord ([2.50.5](#)) raccord ([2.50.9](#)) internes ([2.50.10](#))

Usage:

2.50.6 bord

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

See also: `bord_base` ([2.50.4](#))

Usage:

bord nom defbord

where

- **nom** *str*: Name of block side.
- **defbord** *defbord* ([2.50.6](#)): Definition of block side.

2.50.7 defbord

Description: Class to define an edge.

See also: `objet_lecture` ([34](#)) `defbord_2` ([2.50.7](#)) `defbord_3` ([2.50.8](#))

Usage:

2.50.8 defbord_2

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

See also: ([2.50.6](#))

Usage:

dir eq pos pos2_min inf1 dir2 inf2 pos2_max

where

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

2.50.9 defbord_3

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

See also: ([2.50.6](#))

Usage:

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

where

- **dir** *str into* [*'X'*, *'Y'*, *'Z'*]: Edge is perpendicular to this direction.
- **eq** *str into* [*'='*]: Equality sign.
- **pos** *float*: Position value.
- **pos2_min** *float*: Value minimal.

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

2.50.10 raccord

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

See also: `bord_base` ([2.50.4](#))

Usage:

raccord type1 type2 nom defbord

where

- **type1** *str into* [*'local', 'distant'*]: Contact type.
- **type2** *str into* [*'homogene'*]: Contact type.
- **nom** *str*: Name of block side.
- **defbord** *defbord* ([2.50.6](#)): Definition of block side.

2.50.11 internes

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

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

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

See also: `bord_base` ([2.50.4](#))

Usage:

internes nom defbord

where

- **nom** *str*: Name of block side.
- **defbord** *defbord* ([2.50.6](#)): Definition of block side.

2.50.12 epsilon

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

See also: `mailler_base` ([2.50](#))

Usage:

epsilon eps

where

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

2.50.13 domain

Description: Class to reuse a domain.

See also: `mailler_base` (2.50)

Usage:

domain **domain_name**

where

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

2.51 mailerparallel

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

See also: `interpret` (2)

Usage:

mailerparallel {

```

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

```

}

where

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

- **ghost_thickness** *int*: the number of ghost cells (equivalent to the `epaisseur_joint` parameter of `Decouper`).
- **perio_x** : change the splitting method to provide a valid mesh for periodic boundary conditions.
- **perio_y** : change the splitting method to provide a valid mesh for periodic boundary conditions.
- **perio_z** : change the splitting method to provide a valid mesh for periodic boundary conditions.
- **function_coord_x** *str*: By default, the meshing algorithm creates n_X n_Y n_Z 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 N_x floating point values used as nodes coordinates in the file.
- **file_coord_y** *str*: idem `file_coord_x` for y
- **file_coord_z** *str*: idem `file_coord_x` for z
- **boundary_xmin** *str*: the name of the boundary at the minimum X direction. If it not provided, the default boundary names are `xmin`, `xmax`, `ymin`, `ymax`, `zmin` and `zmax`. If the mesh is periodic in a given direction, only the MIN boundary name is used, for both sides of the box.
- **boundary_xmax** *str*
- **boundary_ymin** *str*
- **boundary_ymax** *str*
- **boundary_zmin** *str*
- **boundary_zmax** *str*

2.52 **modif_bord_to_raccord**

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

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

Usage:

modif_bord_to_raccord **domaine** **nom_bord**

where

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

2.53 **moyenne_volumique**

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

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

Usage:

moyenne_volumique {
 nom_pb *str*
 nom_domaine *str*
 noms_champs *n word1 word2 ... wordn*
 [**nom_fichier_post** *str*]
 [**format_post** *str*]

```

[ localisation str into ['elem', 'som']]
fonction_filtre bloc_lecture
}
where

```

- **nom_pb** *str*: name of the problem where the source fields will be searched.
- **nom_domaine** *str*: name of the destination domain (for example, it can be a coarser mesh, but for optimal performance in parallel, the domain should be split with the same algorithm as the computation mesh, eg, same *tranche* parameters for example)
- **noms_champs** *n word1 word2 ... wordn*: name of the source fields (these fields must be accessible from the 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 : lara)
- **localisation** *str into ['elem', 'som']*: indicates where the convolution product should be computed: either on the elements or on the nodes of the destination domain.
- **fonction_filtre** *bloc_lecture* (2.40): 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.

2.54 nettoiepasnoeuds

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

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

Usage:

nettoiepasnoeuds domain_name

where

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

2.55 option_vdf

Description: Class of VDF options.

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

Usage:

```
option_vdf {  
    [ traitement_coins str into ['oui', 'non']]  
    [ p_imposee_aux_faces str into ['oui', 'non']]  
}
```

where

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

2.56 orientefacesbord

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

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

Usage:

```
orientefacesbord domain_name  
where
```

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

2.57 partition

Synonymous: **decouper**

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

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

Usage:

```
partition domaine bloc_decouper  
where
```

- **domaine** *str*: Name of the domain to be cut.
- **bloc_decouper** *bloc_decouper* ([2.57](#)): Description how to cut a domain.

2.58 bloc_decouper

Description: Auxiliary class to cut a domain.

See also: [objet_lecture \(34\)](#)

Usage:

```
{
```

```

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

```

- **Partition_tool**partitionneur *partitionneur_deriv* (25): Defines the partitionning algorithm (the effective C++ object used is 'Partitionneur_ALGORITHM_NAME').
- **larg_joint** *int*: This keyword specifies the thickness of the virtual ghost zone (data known by one processor though not owned by it). The default value is 1 and is generally correct for all algorithms except the QUICK convection scheme that require a thickness of 2. Since the 1.5.5 version, the VEF discretization imply also a thickness of 2 (except VEF P0). Any non-zero positive value can be used, but the amount of data to store and exchange between processors grows quickly with the thickness.
- **zones_namelnom_zones** *str*: Name of the files containing the different partition of the domain. The files will be :
name_0001.Zones
name_0002.Zones
...
name_000n.Zones. If this keyword is not specified, the geometry is not written on disc (you might just want to generate a 'ecrire_decoupage' or 'ecrire_lata').
- **ecrire_decoupage** *str*: After having called the partitionning algorithm, the resulting partition is written on disc in the specified filename. See also partitionneur Fichier_Decoupage. This keyword is useful to change the partition numbers (for example, to do manually the task of the keyword Echange_domcut): first, you write the partition into a file with the option *ecrire_decoupage*. This file contains the zone number for each element's mesh. Then you can easily permute zone numbers in this file. Then read the new partition to create the .Zones files with the Fichier_Decoupage keyword.
- **ecrire_lata** *str*
- **nb_parts_tot** *int*: Keyword to generates N .Zone files, instead of the default number M obtained after the partitionning algorithm. N must be greater or equal to M. This option might be used to perform coupled parallel computations. Supplemental empty zones from M to N-1 are created. This keyword is used when you want to run a parallel calculation on several domains with for example, 2 processors on a first domain and 10 on the second domain because the first domain is very small compare to second one. You will write Nb_parts 2 and Nb_parts_tot 10 for the first domain and Nb_parts 10 for the second domain.
- **formatte** : Optional keyword to have formatted format for .Zones files. By default, it is binary format.
- **periodique** *n word1 word2 ... wordn*: N BOUNDARY_NAME_1 BOUNDARY_NAME_2 ... : N is the number of boundary names given. Periodic boundaries must be declared by this method. The 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.

2.59 pilote_icoco

Description: not_set

See also: interpret (2)

Usage:

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

where

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

2.60 porosites

Description: To define the volume porosity and surface porosity that are uniform in every direction in space on a sub-area.

Porosity was only usable in VDF discretization, and now available for VEF P1NC/P0.

Observations :

- Surface porosity values must be given in every direction in space (set this value to 1 if there is no porosity),
 - Prior to defining porosity, the problem must have been discretized.
- Can 't be used in VEF discretization, use Porosites_champ instead.

See also: interpret (2)

Usage:

```
porosites pb sous_zone bloc  
where
```

- **pb** *str*: Name of the problem to which the sub-area is attached.
- **sous_zone** *str*: Name of the sub-area to which porosity are allocated.
- **bloc** *bloc_lecture_poro* (2.60): Surface and volume porosity values.

2.61 bloc_lecture_poro

Description: Surface and volume porosity values.

See also: objet_lecture (34)

Usage:

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

where

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

2.62 porosites_champ

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

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

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

Usage:

porosites_champ pb ch

where

- **pb** *str*: Name of the problem to which the sub-area is attached.
- **ch** *champ_base* (16): field used to define the porosity field

2.63 postraiter_domaine

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

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

Usage:

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

where

- **format** *str* into ['lml', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med']: File format.
- **filefichier** *str*: The file name can be changed with the fichier option.
- **domaine** *str*: Name of domain
- **domaines** *bloc_lecture* (2.40): Names of domains : { name1 name2 }
- **joints_non_postraites** *int* into [0, 1]: The joints_non_postraites (1 by default) will not write the boundaries between the partitioned mesh.
- **binaire** *int* into [0, 1]: Binary (binaire 1) or ASCII (binaire 0) may be used. By default, it is 0 for LATA and only ASCII is available for LML and only binary is available for MED.
- **ecrire_frontiere** *int* into [0, 1]: This option will write (if set to 1, the default) or not (if set to 0) the boundaries as fields into the file (it is useful to not add the boundaries when writing a domain extracted from another domain)

2.64 precisiongeom

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

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

Usage:

precisiongeom precision

where

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

2.65 raffiner_anisotrope

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

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

Usage:

raffiner_anisotrope domain_name

where

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

2.66 raffiner_isotrope

Synonymous: **raffiner_simplexes**

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

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

Usage:

raffiner_isotrope domain_name

where

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

2.67 read

Synonymous: **lire**

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

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

Usage:

read a_object bloc

where

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

2.68 read_file

Synonymous: **lire_fichier**

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

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

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

See also: interpret (2) read_unsupported_ascii_file_from_icem (2.70) read_file_binary (2.68)

Usage:

read_file name_obj filename

where

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

2.69 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 (2.67)

Usage:

read_file_binary name_obj filename

where

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

2.70 lire_tgrid

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

See also: interpret (2)

Usage:

lire_tgrid dom filename

where

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

2.71 read_unsupported_ascii_file_from_icem

Description: not_set

See also: read_file (2.67)

Usage:

read_unsupported_ascii_file_from_icem **name_obj** **filename**

where

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

2.72 read_med

Synonymous: **lire_med**

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

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

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

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

Lire_Med

Read_file **domain_name_ssz.geo** ;

During the parallel calculation, you will include something:

Scatter { ... }

Read_file **domain_name_ssz_par.geo** ;

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

Usage:

read_med [**vef**] [**family_names_from_group_names**] [**short_family_names**] **nom_dom** **nom_dom_med** **file**

where

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

2.73 orienter_simplexes

Synonymous: **rectify_mesh**

Description: Keyword to raffine a mesh

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

Usage:

orienter_simplexes domain_name

where

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

2.74 redresser_hexaedres_vdf

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

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

Usage:

redresser_hexaedres_vdf domain_name

where

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

2.75 regroupebord

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

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

Usage:

regroupebord domaine new_bord bords

where

- **domaine** *str*: Name of domain
- **new_bord** *str*: Name of the new boundary
- **bords** *bloc_lecture (2.40)*: { Bound1 Bound2 }

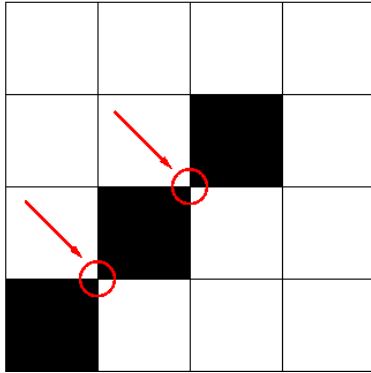
2.76 remove_elem

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

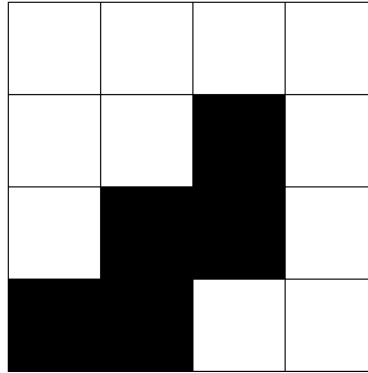
Remove_elem dom { fonction $0.2 * 0.2 - (x - 0.5)^2 - (y - 0.5)^2 > 0$ }

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

UNCORRECT – 2 SINGULAR NODES



CORRECT



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

Usage:

remove_elem **domaine** **bloc**

where

- **domaine** *str*: Name of domain
- **bloc** *remove_elem_bloc* ([2.76](#))

2.77 remove_elem_bloc

Description: not_set

See also: [objet_lecture \(34\)](#)

Usage:

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

where

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

2.78 remove_invalid_internal_boundaries

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

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

Usage:

remove_invalid_internal_boundaries **domain_name**

where

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

2.79 reordonner_faces_periodiques

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

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

Usage:

reordonner_faces_periodiques **domaine** **nom_bord_perio**

where

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

2.80 reorienter_tetraedres

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

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

Usage:

reorienter_tetraedres **domain_name**

where

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

2.81 reorienter_triangles

Description: not_set

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

Usage:

reorienter_triangles **domain_name**

where

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

2.82 reordonner

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

Lire_Fichier dom fichier.geom

Reordonner dom

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

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

Usage:

reordonner domain_name

where

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

2.83 rotation

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

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

Usage:

rotation domain_name dir coord1 coord2 angle

where

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

2.84 scatter

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

See also: interpret ([2](#)) scattermed ([2.85](#)) scatterformatte ([2.84](#))

Usage:

scatter file domaine

where

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

2.85 scatterformatte

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

See also: scatter ([2.83](#))

Usage:

scatterformatte file domaine

where

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

2.86 scattermed

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

See also: scatter ([2.83](#))

Usage:

scattermed file domaine

where

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

2.87 solve

Synonymous: **resoudre**

Description: Interpreter to start calculation with TRUST.

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

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

Usage:

solve pb

where

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

2.88 supprime_bord

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

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

Usage:

supprime_bord domaine bords

where

- **domaine** *str*: Name of domain
- **bords** *list_nom* ([2.88](#)): { Boundary_name1 Boundary_name2 }

2.89 list_nom

Description: List of name.

See also: listobj ([33.3](#))

Usage:

{ object1 object2 }

list of *nom_anonyme* ([24](#))

2.90 system

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

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

Usage:

system cmd

where

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

2.91 test_solveur

Description: To test several solvers

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

Usage:

test_solveur {

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

}

where

- **fichier_secmem** *str*: Filename containing the second member B
- **fichier_matrice** *str*: Filename containing the matrix A
- **fichier_solution** *str*: Filename containing the solution x
- **nb_test** *int*: Number of tests to measure the time resolution (one preconditionnement)
- **impr** : To print the convergence solver
- **solveur** *solveur_sys_base* (9.11): To specify a solver
- **fichier_solveur** *str*: To specify a file containing a list of solvers
- **genere_fichier_solveur** *float*: To create a file of the solver with a threshold convergence
- **seuil_verification** *float*: Check if the solution satisfy $\|Ax-B\| < \text{precision}$
- **pas_de_solution_initiale** : Resolution isn't initialized with the solution x
- **ascii** : Ascii files

2.92 testeur

Description: not_set

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

Usage:

testeur data

where

- **data** *bloc_lecture* (2.40)

2.93 testeur_medcoupling

Description: not_set

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

Usage:

testeur_medcoupling pb_name field_name

where

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

2.94 tetraedriser

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

See also: [interprete \(2\)](#) [tetraedriser_homogene \(2.94\)](#) [tetraedriser_homogene_fin \(2.96\)](#) [tetraedriser_homogene_compact \(2.95\)](#) [tetraedriser_par_prisme \(2.97\)](#)

Usage:

tetraedriser domain_name

where

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

2.95 tetraedriser_homogene

Description: Use the Tetraedriser_homogene (Homogeneous_Tetrahedralisation) interpreter in VEF discretisation to mesh a block in tetrahedrals. Each block hexahedral is no longer divided into 5 tetrahedrals (keyword Tetraedriser (Tetrahedralise)), it is now broken down into 40 tetrahedrals. Thus a block defined with 11 nodes in each X, Y, Z direction will contain $10*10*10*40=40,000$ tetrahedrals. This also allows problems in the mesh corners with the P1NC/P1iso/P1bulle or P1/P1 discretisation items to be avoided.

See also: [tetraedriser \(2.93\)](#)

Usage:

tetraedriser_homogene domain_name

where

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

2.96 tetraedriser_homogene_compact

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

See also: tetraedriser ([2.93](#))

Usage:

tetraedriser_homogene_compact domain_name
where

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

2.97 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 PrePIB),
- a better isotropy of elements than with Tetraedriser_homogene_compact,
- a better alignment of summits (this could have a benefit effect on calculation near walls since first elements in contact with it are all contained in the same constant thickness and ii/ by the way, a 3D cartesian grid based on summits can be engendered and used to realise spectral analysis in HIT for instance).

See also: tetraedriser ([2.93](#))

Usage:

tetraedriser_homogene_fin domain_name
where

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

2.98 tetraedriser_par_prisme

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

See also: tetraedriser ([2.93](#))

Usage:

tetraedriser_par_prisme domain_name
where

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

2.99 transformer

Description: Keyword to transform the coordinates of the geometry.

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

See also: interpret (2)

Usage:

transformer domain_name formule

where

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

Function_forz

2.100 trianguler

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

See also: interpret (2) trianguler_h (2.101) trianguler_fin (2.100)

Usage:

trianguler domain_name

where

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

2.101 trianguler_fin

Description: Trianguler_fin is the recommended option to triangulate rectangles.

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

See also: trianguler (2.99)

Usage:

trianguler_fin domain_name

where

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

2.102 trianguler_h

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

See also: `trianguler` ([2.99](#))

Usage:

trianguler_h domain_name

where

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

2.103 `verifier_qualite_raffinements`

Description: `not_set`

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

Usage:

verifier_qualite_raffinements domain_names

where

- **domain_names** *vect_nom* ([2.103](#))

2.104 `vect_nom`

Description: Vect of name.

See also: `listobj` ([33.3](#))

Usage:

`n object1 object2`

list of *nom_anonyme* ([24](#))

2.105 `verifier_simplexes`

Description: Keyword to refine a simplexes

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

Usage:

verifier_simplexes domain_name

where

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

2.106 `verifiercoin`

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

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

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

Usage:

verifiercoin dom

where

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

2.107 **ecrire**

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

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

Usage:

ecrire name_obj

where

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

2.108 **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](#) (2) [ecrire_fichier_formatte](#) (2.22)

Usage:

ecrire_fichier_bin name_obj filename

where

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

2.109 **ecrire_med**

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

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

Usage:

ecrire_med nom_dom file

where

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

3 pb_gen_base

Description: Basic class for problems.

See also: objet_u (35) Pb_base (3) probleme_couple (3.6) pbc_med (3.35) pb_mg (3.20)

Usage:

3.1 Pb_base

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

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

See also: pb_gen_base (3) pb_thermohydraulique (3.23) pb_hydraulique (3.14) pb_hydraulique_turbulent (3.19) pb_thermohydraulique_turbulent (3.31) pb_conduction (3.12) pb_thermohydraulique_qc (3.28) pb_thermohydraulique_turbulent_qc (3.32) pb_hydraulique_concentration (3.15) pb_hydraulique_concentration_turbulent (3.17) pb_thermohydraulique_concentration (3.24) pb_thermohydraulique_concentration_turbulent (3.26) pb_avec_passif (3.10.2) pb_post (3.22) problem_read_generic (3.37.1) modele_rayo_semi_transp (3.8) pb_phase_field (3.21)

Usage:

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

where

- **Post_processing|postraitement** *corps_postraitement* (3.1): One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28): List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1): This
- **liste_postraitements** *liste_post* (3.4.4): This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3): Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3): The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3): Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors,

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

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

3.2 corps_postraitement

Description: not_set

See also: post_processing (3.4.2)

Usage:

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

where

- **definition_champs** *definition_champs* (3.2) for inheritance: Keyword to create new or more complex field for advanced postprocessing.
- **Probeslsondes** *sondes* (3.2.2) for inheritance: Probe.
- **domaine** *str* for inheritance: This optional parameter specifies the domain on which the data should be interpolated before it is written in the output file. The default is to write the data on the domain of the current problem (no interpolation).
- **format** *str* into ['lml', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med'] for inheritance: This optional parameter specifies the format of the output file. The basename used for the output file is the basename of the data file. For the fmt parameter, choices are lml, lata, or meshtv. A short description of each format can be found below. The default value is lml.
- **fieldslchamps** *champs_posts* (3.2.16) for inheritance: Field's write mode.
- **statistiques** *stats_posts* (3.2.19) for inheritance: Statistics between two points fixed : start of integration time and end of integration time.
- **fichier** *str* for inheritance: Name of file.
- **statistiques_en_serie** *stats_serie_posts* (3.2.27) for inheritance: Statistics between two points not fixed : on period of integration.
- **interfaces** *champs_posts* (3.2.16) for inheritance: Keyword to read all the characteristics of the interfaces. Different kind of interfaces exist as well as different interface initialisations.

3.2.1 definition_champs

Description: List of definition champ

See also: listobj (33.3)

Usage:

{ object1 object2 }
list of *definition_champ* (3.2.1)

3.2.2 definition_champ

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

See also: *objet_lecture* (34)

Usage:

name champ_generique
where

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

3.2.3 sondes

Description: List of probes.

See also: *listobj* (33.3)

Usage:

{ object1 object2 }
list of *sonde* (3.2.3)

3.2.4 sonde

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

See also: *objet_lecture* (34)

Usage:

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

- **nom_sonde** *str*: Name of the file in which the values taken over time will be saved. The complete file name is *nom_sonde.son*.
- **special** *str into* ['*chsom*', '*nodes*', '*grav*', '*som*']: Option to change the positions of the probes. Several options are available:
 - grav* : each probe is moved to the nearest cell center of the mesh;
 - som* : each probe is moved to the nearest vertex of the mesh
 - nodes* : each probe is moved to the nearest face center of the mesh;
 - chsom* : only available for P1NC sampled field. The values of the probes are calculated according to P1-Conform corresponding field.
- **nom_inco** *str*: Name of the sampled field.
- **mperiode** *str into* ['*periode*']: Keyword to set the sampled field measurement frequency.
- **prd** *float*: Period value. Every *prd* seconds, the field value calculated at the previous time step is written to the *nom_sonde.son* file.
- **type** *sonde_base* (3.2.4): Type of probe.

3.2.5 sonde_base

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

See also: objet_lecture (34) points (3.2.5) numero_elem_sur_maitre (3.2.9) position_like (3.2.10) segment (3.2.11) plan (3.2.12) volume (3.2.13) circle (3.2.14) circle_3 (3.2.15)

Usage:

sonde_base

3.2.6 points

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

See also: sonde_base (3.2.4) point (3.2.7) segmentpoints (3.2.8)

Usage:

points points

where

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

3.2.7 listpoints

Description: Points.

See also: listobj (33.3)

Usage:

n object1 object2

list of *un_point* (2.10.2)

3.2.8 point

Description: Point as class-daughter of Points.

See also: points (3.2.5)

Usage:

point points

where

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

3.2.9 segmentpoints

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

See also: points (3.2.5)

Usage:

segmentpoints points

where

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

3.2.10 numero_elem_sur_maitre

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

See also: sonde_base (3.2.4)

Usage:

numero_elem_sur_maitre numero

where

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

3.2.11 position_like

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

See also: sonde_base (3.2.4)

Usage:

position_like autre_sonde

where

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

3.2.12 segment

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

See also: sonde_base (3.2.4)

Usage:

segment nbr point_deb point_fin

where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* (2.10.2): First outer probe segment point.
- **point_fin** *un_point* (2.10.2): Second outer probe segment point.

3.2.13 plan

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

See also: sonde_base (3.2.4)

Usage:

plan nbr nbr2 point_deb point_fin point_fin_2

where

- **nbr** *int*: Number of probes in the first direction.
- **nbr2** *int*: Number of probes in the second direction.
- **point_deb** *un_point* (2.10.2): First point defining the angle. This angle should be positive.
- **point_fin** *un_point* (2.10.2): Second point defining the angle. This angle should be positive.
- **point_fin_2** *un_point* (2.10.2): Third point defining the angle. This angle should be positive.

3.2.14 volume

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

See also: `sonde_base` (3.2.4)

Usage:

volume nbr nbr2 nbr3 point_deb point_fin point_fin_2 point_fin_3

where

- **nbr** *int*: Number of probes in the first direction.
- **nbr2** *int*: Number of probes in the second direction.
- **nbr3** *int*: Number of probes in the third direction.
- **point_deb** *un_point* (2.10.2): Point of origin.
- **point_fin** *un_point* (2.10.2): Point defining the first direction (from point of origin).
- **point_fin_2** *un_point* (2.10.2): Point defining the second direction (from point of origin).
- **point_fin_3** *un_point* (2.10.2): Point defining the third direction (from point of origin).

3.2.15 circle

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

See also: `sonde_base` (3.2.4)

Usage:

circle nbr point_deb [direction] radius theta1 theta2

where

- **nbr** *int*: Number of probes between theta1 and theta2 (angles given in degrees).
- **point_deb** *un_point* (2.10.2): Center of the circle.
- **direction** *int into [0, 1, 2]*: Axis normal to the circle plane (0:x axis, 1:y axis, 2:z axis).
- **radius** *float*: Radius of the circle.
- **theta1** *float*: First angle.
- **theta2** *float*: Second angle.

3.2.16 circle_3

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

See also: `sonde_base` (3.2.4)

Usage:

circle_3 nbr point_deb direction radius theta1 theta2

where

- **nbr** *int*: Number of probes between theta1 and theta2 (angles given in degrees).

- **point_deb** *un_point* (2.10.2): Center of the circle.
- **direction** *int* into [0, 1, 2]: Axis normal to the circle plane (0:x axis, 1:y axis, 2:z axis).
- **radius** *float*: Radius of the circle.
- **theta1** *float*: First angle.
- **theta2** *float*: Second angle.

3.2.17 champs_posts

Description: Field's write mode.

See also: *objet_lecture* (34)

Usage:

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

- **format** *str* into ['binaire', 'formate']: Type of file.
- **mot** *str* into ['dt_post', 'nb_pas_dt_post']: Keyword to set the kind of the field's write frequency. Either a time period or a time step period.
- **period** *str*: Value of the period.
- **fields|champs** *champs_a_post* (3.2.17): Post-processed fields.

3.2.18 champs_a_post

Description: Fields to be post-processed.

See also: *listobj* (33.3)

Usage:

{ object1 object2 }
list of *champ_a_post* (3.2.18)

3.2.19 champ_a_post

Description: Field to be post-processed.

See also: *objet_lecture* (34)

Usage:

champ [**localisation**]
where

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

3.2.20 stats_posts

Description: Field's write mode.

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

dts: frequency value.

t_deb value: Start of integration time

t_fin value: End of integration time

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

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

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

Example:

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

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

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

See also: objet_lecture (34)

Usage:

mot period fields|champs

where

- **mot** *str* into ['dt_post', 'nb_pas_dt_post']: Keyword to set the kind of the field's write frequency. Either a time period or a time step period.
- **period** *str*: Value of the period.
- **fields|champs** *list_stat_post* (3.2.20): Post-processed fields.

3.2.21 list_stat_post

Description: Post-processing for statistics

See also: listobj ([33.3](#))

Usage:

{ object1 object2 }

list of *stat_post_deriv* ([3.2.21](#))

3.2.22 stat_post_deriv

Description: not_set

See also: objet_lecture ([34](#)) t_deb ([3.2.22](#)) t_fin ([3.2.23](#)) moyenne ([3.2.24](#)) ecart_type ([3.2.25](#)) correlation ([3.2.26](#))

Usage:

stat_post_deriv

3.2.23 t_deb

Description: not_set

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

Usage:

t_deb val

where

- **val** *float*

3.2.24 t_fin

Description: not_set

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

Usage:

t_fin val

where

- **val** *float*

3.2.25 moyenne

Synonymous: **champ_post_statistiques_moyenne**

Description: not_set

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

Usage:

moyenne field [localisation]

where

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

3.2.26 ecart_type

Synonymous: **champ_post_statistiques_ecart_type**

Description: not_set

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

Usage:

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

where

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

3.2.27 correlation

Synonymous: **champ_post_statistiques_correlation**

Description: not_set

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

Usage:

correlation **first_field** **second_field** [**localisation**]

where

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

3.2.28 stats_serie_posts

Description: Post-processing for statistics.

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

dt_integr value : Period of integration and write period.

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

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

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

Example:

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

Will calculate and write every dtst seconds the mean value:

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

See also: objet_lecture (34)

Usage:

mot dt_integr stat

where

- **mot** *str* into [*'dt_integr'*]: Keyword is used to set the statistics period of integration and write period.
- **dt_integr** *float*: Average on dt_integr time interval is post-processed every dt_integr seconds.
- **stat** *list_stat_post* (3.2.20)

3.3 post_processings

Synonymous: **postraitements**

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

See also: listobj (33.3)

Usage:

{ object1 object2 }

list of *un_postraitement* (3.3)

3.3.1 un_postraitement

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

See also: objet_lecture (34)

Usage:

nom post

where

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

3.4 liste_post_ok

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

See also: listobj (33.3)

Usage:

{ object1 object2 }

list of *nom_postraitement* (3.4)

3.4.1 nom_postraitement

Description:

See also: `objet_lecture` (34)

Usage:

nom post

where

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

3.4.2 postraitement_base

Description: `not_set`

See also: `objet_lecture` (34) `post_processing` (3.4.2) `postraitement_ft_lata` (3.4.3)

Usage:

3.4.3 post_processing

Synonymous: **postraitement**

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

See also: `postraitement_base` (3.4.1) `corps_postraitement` (3.1)

Usage:

post_processing {

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

}

where

- **definition_champs** *definition_champs* (3.2): Keyword to create new or more complex field for advanced postprocessing.
- **Probeslsondes** *sondes* (3.2.2): Probe.
- **domaine** *str*: This optional parameter specifies the domain on which the data should be interpolated before it is written in the output file. The default is to write the data on the domain of the current problem (no interpolation).
- **format** *str* into ['lml', 'meshtv', 'lata', 'lata_v1', 'lata_v2', 'med']: This optional parameter specifies the format of the output file. The basename used for the output file is the basename of the data file. For the `fmt` parameter, choices are `lml`, `lata`, or `meshtv`. A short description of each format can be found below. The default value is `lml`.

- **fieldslchamps** *champs_posts* (3.2.16): Field's write mode.
- **statistiques** *stats_posts* (3.2.19): Statistics between two points fixed : start of integration time and end of integration time.
- **fichier** *str*: Name of file.
- **statistiques_en_serie** *stats_serie_posts* (3.2.27): Statistics between two points not fixed : on period of integration.
- **interfaces** *champs_posts* (3.2.16): Keyword to read all the characteristics of the interfaces. Different kind of interfaces exist as well as different interface initialisations.

3.4.4 postraitement_ft_lata

Description: not_set

See also: postraitement_base (3.4.1)

Usage:

postraitement_ft_lata **bloc**
where

- **bloc** *str*

3.5 liste_post

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

See also: listobj (33.3)

Usage:

{ object1 object2 }
list of *un_postraitement_spec* (3.5)

3.5.1 un_postraitement_spec

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

See also: objet_lecture (34)

Usage:

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

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

3.5.2 type_un_post

Description: not_set

See also: objet_lecture (34)

Usage:

type post
where

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

3.5.3 type_postraitement_ft_lata

Description: not_set

See also: objet_lecture (34)

Usage:

type nom bloc

where

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

3.6 format_file

Description: File formatted.

See also: objet_lecture (34)

Usage:

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

where

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

3.7 probleme_couple

Description: This instruction causes a `probleme_couple` type object to be created. This type of object has an associated problem list, that is, the coupling of *n* problems among them may be processed. Coupling between these problems is carried out explicitly via conditions at particular contact limits. Each problem may be associated either with the `Associer` keyword or with the `Lire/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`

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

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

Each problem is resolved in a domain.

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

See also: `pb_gen_base` (3) `pb_couple_rayonnement` (3.38) `pb_couple_rayo_semi_transp` (3.13)

Usage:

probleme_couple *obj* Lire *obj* {


```
[ groupes list_list_nom]
```

```
}
```

where

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

3.8 list_list_nom

Description: pour les groupes

See also: listobj (33.3)

Usage:

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

list of *list_un_pb* (33) separated with ,

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

See also: Pb_base (3)

Usage:

```
modele_rayo_semi_transp obj Lire obj {
```

```
  [ eq_rayo_semi_transp eq_rayo_semi_transp]
  [ Post_processing|postraitemnt corps_postraitemnt]
  [ 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* (3.9): Irradiancy G equation. Radiative flux equals $-\text{grad}(G)/3/\kappa$.
- **Post_processing|postraitemnt** *corps_postraitemnt* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.

- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.10 eq_rayo_semi_transp

Description: Irradiancy equation.

See also: objet_lecture (34)

Usage:

```
{
    solveur solveur_sys_base
    [ boundary_conditions|conditions_limités condlims]
```

```
}
```

where

- **solveur** *solveur_sys_base* (9.11): Solver of the irradiancy equation.
- **boundary_conditions|conditions_limités condlims** (3.10): Boundary conditions.

3.10.1 condlims

Description: Boundary conditions.

See also: listobj (33.3)

Usage:

```
{ object1 object2 ... }
list of condlimlu (3.10.1)
```

3.10.2 condlimlu

Description: Boundary condition specified.

See also: objet_lecture (34)

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).

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

See also: Pb_base (3) pb_thermohydraulique_concentration_turbulent_scalaires_passifs (3.27) pb_thermohydraulique_concentration_scalaires_passifs (3.25) pb_thermohydraulique_turbulent_scalaires_passifs (3.34) pb_thermohydraulique_scalaires_passifs (3.30) pb_hydraulique_concentration_turbulent_scalaires_passifs (3.18) pb_hydraulique_concentration_scalaires_passifs (3.16) pb_thermohydraulique_qc_fraction_massique (3.29) pb_thermohydraulique_turbulent_qc_fraction_massique (3.33)

Usage:

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

where

- **equations_scalaires_passifs** *listeqn* (3.11): Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file

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

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

3.12 listeqn

Description: List of equations.

See also: listobj (33.3)

Usage:

{ object1 object2 }

list of *eqn_base* (4.20)

3.13 pb_conduction

Description: Resolution of the heat equation.

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

See also: Pb_base (3)

Usage:

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

where

- **conduction** *conduction* (4): Heat equation.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.

- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.14 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 (3.6)

Usage:

pb_couple_rayo_semi_transp obj Lire obj {

[**groupes** *list_list_nom*]

}

where

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

3.15 pb_hydraulique

Description: Resolution of the NAVIER STOKES equations.

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

See also: Pb_base (3)

Usage:

pb_hydraulique obj Lire obj {

navier_stokes_standard *navier_stokes_standard*

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

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

```

[ liste_de_postraitements liste_post_ok]
[ liste_postraitements liste_post]
[ sauvegarde format_file]
[ sauvegarde_simple format_file]
[ reprise format_file]
[ resume_last_time format_file]
}
where

```

- **navier_stokes_standard** *navier_stokes_standard* (4.28): NAVIER STOKES equations.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.16 pb_hydraulique_concentration

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

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

See also: Pb_base (3)

Usage:

```

pb_hydraulique_concentration obj Lire obj {
  [ navier_stokes_standard navier_stokes_standard]
  [ convection_diffusion_concentration convection_diffusion_concentration]
  [ Post_processing|postraitement corps_postraitement]
  [ Post_processings|postraitements post_processings]
  [ liste_de_postraitements liste_post_ok]
  [ liste_postraitements liste_post]
}

```

```

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

```

- **navier_stokes_standard** *navier_stokes_standard* (4.28): NAVIER STOKES equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (4.9): Constituent transportation vectorial equation (concentration diffusion convection).
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.17 pb_hydraulique_concentration_scalaires_passifs

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

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

See also: pb_avec_passif (3.10.2)

Usage:

```

pb_hydraulique_concentration_scalaires_passifs obj Lire obj {
    [ navier_stokes_standard navier_stokes_standard]
    [ convection_diffusion_concentration convection_diffusion_concentration]
    equations_scalaires_passifs listeqn
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
}

```

```

[ liste_de_postraitements liste_post_ok]
[ liste_postraitements liste_post]
[ sauvegarde format_file]
[ sauvegarde_simple format_file]
[ reprise format_file]
[ resume_last_time format_file]
}
where

```

- **navier_stokes_standard** *navier_stokes_standard* (4.28): NAVIER STOKES equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (4.9): Constituent transportation equations (concentration diffusion convection).
- **equations_scalaires_passifs** *listeqn* (3.11) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \neq P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.18 pb_hydraulique_concentration_turbulent

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

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

See also: Pb_base (3)

Usage:


```

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

```

where

- **navier_stokes_turbulent** *navier_stokes_turbulent* (4.29): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (4.11): Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.19 pb_hydraulique_concentration_turbulent_scalaires_passifs

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

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

See also: `pb_avec_passif` (3.10.2)

Usage:

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

where

- **navier_stokes_turbulent** *navier_stokes_turbulent* (4.29): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (4.11): Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (3.11) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved

files).

3.20 pb_hydraulique_turbulent

Description: Resolution of NAVIER STOKES equations with turbulence modelling.

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

See also: Pb_base (3)

Usage:

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

```
}
```

where

- **navier_stokes_turbulent** *navier_stokes_turbulent* (4.29): NAVIER STOKES equations as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.21 pb_mg

Description: Multi-grid problem.

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

See also: `pb_gen_base` (3)

Usage:

pb_mg

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

See also: `Pb_base` (3)

Usage:

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

where

- **navier_stokes_phase_field** *navier_stokes_phase_field* (4.26.13): Navier Stokes equation for the Phase Field problem.
- **convection_diffusion_phase_field** *convection_diffusion_phase_field* (4.14): Cahn-Hilliard equation of the Phase Field problem. The unknown of this equation is the concentration C.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.

- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.23 pb_post

Description: not_set

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

See also: Pb_base (3)

Usage:

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

where

- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the

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

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

3.24 pb_thermohydraulique

Description: Resolution of thermohydraulic problem.

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

See also: Pb_base (3)

Usage:

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

where

- **navier_stokes_standard** *navier_stokes_standard* (4.28): NAVIER STOKES equations.
- **convection_diffusion_temperature** *convection_diffusion_temperature* (4.15): Energy equation (temperature diffusion convection).
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \geq P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the

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

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

3.25 pb_thermohydraulique_concentration

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

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

See also: Pb_base (3)

Usage:

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

where

- **navier_stokes_standard** *navier_stokes_standard* (4.28): NAVIER STOKES equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (4.9): Constituent transportation equations (concentration diffusion convection).
- **convection_diffusion_temperature** *convection_diffusion_temperature* (4.15): Energy equation (temperature diffusion convection).
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file

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

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

3.26 pb_thermohydraulique_concentration_scalaires_passifs

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

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

See also: pb_avec_passif (3.10.2)

Usage:

pb_thermohydraulique_concentration_scalaires_passifs obj Lire obj {

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

}

where

- **navier_stokes_standard** *navier_stokes_standard* (4.28): NAVIER STOKES equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (4.9): Constituent transportation equations (concentration diffusion convection).
- **convection_diffusion_temperature** *convection_diffusion_temperature* (4.15): Energy equations (temperature diffusion convection).
- **equations_scalaires_passifs** *listeqn* (3.11) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_masseN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and

in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.

- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.27 pb_thermohydraulique_concentration_turbulent

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

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

See also: Pb_base (3)

Usage:

```
pb_thermohydraulique_concentration_turbulent obj Lire obj {
    [ navier_stokes_turbulent navier_stokes_turbulent]
    [ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent]
    [ convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent]
    [ Post_processing|postraitements corps_postraitements]
    [ Post_processing|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file]
    [ sauvegarde_simple format_file]
    [ reprise format_file]
    [ resume_last_time format_file]
}
where
```

- **navier_stokes_turbulent** *navier_stokes_turbulent* (4.29): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (4.11): Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (4.19): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.

- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.28 pb_thermohydraulique_concentration_turbulent_scalaires_passifs

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

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

See also: pb_avec_passif (3.10.2)

Usage:

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

where

- **navier_stokes_turbulent** *navier_stokes_turbulent* (4.29): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (4.11): Constituent transportation equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (4.19): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (3.11) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.29 pb_thermohydraulique_qc

Description: Resolution of thermohydraulic problem under small Mach number.

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

masse_volumique : density

enthalpie : enthalpy

pression : reduced pressure

pression_tot : total pressure.

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

See also: Pb_base (3)

Usage:

```

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

```

where

- **navier_stokes_qc** *navier_stokes_qc* (4.27): NAVIER STOKES equations under smal Mach number.
- **convection_diffusion_chaleur_qc** *convection_diffusion_chaleur_qc* (4.6.2): Energy equation under smal Mach number.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.30 pb_thermohydraulique_qc_fraction_massique

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

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

See also: `pb_avec_passif` (3.10.2)

Usage:

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

where

- **navier_stokes_qc** *navier_stokes_qc* (4.27): NAVIER STOKES equations under small Mach number.
- **convection_diffusion_chaleur_qc** *convection_diffusion_chaleur_qc* (4.6.2): Energy equation under small Mach number.
- **equations_scalaires_passifs** *listeqn* (3.11) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.31 pb_thermohydraulique_scalaires_passifs

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

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

See also: pb_avec_passif (3.10.2)

Usage:

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

where

- **navier_stokes_standard** *navier_stokes_standard* (4.28): NAVIER STOKES equations.
- **convection_diffusion_temperature** *convection_diffusion_temperature* (4.15): Energy equations (temperature diffusion convection).
- **equations_scalaires_passifs** *listeqn* (3.11) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_masseN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.

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

3.32 pb_thermohydraulique_turbulent

Description: Resolution of thermohydraulic problem, with turbulence modelling.

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

See also: Pb_base (3)

Usage:

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

where

- **navier_stokes_turbulent** *navier_stokes_turbulent* (4.29): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (4.19): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the

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

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

3.33 pb_thermohydraulique_turbulent_qc

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

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

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

See also: Pb_base (3)

Usage:

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

where

- **navier_stokes_turbulent_qc** *navier_stokes_turbulent_qc* (4.30): NAVIER STOKES equations under small Mach number as well as the associated turbulence model equations.
- **convection_diffusion_chaleur_turbulent_qc** *convection_diffusion_chaleur_turbulent_qc* (4.8.23): Energy equation under small Mach number as well as the associated turbulence model equations.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on

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

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

3.34 pb_thermohydraulique_turbulent_qc_fraction_massique

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

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

See also: pb_avec_passif (3.10.2)

Usage:

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

where

- **navier_stokes_turbulent_qc** *navier_stokes_turbulent_qc* (4.30): NAVIER STOKES equations under small Mach number as well as the associated turbulence model equations.
- **convection_diffusion_chaleur_turbulent_qc** *convection_diffusion_chaleur_turbulent_qc* (4.8.23): Energy equation under small Mach number as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (3.11) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.

- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \neq P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.35 pb_thermohydraulique_turbulent_scalaires_passifs

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

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

See also: pb_avec_passif (3.10.2)

Usage:

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

- **navier_stokes_turbulent** *navier_stokes_turbulent* (4.29): NAVIER STOKES equations as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (4.19): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (3.11) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.

- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

3.36 pbc_med

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

See also: pb_gen_base (3)

Usage:

pbc_med list_info_med

where

- **list_info_med** *list_info_med* (3.36)

3.37 list_info_med

Description: not_set

See also: listobj (33.3)

Usage:

{ object1 , object2 }

list of *info_med* (3.37) separated with ,

3.37.1 info_med

Description: not_set

See also: `objet_lecture` (34)

Usage:

file_med **domaine** **pb_post**
where

- **file_med** *str*: Name of file med.
- **domaine** *str*: Name of domain.
- **pb_post** *pb_post* (3.22)

3.38 `problem_read_generic`

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

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

See also: `Pb_base` (3) `probleme_ft_disc_gen` (3.39)

Usage:

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

where

- **Post_processing|postraitement** *corps_postraitement* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is `lata` in order to use `OpenDX` to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory `lata` used in this example should be created before running the computation or the `lata` files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than `Sauvegarde` except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the `name_file` file (see the class `format_file`). If `format_reprise` is `xyz`, the `name_file` file should be the `.xyz` file created by the previous calculation. With this file, it is possible to restart a parallel calculation on

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

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

3.39 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 (3.6)

Usage:

pb_couple_rayonnement obj Lire obj {

[**groupes** *list_list_nom*]

}

where

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

3.40 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 predefined in the structure of the problem, each equation has to be distinctively associated with the problem with the Associer keyword.

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

See also: problem_read_generic (3.37.1)

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* (3.1) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (3.2.28) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (3.3.1) for inheritance: This
- **liste_postraitements** *liste_post* (3.4.4) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file* (3.5.3) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.
- **sauvegarde_simple** *format_file* (3.5.3) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be restarted, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file* (3.5.3) for inheritance: Keyword to restart a calculation based on the name_file file, restart the calculation at the last time found in the file (tinit is set to last time of saved files).

4 mor_eqn

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

See also: objet_u (35) eqn_base (4.20)

Usage:

4.1 conduction

Description: Heat equation.

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

See also: eqn_base (4.20)

Usage:

conduction obj Lire obj {

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

}
where

- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbyname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbyname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.2 bloc_diffusion

Description: not_set

See also: objet_lecture (34)

Usage:

aco [**operateur**] [**op_implicite**] **acof**

where

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

4.2.1 diffusion_deriv

Description: not_set

See also: objet_lecture (34) negligable (4.2.1) p1b (4.2.2) p1ncp1b (4.2.3) stab (4.2.4) standard (4.2.5) option (4.2.7)

Usage:

diffusion_deriv

4.2.2 **negligeable**

Description: the diffusivity will not taken in count

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

Usage:

negligeable

4.2.3 **p1b**

Description: not_set

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

Usage:

p1b

4.2.4 **p1ncp1b**

Description: not_set

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

Usage:

4.2.5 **stab**

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

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

Usage:

stab {

```
[ standard int]  
[ info int]  
[ new_jacobian int]  
[ nu int]  
[ nut int]  
[ nu_transp int]  
[ nut_transp int]
```

}

where

- **standard** *int*: to recover the same results as calculations made by standard laminar diffusion operator. However, no stabilization technique is used and calculations may be unstable when working with obtuse angle meshes (by default 0)
- **info** *int*: developer option to get the stabilizing ratio (by default 0)
- **new_jacobian** *int*: when implicit time schemes are used, this option defines a new jacobian that may be more suitable to get stationary solutions (by default 0)

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

4.2.6 standard

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

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

See also: diffusion_deriv (4.2)

Usage:

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

where

- **mot1** *str into* [*'defaut_bar'*]: equivalent to grad_Ubar 1 nu 1 nut 1 nu_transp 1 nut_transp 1 filtrer_resu 1
- **bloc_diffusion_standard** *bloc_diffusion_standard* (4.2.6)

4.2.7 bloc_diffusion_standard

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

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

filtrer_resu 1 allows to filter the resulting diffusive fluxes contribution.

See also: objet_lecture (34)

Usage:

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

where

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

- **mot6** *str* into ['grad_Ubar', 'nu', 'nut', 'nu_transp', 'nut_transp', 'filtrer_resu']
- **val6** *int* into [0, 1]

4.2.8 option

Description: not_set

See also: diffusion_deriv (4.2)

Usage:

option bloc_lecture

where

- **bloc_lecture** *bloc_lecture* (2.40)

4.2.9 op_implicite

Description: not_set

See also: objet_lecture (34)

Usage:

implicite mot solveur

where

- **implicite** *str* into ['implicite']
- **mot** *str* into ['solveur']
- **solveur** *solveur_sys_base* (9.11)

4.3 condinits

Description: Initial conditions.

See also: objet_lecture (34)

Usage:

aco condinit acof

where

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

4.3.1 condinit

Description: Initial condition.

See also: objet_lecture (34)

Usage:

nom ch

where

- **nom** *str*: Name of initial condition field.
- **ch** *champ_base* (16): Type field and the initial values.

4.4 sources

Description: The sources.

See also: `listobj` (33.3)

Usage:

{ object1 , object2 }

list of `source_base` (29) separated with ,

4.5 ecrire_fichier_xyz_valeur_param

Description: `not_set`

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

See also: `listobj` (33.3)

Usage:

n object1 , object2

list of `ecrire_fichier_xyz_valeur_item` (4.5) separated with ,

4.5.1 ecrire_fichier_xyz_valeur_item

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

The name of the files is `pb_name_field_name_time.dat`

Several `Ecrire_fichier_xyz_valeur` keywords may be written into an equation to write several fields. This kind of files may be read by `Champ_don_lu` or `Champ_front_lu` for example.

See also: `objet_lecture` (34)

Usage:

name **dt_ecrire_fic** [**bords**]

where

- **name** *str*: Name of the field to write (`Champ_Inc`, `Champ_Fonc` or a post-processed field).
- **dt_ecrire_fic** *float*: Time period for printing in the file.
- **bords** *bords_ecrire* (4.5.1): to post-process only on some boundaries

4.5.2 bords_ecrire

Description: `not_set`

See also: `objet_lecture` (34)

Usage:

chaîne **bords**

where

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

4.6 parametre_equation_base

Description: Basic class for parametre_equation

See also: objet_lecture (34) parametre_diffusion_implicite (4.6) parametre_implicite (4.6.1)

Usage:

4.6.1 parametre_diffusion_implicite

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

See also: parametre_equation_base (4.5.2)

Usage:

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

where

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

4.6.2 parametre_implicite

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

See also: parametre_equation_base (4.5.2)

Usage:

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

where

- **seuil_convergence_implicit** *float*: Keyword to change for this equation only the value of `seuil_convergence_implicit` used in the implicit scheme.
- **seuil_convergence_solveur** *float*: Keyword to change for this equation only the value of `seuil_convergence_solveur` used in the implicit scheme
- **solveur** *solveur_sys_base* (9.11): Keyword to change for this equation only the solver used in the implicit scheme
- **resolution_explicite** : To solve explicitly the equation whereas the scheme is an implicit scheme.
- **equation_non_resolue** : Keyword to specify that the equation is not solved.
- **equation_frequence_resolue** *str*: Keyword to specify that the equation is solved only every *n* time steps (*n* is an integer or given by a time-dependent function *f(t)*).

4.7 convection_diffusion_chaleur_qc

Description: Energy equation under small Mach number.

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

See also: `eqn_base` (4.20) `convection_diffusion_chaleur_turbulent_qc` (4.8.23)

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_limite condlims]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
[ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
```

}

where

- **mode_calcul_convection** *str* into ['ancien', 'divuT_moins_Tdivu', 'divrhout_moins_Tdivrhout']:
Option to set the form of the convective operator
 $\text{divrhout_moins_Tdivrhout} \rightarrow$ (the default since 1.6.8): $\rho \cdot u \cdot \text{grad} T = \text{div}(\rho \cdot u \cdot T) - T \cdot \text{div}(\rho \cdot u)$
ancien: $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \cdot \text{div}(u)$
 divuT_moins_Tdivu : $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \cdot \text{div}(u)$
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limite** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: *n_valeur*
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : *pbname_fieldname_[boundaryname]_time.dat*

- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.8 bloc_convection

Description: not_set

See also: objet_lecture (34)

Usage:

aco operateur acof
where

- **aco** *str* into ['']: Open accodance sign.
- **operateur** *convection_deriv* (4.8)
- **acof** *str* into ['']: Closed accodance sign.

4.8.1 convection_deriv

Description: not_set

See also: objet_lecture (34) *amount* (4.8.1) *amount_old* (4.8.2) *centre* (4.8.3) *centre4* (4.8.4) *centre_old* (4.8.5) *di_l2* (4.8.6) *ef* (4.8.7) *muscl3* (4.8.9) *ef_stab* (4.8.10) *generic* (4.8.13) *kquick* (4.8.14) *muscl* (4.8.15) *muscl_old* (4.8.16) *muscl_new* (4.8.17) *negligeable* (4.8.18) *quick* (4.8.19) *supg* (4.8.20) *btd* (4.8.21) *ale* (4.8.22)

Usage:

convection_deriv

4.8.2 amount

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

See also: convection_deriv (4.8)

Usage:

amount

4.8.3 **amont_old**

Description: not_set

See also: convection_deriv (4.8)

Usage:

amont_old

4.8.4 **centre**

Description: not_set

See also: convection_deriv (4.8)

Usage:

centre

4.8.5 **centre4**

Description: not_set

See also: convection_deriv (4.8)

Usage:

centre4

4.8.6 **centre_old**

Description: not_set

See also: convection_deriv (4.8)

Usage:

centre_old

4.8.7 **di_l2**

Description: not_set

See also: convection_deriv (4.8)

Usage:

di_l2

4.8.8 **ef**

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

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

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

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

Remark:

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

See also: `convection_deriv` (4.8)

Usage:

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

where

- **mot1** *str* into ['default_bar']: equivalent to `transportant_bar 0 transporte_bar 1 filtrer_resu 1 antisym 1`
- **bloc_ef** *bloc_ef* (4.8.8)

4.8.9 bloc_ef

Description: `not_set`

See also: `objet_lecture` (34)

Usage:

mot1 val1 mot2 val2 mot3 val3 mot4 val4

where

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

4.8.10 muscl3

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

See also: `convection_deriv` (4.8)

Usage:

muscl3 {

 [**alpha** *float*]

}

where

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

4.8.11 ef_stab

Description: Keyword for a VEF convective scheme.

See also: [convection_deriv \(4.8\)](#)

Usage:

```
ef_stab {  
    [ alpha float]  
    [ test int]  
    [ tdivu ]  
    [ old ]  
    [ volumes_etendus ]  
    [ volumes_non_etendus ]  
    [ montant_sous_zone str]  
    [ alpha_sous_zone listsous_zone_valeur]  
}  
where
```

- **alpha float**: To weight the scheme centering with the factor double (between 0 (full centered) and 1 (mix between upwind and centered), by default 1). For scalar equation, it is advised to use alpha=1 and for the momentum equation, alpha=0.2 is advised.
- **test int**: Developer option to compare old and new version of EF_stab
- **tdivu** : To have the convective operator calculated as $\text{div}(\mathbf{TU}) - \mathbf{T} \text{div} \mathbf{U} (= \mathbf{U} \text{grad} \mathbf{T})$.
- **old** : To use old version of EF_stab scheme (default no).
- **volumes_etendus** : Option for the scheme to use the extended volumes (default, yes).
- **volumes_non_etendus** : Option for the scheme to not use the extended volumes (default, no).
- **montant_sous_zone str**: Option to degenerate EF_stab scheme into Amount (upwind) scheme in the sub zone of name sz_name. The sub zone may be located arbitrarily in the domain but the more often this option will be activated in a zone where EF_stab scheme generates instabilities as for free outlet for example.
- **alpha_sous_zone listsous_zone_valeur (4.8.11)**: Option to change locally the alpha value on N sub-zones named sub_zone_name_I. Generally, it is used to prevent from a local divergence by increasing locally the alpha parameter.

4.8.12 listsous_zone_valeur

Description: List of groups of two words.

See also: [listobj \(33.3\)](#)

Usage:

n object1 object2

list of *sous_zone_valeur* ([4.8.12](#))

4.8.13 sous_zone_valeur

Description: Two words.

See also: [objet_lecture \(34\)](#)

Usage:

sous_zone valeur

where

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

4.8.14 generic

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

Examples:

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

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

See also: convection_deriv (4.8)

Usage:

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

where

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

4.8.15 kquick

Description: not_set

See also: convection_deriv (4.8)

Usage:

kquick

4.8.16 muscl

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

See also: convection_deriv (4.8)

Usage:

muscl

4.8.17 muscl_old

Description: not_set

See also: [convection_deriv \(4.8\)](#)

Usage:

muscl_old

4.8.18 muscl_new

Description: not_set

See also: [convection_deriv \(4.8\)](#)

Usage:

muscl_new

4.8.19 negligeable

Description: suppresses the convection operator.

See also: [convection_deriv \(4.8\)](#)

Usage:

negligeable

4.8.20 quick

Description: not_set

See also: [convection_deriv \(4.8\)](#)

Usage:

quick

4.8.21 supg

Description: not_set

See also: [convection_deriv \(4.8\)](#)

Usage:

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

}

where

- **facteur** *float*

4.8.22 btd

Description: not_set

See also: [convection_deriv \(4.8\)](#)

Usage:

btd {

```
[ facteur float]  
btd float
```

```
}  
where
```

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

4.8.23 ale

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

See also: convection_deriv (4.8)

Usage:

```
ale opconv  
where
```

- **opconv** *bloc_convection* (4.7)

4.9 convection_diffusion_chaleur_turbulent_qc

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

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

See also: convection_diffusion_chaleur_qc (4.6.2)

Usage:

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

- **modele_turbulence** *modele_turbulence_scal_base* (23): 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 $\hat{=}$ (the default since 1.6.8): $\rho \cdot u \cdot \text{grad} T = \text{div}(\rho \cdot u \cdot T) - T \cdot \text{div}(\rho \cdot u)$
ancien: $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \cdot \text{div}(u)$
divuT_moins_Tdivu : $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \cdot \text{div}(u)$

- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.10 convection_diffusion_concentration

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

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

See also: eqn_base (4.20) convection_diffusion_concentration_turbulent (4.11) convection_diffusion_phase-field (4.14) convection_diffusion_concentration_ft_disc (4.10)

Usage:

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

where

- **nom_inconnue** *str*: Keyword `Nom_inconnue` will rename the unknown of this equation with the given name. In the postprocessing part, the concentration field will be accessible with this name. This is useful if you want to track more than one concentration (otherwise, only the concentration field in the first concentration equation can be accessed).
- **masse_molaire** *float*
- **alias** *str*
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: `n_valeur`
`x_1 y_1 [z_1] val_1`
`...`
`x_n y_n [z_n] val_n`
The created files are named : `pname_fieldname_[boundaryname]_time.dat`
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: `n_valeur`
`x_1 y_1 [z_1] val_1`
`...`
`x_n y_n [z_n] val_n`
The created files are named : `pname_fieldname_[boundaryname]_time.dat`
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while `condition(t)` is verified if `equation_non_resolue` keyword is used. Exemple: The Navier Stokes is not solved between time `t0` and `t1`.
`Navier_Sokes_Standard`
`{ equation_non_resolue (t>t0)*(t<t1) }`

4.11 convection_diffusion_concentration_ft_disc

Description: `not_set`

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

See also: `convection_diffusion_concentration` (4.9)

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 condinit]
[ 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* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinit* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: *n_valeur*

```

x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n

```

The created files are named : *pbname_fieldname_[boundaryname]_time.dat*
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: *n_valeur*

```

x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n

```

The created files are named : *pbname_fieldname_[boundaryname]_time.dat*
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.

```

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

```

4.12 convection_diffusion_concentration_turbulent

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

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

See also: `convection_diffusion_concentration` (4.9)

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* (23): Turbulence model to be used in the constituent transportation equations. The only model currently available is Schmidt.
- **nom_inconnue** *str* for inheritance: Keyword `Nom_inconnue` will rename the unknown of this equation with the given name. In the postprocessing part, the concentration field will be accessible with this name. This is usefull if you want to track more than one concentration (otherwise, only the concentration field in the first concentration equation can be accessed).
- **masse_molaire** *float* for inheritance
- **alias** *str* for inheritance
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: `n_valeur`

```
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
```

The created files are named : `pdbname_fieldname_[boundaryname]_time.dat`
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: `n_valeur`

```
x_1 y_1 [z_1] val_1
...

```


x_n y_n [z_n] val_n

The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.13 convection_diffusion_fraction_massique_qc

Description: not_set

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

See also: eqn_base (4.20)

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* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...

x_n y_n [z_n] val_n

The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier Stokes is not solved between time t0 and t1.

Navier_Sokes_Standard

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

4.14 convection_diffusion_fraction_massique_turbulent_qc

Description: not_set

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

See also: eqn_base (4.20)

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* (23): Turbulence model to be used.
- **espece** *espece* (15)
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur

x_1 y_1 [z_1] val_1

...

x_n y_n [z_n] val_n

The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.

Navier_Sokes_Standard

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

4.15 convection_diffusion_phase_field

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

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

See also: convection_diffusion_concentration (4.9)

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 usefull if you want to track more than one concentration (otherwise, only the concentration field in the first concentration equation can be accessed).

- **masse_molaire** *float* for inheritance
- **alias** *str* for inheritance
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: *n_valeur*
`x_1 y_1 [z_1] val_1`
`...`
`x_n y_n [z_n] val_n`
The created files are named : `pbname_fieldname_[boundaryname]_time.dat`
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: *n_valeur*
`x_1 y_1 [z_1] val_1`
`...`
`x_n y_n [z_n] val_n`
The created files are named : `pbname_fieldname_[boundaryname]_time.dat`
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if `equation_non_resolue` keyword is used. Exemple: The Navier Stokes is not solved between time `t0` and `t1`.
`Navier_Sokes_Standard`
`{ equation_non_resolue (t>t0)*(t<t1) }`

4.16 convection_diffusion_temperature

Description: Energy equation (temperature diffusion convection).

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

See also: `eqn_base` (4.20) `convection_diffusion_temperature_ft_disc` (4.17.1)

Usage:

```
convection_diffusion_temperature obj Lire obj {
    [ penalisation_l2_ftd pp]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ initial_conditions|conditions_initiales condinits]
    [ boundary_conditions|conditions_limites condlims]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
    [ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
}
```

where

- **penalisation_l2_ftd** *pp* (4.16): to activate or not (the default is Direct Forcing method) the Penalized Direct Forcing method to impose the specified temperature on the solid-fluid interface.
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: *n_valeur*
`x_1 y_1 [z_1] val_1`
`...`
`x_n y_n [z_n] val_n`
The created files are named : *pbname_fieldname_[boundaryname]_time.dat*
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: *n_valeur*
`x_1 y_1 [z_1] val_1`
`...`
`x_n y_n [z_n] val_n`
The created files are named : *pbname_fieldname_[boundaryname]_time.dat*
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if *equation_non_resolue* keyword is used. Exemple: The Navier Stokes is not solved between time *t0* and *t1*.
`Navier_Sokes_Standard`
`{ equation_non_resolue (t>t0)*(t<t1) }`

4.17 pp

Description: *not_set*

See also: *listobj* (33.3)

Usage:

`{ object1 object2 }`

list of *penalisation_l2_ftd_lec* (4.17)

4.17.1 penalisation_l2_ftd_lec

Description: *not_set*

See also: *objet_lecture* (34)

Usage:

bord val

where

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

4.18 convection_diffusion_temperature_ft_disc

Description: not_set

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

See also: convection_diffusion_temperature (4.15)

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 orther 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* (4.18): `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* (4.16) 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* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)

- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.19 objet_lecture_maintien_temperature

Description: not_set

See also: objet_lecture (34)

Usage:

sous_zone temperature_moyenne

where

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

4.20 convection_diffusion_temperature_turbulent

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

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

See also: eqn_base (4.20)

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* (23): Turbulence model for the energy equation.
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.21 eqn_base

Description: Basic class for equations.

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

See also: mor_eqn (4) navier_stokes_standard (4.28) convection_diffusion_temperature (4.15) convection_diffusion_temperature_turbulent (4.19) conduction (4) convection_diffusion_chaleur_qc (4.6.2) transport_k_epsilon (4.36.2) convection_diffusion_concentration (4.9) convection_diffusion_fraction_massique_qc (4.12) convection_diffusion_fraction_massique_turbulent_qc (4.13) transport_interfaces_ft_disc (4.31) transport_marqueur_ft (4.37)

Usage:

```

eqn_base obj Lire obj {
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]

```



```

[ initial_conditions|conditions_initiales condinits]
[ boundary_conditions|conditions_limites condlims]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
[ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
}
where

```

- **convection** *bloc_convection* (4.7): Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1): Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9): Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10): Boundary conditions.
- **sources** *sources* (4.3.1): To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4): This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: *n_valeur*
 x_1 y_1 [z_1] val_1
...
 x_n y_n [z_n] val_n
The created files are named : *pbname_fieldname_[boundaryname]_time.dat*
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4): This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: *n_valeur*
 x_1 y_1 [z_1] val_1
...
 x_n y_n [z_n] val_n
The created files are named : *pbname_fieldname_[boundaryname]_time.dat*
- **parametre_equation** *parametre_equation_base* (4.5.2): Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str*: The equation will not be solved while condition(t) is verified if equation-*_non_resolue* keyword is used. Exemple: The Navier Stokes is not solved between time t_0 and t_1 .
Navier_Sokes_Standard
{ *equation_non_resolue* ($t > t_0$)*($t < t_1$) }

4.22 navier_stokes_ft_disc

Description: Two-phase momentum balance equation.

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

See also: *navier_stokes_turbulent* (4.29)

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

```

- **equation_interfaces_proprietes_fluide** *str*: This keyword is used for liquid-gas, liquid-vapor and fluid-fluid deformable interface, which transported at the Eulerian velocity. When this case is selected, the keyword sequence *Methode_transport vitesse_interpolee* is used in the block *Transport_Interfaces_FT_Disc* to define the velocity field for the displacement of the interface.
- **equation_interfaces_vitesse_imposee** *str*: This keyword is used to specify the velocity field to be used when using an interface that mimics a solid interface moving with a given solid speed of displacement. When this case is selected, the keyword sequence *Methode_transport vitesse_imposee* in the *Transport_Interfaces_FT_Disc* block will define the velocity field for the displacement of the interface.
- **equations_interfaces_vitesse_imposee** *n word1 word2 ... wordn*: This keyword is used to specify the velocity field to be used when using an interface that mimics a solid interface moving with a given solid speed of displacement. When this case is selected, the keyword sequence *Methode_transport vitesse_imposee* in the *Transport_Interfaces_FT_Disc* block will define the velocity field for the displacement of the interface. If two or more solid interfaces are defined, then the keyword *equations_interfaces_vitesse_imposee* should be used.
- **clipping_courbure_interface** *int*: This keyword is used to numerically limit the values of curvature used in the momentum balance equation. Curvature is computed as usual, but values exceeding the clipping value are replaced by this threshold, before using the clipped curvature in the momentum balance. Each time a curvature value is clipped, a counter is increased by one unity and the value of the counter is written in the .err file at the end of the time step. This clipping allows not reducing drastically the time stepping when a geometrical singularity occurs in the interface mesh. However, physical phenomena may be concealed with the use of such a clipping.
- **terme_gravite** *str* into [*'rho_g'*, *'grad_i'*]: The *Terme_gravite* keyword changes the numerical scheme used for the gravity source term. The default is *grad_i*, which is designed to remove spurious currents around the interface. In this case, the pressure field does not contain the hydrostatic part but only a jump across the interface. This scheme seems not to work very well in vef. The *rho_g* option uses the more traditional source term, equal to $\rho \cdot g$ in the volume. In this case, the hydrostatic pressure is visible in the pressure field and the boundary conditions in pressure must be set accordingly. This model produces spurious currents in the vicinity of the fluid-fluid interfaces and with the immersed boundary conditions.

- **equation_temperature_mpoint** *str*: The equation_temperature_mpoint should be used in the case of liquid-vapor flow with phase-change (see the TRUST_ROOT/doc/TRUST/ft_chgt_phase.pdf written in French for more information about the model). The name of the temperature equation, defined with the convection_diffusion_temperature_ft_disc keyword, should be given.
- **matrice_pression_invariante**: This keyword is a shortcut to be used only when the flow is a single-phase one, with interface tracking only used for solid-fluid interfaces. In this peculiar case, the density of the fluid does not evolve during the computation and the pressure matrix does not need to be actuated at each time step.
- **penalisation_forcage** *penalisation_forcage* (4.22): 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.
- **modele_turbulence** *modele_turbulence_hyd_deriv* (4.23) 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 equation) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (9.11) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (9.11) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source_Qdm_lambdaup). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt_projection** *deuxmots* (4.24.25) for inheritance: nb value : This keyword checks every nb time-steps the equality of velocity divergence to zero. value is the criteria convergency for the solver used.
- **seuil_divU** *floatfloat* (4.24.28) 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* (4.25) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of

several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)

- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur

x_1 y_1 [z_1] val_1

...

x_n y_n [z_n] val_n

The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur

x_1 y_1 [z_1] val_1

...

x_n y_n [z_n] val_n

The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.23 penalisation_forcage

Description: penalisation_forcage

See also: objet_lecture (34)

Usage:

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

where

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

4.24 modele_turbulence_hyd_deriv

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

See also: objet_lecture (34) NUL (4.24.1) mod_turb_hyd_ss_maille (4.24.2) k_epsilon (4.24.18) k_epsilon_bas_reynolds (4.24.24) k_epsilon_v2 (4.24.26) k_epsilon_2_couches (4.24.27)

Usage:

```
modele_turbulence_hyd_deriv {
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
```

```

[ turbulence_paroi turbulence_paroi_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
[ eps_min float]
[ k_min float]
[ prandtl_k float]
[ prandtl_eps float]
}
where

```

- **correction_visco_turb_pour_controle_pas_de_temps** : Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float*: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro**i *turbulence_paro*i_base (31): Keyword to set the wall law.
- **dt_impr_ustar** *float*: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24): This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float*: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float*: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float*: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float*: Keyword to change the Pr_k value (default 1.0).
- **prandtl_eps** *float*: Keyword to change the Pr_ϵ value (default 1.3).

4.24.1 dt_impr_ustar_mean_only

Description: `not_set`

See also: `objet_lecture` (34)

Usage:

```

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

```

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

4.24.2 NUL

Description: not_set

See also: modele_turbulence_hyd_deriv (4.23)

Usage:

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

where

- **correction_visco_turb_pour_controle_pas_de_temps** : Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float*: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paroil** *turbulence_paroil_base* (31): Keyword to set the wall law.
- **dt_impr_ustar** *float*: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24): This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float*: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float*: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float*: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float*: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float*: Keyword to change the Pre value (default 1.3).

4.24.3 mod_turb_hyd_ss_maille

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

See also: modele_turbulence_hyd_deriv (4.23) sous_maille_wale (4.24.4) sous_maille_smago (4.24.5) combinaison (4.24.6) longueur_melange (4.24.7) sous_maille (4.24.8) sous_maille_selectif_mod (4.24.9) sous_maille_selectif (4.24.12) sous_maille_elt (4.24.13) sous_maille_axi (4.24.15) sous_maille_smago_filtre (4.24.16) sous_maille_smago_dyn (4.24.17)

Usage:

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

```

[ turbulence_paroit turbulence_paroit_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
[ eps_min float]
[ k_min float]
[ prandtl_k float]
[ prandtl_eps float]

```

}

where

- **formulation_a_nb_points** *form_a_nb_points* (4.24.3): The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']*: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the *corr_visco_turb* field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paroit** *turbulence_paroit_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named *datafile_ProblemName_Ustar.face* and *periode* refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named *datafile_ProblemName_Ustar_mean_only.out*. *periode* refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword *boundaries*, all the boundaries will be considered. If you use it, you must specify *nb_boundaries* which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.4 form_a_nb_points

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

See also: objet_lecture (34)

Usage:

nb dir1 dir2

where

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

4.24.5 sous_maille_wale

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

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

See also: mod_turb_hyd_ss_maille (4.24.2)

Usage:

sous_maille_wale {

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

}

where

- **cw** *float*: The unique parameter (constant) of the WALE-model (by default value 0.5).
- **formulation_a_nb_points** *form_a_nb_points* (4.24.3) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
 - volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 - volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume

cells (without smoothing procedure).

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

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

- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity; it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.6 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` (4.24.2)

Usage:

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

```
[ prandtl_eps float]
}
```

where

- **cs** *float*: This is an optional keyword and the value is used to set the constant used in the Smagorinsky model (This is currently only valid for Smagorinsky models and it is set to 0.18 by default) .
- **formulation_a_nb_points** *form_a_nb_points* (4.24.3) for inheritance: The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.7 combinaison

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

See also: mod_turb_hyd_ss_maille (4.24.2)

Usage:

```

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

```

- **nb_var** *n word1 word2 ... wordn*: Number and names of variables which will be used in the turbulent viscosity definition (by default 0)
- **fonction** *str*: Fonction for turbulent viscosity. X,Y,Z and variables defined previously can be used.
- **formulation_a_nb_points** *form_a_nb_points* (4.24.3) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str* into [*'volume'*, *'volume_sans_lissage'*, *'scotti'*, *'arrete'*] for inheritance: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the *corr_visco_turb* field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named *datafile_ProblemName_Ustar.face* and *periode* refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named *datafile_ProblemName_Ustar_mean_only.out*. *periode* refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword *boundaries*, all the boundaries will be

considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.

- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.8 longueur_melange

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

$$\nu_{u,t} = (Kappa.y)^2.dU/dy$$

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

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

See also: `mod_turb_hyd_ss_maille` ([4.24.2](#))

Usage:

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

- **canalx** *float*: [height] : plane channel according to Ox direction (for the moment, formulation in the code relies on fixed height : $H=2$).
- **tuyauz** *float*: [diameter] : pipe according to Oz direction (for the moment, formulation in the code relies on fixed diameter : $D=2$).
- **verif_dparoi** *str*
- **dmax** *float*: Maximum distance.
- **fichier** *str*
- **fichier_ecriture_K_Eps** *str*: When a restart with k-epsilon model is envisaged, this keyword allows to generate external MED-format file with evaluation of k and epsilon quantities (based on eddy turbulent viscosity and turbulent characteristic length returned by mixing length model). The frequency

of the MED file print is set equal to `dt_impr_ustar`. Moreover, k-eps MED field is automatically saved at the last time step. MED file is then used for the restarting K-Epsilon calculation with the `Champ_Fonc_Med` keyword.

- **formulation_a_nb_points** *form_a_nb_points* (4.24.3) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.9 sous_maille

Description: Structure sub-grid function model.

See also: `mod_turb_hyd_ss_maille` (4.24.2)

Usage:

```
sous_maille {
    [ formulation_a_nb_points form_a_nb_points]
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]
}
```

```

[ correction_visco_turb_pour_controle_pas_de_temps ]
[ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
[ turbulence_paroit turbulence_paroit_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
[ eps_min float]
[ k_min float]
[ prandtl_k float]
[ prandtl_eps float]
}
where

```

- **formulation_a_nb_points** *form_a_nb_points* (4.24.3) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paroit** *turbulence_paroit_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.10 sous_maille_selectif_mod

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

See also: mod_turb_hyd_ss_maille (4.24.2)

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_parois turbulence_parois_base]  
    [ dt_impr_ustar float]  
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]  
    [ nut_max float]  
    [ eps_min float]  
    [ k_min float]  
    [ prandtl_k float]  
    [ prandtl_eps float]  
}
```

where

- **thi** *deuxentiers* (4.24.10): For homogeneous isotropic turbulence (THI), two integers k_i and k_c are needed in VDF (not in VEF).
- **canal** *floatentier* (4.24.11): $h_{dir_faces_parois}$: For a channel flow, the half width h and the orientation of the wall dir_faces_parois are needed.
- **formulation_a_nb_points** *form_a_nb_points* (4.24.3) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]

- **turbulence_paro** *turbulence_paro_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.11 deuxentiers

Description: Two integers.

See also: `objet_lecture` (34)

Usage:

int1 int2

where

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

4.24.12 floatentier

Description: A real and an integer.

See also: `objet_lecture` (34)

Usage:

the_float the_int

where

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

4.24.13 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` (4.24.2)

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

```

- **formulation_a_nb_points** *form_a_nb_points* (4.24.3) for inheritance: The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paroit** *turbulence_paroibase* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.14 sous_maille_1elt

Description: Turbulence model sous_maille_1elt.

See also: mod_turb_hyd_ss_maille (4.24.2) sous_maille_1elt_selectif_mod (4.24.14)

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

where

- **formulation_a_nb_points** *form_a_nb_points* (4.24.3) for inheritance: The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the 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* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named

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

- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.15 sous_maille_1elt_selectif_mod

Description: Turbulence model sous_maille_1elt_selectif_mod.

See also: sous_maille_1elt ([4.24.13](#))

Usage:

```
sous_maille_1elt_selectif_mod {
    [ formulation_a_nb_points form_a_nb_points]
    [ longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']]
    [ correction_visco_turb_pour_controle_pas_de_temps ]
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
    [ turbulence_paroit turbulence_paroit_base]
    [ dt_impr_ustar float]
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
    [ nut_max float]
    [ eps_min float]
    [ k_min float]
    [ prandtl_k float]
    [ prandtl_eps float]
}
```

where

- **formulation_a_nb_points** *form_a_nb_points* ([4.24.3](#)) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.

- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.16 sous_maille_axi

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

See also: `mod_turb_hyd_ss_maille` (4.24.2)

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

where

- **formulation_a_nb_points** *form_a_nb_points* (4.24.3) for inheritance: The structure function is calculated on `nb_points` and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']* for inheritance: different ways to calculate the characteristic length may be specified :
`volume` : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
`volume_sans_lissage` : For VEF only. Characteristic length is based on the cubic root of the volume

cells (without smoothing procedure).

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

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

- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity; it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Pr_k value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pr_ϵ value (default 1.3).

4.24.17 sous_maille_smago_filtre

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

See also: `mod_turb_hyd_ss_maille` (4.24.2)

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

where

- **formulation_a_nb_points** *form_a_nb_points* (4.24.3) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str* into ['volume', 'volume_sans_lissage', 'scotti', 'arrete'] for inheritance: different ways to calculate the characteristic length may be specified :
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.18 sous_maille_smago_dyn

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

See also: mod_turb_hyd_ss_maille (4.24.2)

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

```

where

- **stabilise** *str into* ['6_points', 'moy_euler', 'plans_paralleles']
- **nb_points** *int*
- **formulation_a_nb_points** *form_a_nb_points* (4.24.3) for inheritance: The structure function is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille** *str into* ['volume', 'volume_sans_lissage', 'scotti', 'arrete'] for inheritance: different ways to calculate the characteristic length may be specified :
 volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
 volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
 scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
 arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the corr_visco_turb field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paroit** *turbulence_paroit_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).

- **prandtl_eps** *float* for inheritance: Keyword to change the Pr value (default 1.3).

4.24.19 k_epsilon

Description: Turbulence model (k-eps).

See also: `modele_turbulence_hyd_deriv` (4.23)

Usage:

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

where

- **cmu** *float*: Keyword to modify the Cmu constant of k-eps model : $Nut = Cmu * k * k / eps$ Default value is 0.09
- **transport_k_epsilon** *transport_k_epsilon* (4.36.2): Keyword to define the (k-eps) transportation equation.
- **modele_fonc_bas_reynolds** *modele_fonction_bas_reynolds_base* (4.24.19): This keyword is used to set the bas Reynolds model used.
- **correction_visco_turb_pour_controle_pas_de_temps** for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the `corr_visco_turb` field which is the correction of turbulent viscosity: it should be 1. on the whole domain.
- **correction_visco_turb_pour_controle_pas_de_temps_parametre** *float* for inheritance: Keyword to set a limitation to low time steps due to high values of turbulent viscosity. The limit for turbulent viscosity is the ratio between diffusive time-step and convective time-step is higher or equal to the given value [0-1]
- **turbulence_paro** *turbulence_paro_base* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.

- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.20 modele_fonction_bas_reynolds_base

Description: not_set

See also: objet_lecture (34) Lam_Bremhorst (4.24.20) Launder_Sharma (4.24.22) Jones_Launder (4.24.23)

Usage:

4.24.21 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 (4.24.19) standard_KEps (4.24.21)

Usage:

```
Lam_Bremhorst {
    [ fichier_distance_paroï str]
    [ reynolds_stress_isotrope int]
}
```

where

- **fichier_distance_paroï** *str*: refer to distance_paroï keyword
- **reynolds_stress_isotrope** *int*: keyword for isotropic Reynolds stress

4.24.22 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 (4.24.20)

Usage:

```
standard_KEps {
    [ fichier_distance_paroï str]
    [ reynolds_stress_isotrope int]
}
```

where

- **fichier_distance_paroï** *str* for inheritance: refer to distance_paroï keyword
- **reynolds_stress_isotrope** *int* for inheritance: keyword for isotropic Reynolds stress

4.24.23 Launder_Sharma

Description: Model described in ' Launder, B. E. and Sharma, B. I. (1974), Application of the Energy-Dissipation Model of Turbulence to the Calculation of Flow Near a Spinning Disc, Letters in Heat and Mass Transfer, Vol. 1, No. 2, pp. 131-138.'

See also: `modele_fonction_bas_reynolds_base` ([4.24.19](#))

Usage:

4.24.24 Jones_Launder

Description: Model described in ' Jones, W. P. and Launder, B. E. (1972), The prediction of laminarization with a two-equation model of turbulence, Int. J. of Heat and Mass transfer, Vol. 15, pp. 301-314.'

See also: `modele_fonction_bas_reynolds_base` ([4.24.19](#))

Usage:

4.24.25 k_epsilon_bas_reynolds

Description: Bas Reynolds k-eps turbulence model. Caution: this model is only available in the VDF module.

See also: `modele_turbulence_hyd_deriv` ([4.23](#))

Usage:

```
k_epsilon_bas_reynolds {  
    [ transport_k_epsilon_bas_reynolds bloc_lecture ]  
    [ modele_fonc_bas_reynolds deuxmots ]  
    [ correction_visco_turb_pour_controle_pas_de_temps ]  
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float ]  
    [ turbulence_paro turbulence_paro_base ]  
    [ dt_impr_ustar float ]  
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only ]  
    [ nut_max float ]  
    [ eps_min float ]  
    [ k_min float ]  
    [ prandtl_k float ]  
    [ prandtl_eps float ]  
}
```

where

- **transport_k_epsilon_bas_reynolds** *bloc_lecture* ([2.40](#)): Keyword to define the bas Reynolds k-eps transportation equation.
- **modele_fonc_bas_reynolds** *deuxmots* ([4.24.25](#)): Keyword to set the bas Reynolds model used. Currently, two models are available for VDF and VEF discretizations : Launder_Sharma for Launder-Sharma model or Jones_Launder for Jones-Launder model. When Launder Sharma's model is used, one must specify the correct constants C1 and C2 for K_eps transport equation source terms (C1=1.44 and C2=1.92).
- **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* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.26 deuxmots

Description: Two words.

See also: `objet_lecture` (34)

Usage:

mot_1 mot_2
where

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

4.24.27 k_epsilon_v2

Description: Keyword to refer to a turbulence model available in VDF discretization. This model is a variant of the k-eps turbulence model called K-Eps-V2. A transport equation for V2 is added to calculate turbulent viscosity ($Nut = C_{\mu} V2$).

See also: `modele_turbulence_hyd_deriv` (4.23)

Usage:

k_epsilon_v2 {
 [**transport_k_epsilon_v2** *bloc_lecture*]
 [**transport_v2** *bloc_lecture*]
 [**eqnf22** *bloc_lecture*]
 [**correction_visco_turb_pour_controle_pas_de_temps**]
 [**correction_visco_turb_pour_controle_pas_de_temps_parametre** *float*]
 [**turbulence_paro** *turbulence_paro_base*]
 [**dt_impr_ustar** *float*]
 }

```

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

```

- **transport_k_epsilon_v2** *bloc_lecture* (2.40): Keyword to define the (k-eps) transportation equation.
- **transport_v2** *bloc_lecture* (2.40): Transport equation for V2.
- **eqnf22** *bloc_lecture* (2.40): Elliptic equation to calculate the V2 transport source term (solver like GMRES is needed).
- **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* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.24.28 k_epsilon_2_couches

Description: Turbulence model at two layers for the hydraulic equation is a variant of the k-eps turbulence model.

Warning : Model only available in VDF discretization.

See also: modele_turbulence_hyd_deriv (4.23)

Usage:

```

k_epsilon_2_couches {
    [ transport_k_kepsilon bloc_lecture]
    [ correction_visco_turb_pour_controle_pas_de_temps ]
}

```

```

[ correction_visco_turb_pour_controle_pas_de_temps_parametre float]
[ turbulence_paroit turbulence_paroit_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
[ eps_min float]
[ k_min float]
[ prandtl_k float]
[ prandtl_eps float]
}
where

```

- **transport_k_kepsilon** *bloc_lecture* (2.40): Transport equation for 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* (31) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (4.24) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **eps_min** *float* for inheritance: Lower limitation of epsilon (default value 1.e-10).
- **k_min** *float* for inheritance: Lower limitation of k (default value 1.e-10).
- **prandtl_k** *float* for inheritance: Keyword to change the Prk value (default 1.0).
- **prandtl_eps** *float* for inheritance: Keyword to change the Pre value (default 1.3).

4.25 floatfloat

Description: Two reals.

See also: `objet_lecture` (34)

Usage:

a b
where

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

4.26 traitement_particulier

Description: Auxiliary class to post-process particular values.

See also: objet_lecture (34)

Usage:

aco trait_part acof

where

- **aco** *str* into [' ']: Open accodance sign.
- **trait_part** *traitement_particulier_base* (4.26): Type of *traitement_particulier*.
- **acof** *str* into [' ']: Closed accodance sign.

4.26.1 traitement_particulier_base

Description: Basic class to post-process particular values.

See also: objet_lecture (34) temperature (4.26.1) canal (4.26.2) ec (4.26.3) thi (4.26.4) chmoy_faceperio (4.26.6) concmoy (4.26.7) profils_thermo (4.26.8) brech (4.26.9) ceg (4.26.10)

Usage:

4.26.2 temperature

Description: not_set

See also: traitement_particulier_base (4.26)

Usage:

temperature {

bord *str*

direction *int*

}

where

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

4.26.3 canal

Description: Keyword for statistics on a periodic plane channel.

See also: traitement_particulier_base (4.26)

Usage:

canal {

[**dt_impr_moy_spat** *float*]

[**dt_impr_moy_temp** *float*]

[**debut_stat** *float*]

[**fin_stat** *float*]

[**pulsation_w** *float*]

```

[ nb_points_par_phase int]
[ reprise str]
}

```

where

- **dt_impr_moy_spat** *float*: Period to print the spatial average (default value is 1e6).
- **dt_impr_moy_temp** *float*: Period to print the temporal average (default value is 1e6).
- **debut_stat** *float*: Time to start the temporal averaging (default value is 1e6).
- **fin_stat** *float*: Time to end the temporal averaging (default value is 1e6).
- **pulsation_w** *float*: Pulsation for phase averaging (in case of pulsating forcing term) (no default value).
- **nb_points_par_phase** *int*: Number of samples to represent phase average all along a period (no default value).
- **reprise** *str*: val_moy_temp_XXXXXX.sauv : Keyword to restart a calculation with previous average quantities.

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

4.26.4 ec

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

See also: traitement_particulier_base (4.26)

Usage:

```

ec {
    [ Ec ]
    [ Ec_dans_repere_fixe ]
    [ periode float]
}

```

where

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

4.26.5 thi

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

See also: traitement_particulier_base (4.26) thi_thermo (4.26.5)

Usage:

```

thi {
    init_Ec int
}

```

```

[ val_Ec float]
[ facon_init int into [0, 1]]
[ calc_spectre int into [0, 1]]
[ periode_calc_spectre float]
[ 3D int into [0, 1]]
[ 1D int into [0, 1]]
[ conservation_Ec ]
[ longueur_boite float]
}
where

```

- **init_Ec** *int*: Keyword to renormalize initial velocity so as kinetic energy equals to the value given by keyword **val_Ec**.
- **val_Ec** *float*: Keyword to impose a value for kinetic energy by velocity renormalized if **init_Ec** value is 1.
- **facon_init** *int into [0, 1]*: Keyword to specify how kinetic energy is computed (0 or 1).
- **calc_spectre** *int into [0, 1]*: Calculate or not the spectrum of kinetic energy.
Files called **Sorties_THI** are written with inside four columns :
time:t global_kinetic_energy:Ec enstrophy:D skewness:S
If **calc_spectre** is set to 1, a file **Sorties_THI2_2** is written with three columns :
time:t kinetic_energy_at_kc=32 enstrophy_at_kc=32
If **calc_spectre** is set to 1, a file **spectre_XXXXX** is written with two columns at each time **XXXXX** :
frequency:k energy:E(k).
- **periode_calc_spectre** *float*: Period for calculating spectrum of kinetic energy
- **3D** *int into [0, 1]*: Calculate or not the 3D spectrum
- **1D** *int into [0, 1]*: Calculate or not the 1D spectrum
- **conservation_Ec** : If set to 1, velocity field will be changed as to have a constant kinetic energy (default 0)
- **longueur_boite** *float*: Length of the calculation domain

4.26.6 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 ([4.26.4](#))

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 as 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

4.26.7 **chmoy_faceperio**

Description: non documente

See also: **traitement_particulier_base** ([4.26](#))

Usage:

chmoy_faceperio bloc

where

- **bloc** *bloc_lecture* ([2.40](#))

4.26.8 **concmoy**

Description: Keyword for printing concentration rates for a concentration equation

See also: **traitement_particulier_base** ([4.26](#))

Usage:

concmoy {

[**concmoy**]
[**periode** *float*]
[**tx1** *float*]
[**tx2** *float*]
[**tx3** *float*]

}

where

- **concmoy**

- **periode** *float*: periode is the keyword to set the period of printing into the file datafile_ConcMoy.son
- **tx1** *float*: tx1 is the limit 1 for concentration rate
- **tx2** *float*: tx2 is the limit 2 for concentration rate
- **tx3** *float*: tx3 is the limit 3 for concentration rate

4.26.9 **profils_thermo**

Description: non documente

See also: traitement_particulier_base (4.26)

Usage:

profils_thermo bloc

where

- **bloc** *bloc_lecture* (2.40)

4.26.10 **brech**

Description: non documente

See also: traitement_particulier_base (4.26)

Usage:

brech bloc

where

- **bloc** *bloc_lecture* (2.40)

4.26.11 **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 (4.26)

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* (4.26.11): AREVA's criterion
- **cea_jaea** *ceg_cea_jaea* (4.26.12): CEA_JAEA's criterion

4.26.12 ceg_areva

Description: not_set

See also: objet_lecture (34)

Usage:

```
{
    [ c float]
}
```

where

- **c** *float*

4.26.13 ceg_cea_jaea

Description: not_set

See also: objet_lecture (34)

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

4.27 navier_stokes_phase_field

Description: Navier Stokes equation for the Phase Field problem.

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

See also: navier_stokes_standard (4.28)

Usage:

```
navier_stokes_phase_field obj Lire obj {
```

```

approximation_de_boussinesq str into ['oui', 'non']
viscosite_dynamique_constante str into ['oui', 'non']
gravite n x1 x2 ... xn
[ methode_calcul_pression_initiale str into ['avec_les_cl', 'avec_sources', 'avec_sources_et-
_operateurs', 'sans_rien']]
[ projection_initiale int]
[ solveur_pression solveur_sys_base]
[ solveur_bar solveur_sys_base]
[ dt_projection deuxmots]
[ seuil_divU floatfloat]
[ traitement_particulier traitement_particulier]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ initial_conditions|conditions_initiales condinits]
[ boundary_conditions|conditions_limites condlims]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
[ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
}
where

```

- **approximation_de_boussinesq** *str* into ['oui', 'non']: To use or not the Boussinesq approximation.
- **viscosite_dynamique_constante** *str* into ['oui', 'non']: To use or not a viscosity which will depends on concentration C (in fact, C is the unknown of Cahn-Hilliard equation).
- **gravite** *n x1 x2 ... xn*: Keyword to define gravity in the case Boussinesq approximation is not used.
- **methode_calcul_pression_initiale** *str* into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']: for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : avec_les_cl (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), avec_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier Stokes equation) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (9.11) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (9.11) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source_Qdm_lambdaup). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt_projection** *deuxmots* (4.24.25) for inheritance: nb value : This keyword checks every nb time-steps the equality of velocity divergence to zero. value is the criteria convergency for the solver used.
- **seuil_divU** *floatfloat* (4.24.28) 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 ($\text{lmax}(\text{DivU}) * dt < \text{value}$)

```

Seuil(tn+1)= Seuil(tn)*factor
Else
Seuil(tn+1)= Seuil(tn)*factor
Endif

```

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

- **traitement_particulier** *traitement_particulier* (4.25) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.28 navier_stokes_qc

Description: NAVIER STOKES equations under small Mach number.

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

See also: navier_stokes_standard (4.28)

Usage:

navier_stokes_qc obj Lire obj {

```

[ methode_calcul_pression_initiale str into ['avec_les_cl', 'avec_sources', 'avec_sources_et-
_operateurs', 'sans_rien']]
[ projection_initiale int]
[ solveur_pression solveur_sys_base]
[ solveur_bar solveur_sys_base]
[ dt_projection deuxmots]

```

```

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

```

where

- **methode_calcul_pression_initiale** *str* into [*'avec_les_cl'*, *'avec_sources'*, *'avec_sources_et_operateurs'*, *'sans_rien'*] for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : *avec_les_cl* (default option $\text{lapP}=0$ is solved with Neuman boundary conditions on pressure if any), *avec_sources* ($\text{lapP}=f$ is solved with Neuman boundaries conditions and f integrating the source terms of the Navier Stokes equation) and *avec_sources_et_operateurs* ($\text{lapP}=f$ is solved as with the previous option *avec_sources* but f integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks $\text{DivU}=0$. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (9.11) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (9.11) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and *Source_Qdm_lambdaup*). A file (*solveur.bar*) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt_projection** *deuxmots* (4.24.25) for inheritance: *nb* value : This keyword checks every *nb* time-steps the equality of velocity divergence to zero. *value* is the criteria convergency for the solver used.
- **seuil_divU** *floatfloat* (4.24.28) 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 ($\text{lmax}(\text{DivU}) \cdot dt < \text{value}$)
 $\text{Seuil}(t_{n+1}) = \text{Seuil}(t_n) \cdot \text{factor}$
 Else
 $\text{Seuil}(t_{n+1}) = \text{Seuil}(t_n) \cdot \text{factor}$
 Endif
 The first parameter (*value*) is the mass evolution the user is ready to accept per timestep, and the second one (*factor*) is the factor of evolution for 'seuil' (for example 1.1, so 10)
- **traitement_particulier** *traitement_particulier* (4.25) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to

be separated by a comma)

- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.29 navier_stokes_standard

Description: NAVIER STOKES equations.

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

See also: eqn_base (4.20) navier_stokes_turbulent (4.29) navier_stokes_qc (4.27) navier_stokes_phase_field (4.26.13)

Usage:

```

navier_stokes_standard obj Lire obj {

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

}
where
```

- **methode_calcul_pression_initiale** *str into* [*'avec_les_cl'*, *'avec_sources'*, *'avec_sources_et_operateurs'*, *'sans_rien'*]: Keyword to select an option for the pressure calculation before the first time step. Options are : *avec_les_cl* (default option $\text{lapP}=0$ is solved with Neuman boundary conditions on pressure if any), *avec_sources* ($\text{lapP}=f$ is solved with Neuman boundaries conditions and f integrating the source terms of the Navier Stokes equation) and *avec_sources_et_operateurs* ($\text{lapP}=f$ is solved as with the previous option *avec_sources* but f integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.
- **projection_initiale** *int*: Keyword to suppress, if boolean equals 0, the initial projection which checks $\text{DivU}=0$. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (9.11): Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (9.11): This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and *Source_Qdm_lambdaup*). A file (*solveur.bar*) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt_projection** *deuxmots* (4.24.25): *nb value* : This keyword checks every *nb* time-steps the equality of velocity divergence to zero. *value* is the criteria convergency for the solver used.
- **seuil_divU** *floatfloat* (4.24.28): *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 ($\text{lmax}(\text{DivU}) \cdot dt < \text{value}$)
 $\text{Seuil}(t_{n+1}) = \text{Seuil}(t_n) \cdot \text{factor}$
 Else
 $\text{Seuil}(t_{n+1}) = \text{Seuil}(t_n) \cdot \text{factor}$
 Endif
 The first parameter (*value*) is the mass evolution the user is ready to accept per timestep, and the second one (*factor*) is the factor of evolution for 'seuil' (for example 1.1, so 10)
- **traitement_particulier** *traitement_particulier* (4.25): Keyword to post-process particular values.
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: *n_valeur*
 $x_1 \ y_1 \ [z_1] \ \text{val}_1$
 ...
 $x_n \ y_n \ [z_n] \ \text{val}_n$
 The created files are named : *pbname_fieldname_[boundaryname]_time.dat*
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: *n_valeur*
 $x_1 \ y_1 \ [z_1] \ \text{val}_1$
 ...
 $x_n \ y_n \ [z_n] \ \text{val}_n$
 The created files are named : *pbname_fieldname_[boundaryname]_time.dat*
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation

- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.

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

4.30 navier_stokes_turbulent

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

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

See also: `navier_stokes_standard` (4.28) `navier_stokes_turbulent_qc` (4.30) `navier_stokes_ft_disc` (4.21)

Usage:

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

where

- **modele_turbulence** *modele_turbulence_hyd_deriv* (4.23): 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 equation) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (9.11) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (9.11) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source-Qdm_lambdaup). A file (solveur.bar) is then created and used for inversion procedure. Syntax is

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

- **dt_projection** *deuxmots* (4.24.25) for inheritance: nb value : This keyword checks every nb time-steps the equality of velocity divergence to zero. value is the criteria convergency for the solver used.
- **seuil_divU** *floatfloat* (4.24.28) 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* (4.25) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limite** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur
 x_1 y_1 [z_1] val_1
 ...
 x_n y_n [z_n] val_n
 The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur
 x_1 y_1 [z_1] val_1
 ...
 x_n y_n [z_n] val_n
 The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier Stokes is not solved between time t0 and t1.
 Navier_Sokes_Standard
 { equation_non_resolue (t>t0)*(t<t1) }

4.31 navier_stokes_turbulent_qc

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

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

See also: `navier_stokes_turbulent` (4.29)

Usage:

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

- **modele_turbulence** *modele_turbulence_hyd_deriv* (4.23) for inheritance: Turbulence model for NAVIER STOKES equations.
- **methode_calcul_pression_initiale** *str* into [*'avec_les_cl'*, *'avec_sources'*, *'avec_sources_et_operateurs'*, *'sans_rien'*] for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : *avec_les_cl* (default option $\text{lapP}=0$ is solved with Neuman boundary conditions on pressure if any), *avec_sources* ($\text{lapP}=f$ is solved with Neuman boundaries conditions and f integrating the source terms of the Navier Stokes equation) and *avec_sources_et_operateurs* ($\text{lapP}=f$ is solved as with the previous option *avec_sources* but f integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier Stokes equation.
- **projection_initiale** *int* for inheritance: Keyword to suppress, if boolean equals 0, the initial projection which checks $\text{DivU}=0$. By default, boolean equals 1.
- **solveur_pression** *solveur_sys_base* (9.11) for inheritance: Linear pressure system resolution method.
- **solveur_bar** *solveur_sys_base* (9.11) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and `Source_Qdm_lambdaup`). A file (`solveur.bar`) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **dt_projection** *deuxmots* (4.24.25) for inheritance: nb value : This keyword checks every nb time-steps the equality of velocity divergence to zero. value is the criteria convergency for the solver used.
- **seuil_divU** *floatfloat* (4.24.28) 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* (4.25) for inheritance: Keyword to post-process particular values.
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limite** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.32 transport_interfaces_ft_disc

Description: Interface tracking equation for Front-Tracking problem in the discontinuous version.

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

See also: eqn_base (4.20)

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* (2.40): 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* (4.32): 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* (4.33.3): 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* (4.34): *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* (4.35): 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* (2.40)
- **distance_projete_faces** *str* into [*'simplifiee'*, *'initiale'*, *'modifiee'*]
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: *n_valeur*

```
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
```

The created files are named : *pbname_fieldname_[boundaryname]_time.dat*
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This key-

word is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur

x_1 y_1 [z_1] val_1

...

x_n y_n [z_n] val_n

The created files are named : pbname_fieldname_[boundaryname]_time.dat

- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.

Navier_Sokes_Standard

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

4.33 methode_transport_deriv

Description: Basic class for method of transport of interface.

See also: objet_lecture (34) loi_horaire (4.33) vitesse_imposee (4.33.1) vitesse_interpolee (4.33.2)

Usage:

methode_transport_deriv

4.33.1 loi_horaire

Description: not_set

See also: methode_transport_deriv (4.32)

Usage:

loi_horaire nom_loi

where

- **nom_loi** *str*

4.33.2 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 (4.32)

Usage:

vitesse_imposee val

where

- **val** *word1 word2 (word3)*: Analytical formula.

4.33.3 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` (4.32)

Usage:

vitesse_interpolee `val`
where

- **val** *str*: Navier-Stokes equation.

4.34 bloc_lecture_remaillage

Description: Parameters for remeshing.

See also: `objet_lecture` (34)

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.

4.35 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` (34)

Usage:

```
{  
    [ correction_parcours_thomas ]  
}  
where
```

- **correction_parcours_thomas**

4.36 interpolation_champ_face_deriv

Description: not_set

See also: objet_lecture (34) base (4.36) lineaire (4.36.1)

Usage:

4.36.1 base

Description: not_set

See also: interpolation_champ_face_deriv (4.35)

Usage:

base

4.36.2 lineaire

Description: not_set

See also: interpolation_champ_face_deriv (4.35)

Usage:

```
lineaire {  
    [ vitesse_fluide_explicite ]  
}  
where
```

- **vitesse_fluide_explicite**

4.37 transport_k_epsilon

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

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

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

See also: eqn_base (4.20)

Usage:

```
transport_k_epsilon obj Lire obj {
```

```

[ with_nu str into ['yes', 'no']]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ initial_conditions|conditions_initiales condinits]
[ boundary_conditions|conditions_limites condlims]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur_param]
[ ecrire_fichier_xyz_valeur_bin ecrire_fichier_xyz_valeur_param]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
}
where

```

- **with_nu** *str* into ['yes', 'no']: yes/no
- **convection** *bloc_convection* (4.7) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **initial_conditions|conditions_initiales** *condinits* (4.2.9) for inheritance: Initial conditions.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: *n_valeur*
 $x_1 \ y_1 \ [z_1] \ val_1$
...
 $x_n \ y_n \ [z_n] \ val_n$
The created files are named : *pbname_fieldname_[boundaryname]_time.dat*
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: *n_valeur*
 $x_1 \ y_1 \ [z_1] \ val_1$
...
 $x_n \ y_n \ [z_n] \ val_n$
The created files are named : *pbname_fieldname_[boundaryname]_time.dat*
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if *equation_non_resolue* keyword is used. Exemple: The Navier Stokes is not solved between time *t0* and *t1*.
Navier_Sokes_Standard
{ *equation_non_resolue* (*t*>*t0*)*(*t*<*t1*) }

4.38 transport_marqueur_ft

Description: *not_set*

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

See also: *eqn_base* (4.20)

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* (2.40): ne semble pas standard
- **injection** *injection_marqueur* (4.38): 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* (2.40): 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* (4.7) for inheritance: Keyword to alter the convection scheme.

- **diffusion** *bloc_diffusion* (4.1) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (3.10) for inheritance: Boundary conditions.
- **sources** *sources* (4.3.1) for inheritance: To introduce a source term into an equation (in case of several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma)
- **ecrire_fichier_xyz_valeur** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a text file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **ecrire_fichier_xyz_valeur_bin** *ecrire_fichier_xyz_valeur_param* (4.4) for inheritance: This keyword is used to write the values of a field for the whole domain or only for some boundaries in a binary file with the following format: n_valeur
x_1 y_1 [z_1] val_1
...
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat
- **parametre_equation** *parametre_equation_base* (4.5.2) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

4.39 injection_marqueur

Description: not_set

See also: objet_lecture (34)

Usage:

```
{
    ensemble_points bloc_lecture
    proprietes_particules bloc_lecture
    [ t_debut_injection float]
    [ dt_injection float]
}
```

where

- **ensemble_points** *bloc_lecture* (2.40)
- **proprietes_particules** *bloc_lecture* (2.40)
- **t_debut_injection** *float*
- **dt_injection** *float*

5 algo_base

Description: Basic class for multi-grid algorithms.

See also: [objet_u \(35\)](#) [algo_couple_1 \(5\)](#)

Usage:

5.1 algo_couple_1

Description: not_set

See also: [algo_base \(5\)](#)

Usage:

```
algo_couple_1 obj Lire obj {  
    [ dt_uniforme ]
```

```
}
```

where

- **dt_uniforme**

6 /*

6.1 /*

Description: bloc of Comment in a data file.

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

Usage:

```
/* comm
```

where

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

7 champ_generique_base

Description: not_set

See also: [objet_u \(35\)](#) [champ_post_de_champs_post \(7\)](#) [predefini \(7.14\)](#) [champ_post_refchamp \(7.16\)](#)

Usage:

7.1 champ_post_de_champs_post

Description: not_set

See also: [champ_generique_base \(7\)](#) [champ_post_operateur_eqn \(7.4\)](#) [champ_post_transformation \(7.18\)](#) [champ_post_reduction_0d \(7.15\)](#) [champ_post_operateur_base \(7.3\)](#) [champ_post_statistiques_base \(7.5\)](#) [champ_post_extraction \(7.9\)](#) [champ_post_morceau_equation \(7.12\)](#) [champ_post_tparoi_vef \(7.17\)](#) [champ_post_interpolation \(7.11\)](#)

Usage:

```
champ_post_de_champs_post obj Lire obj {
```

```

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

```

- **source** *champ_generique_base* (7): the source field.
- **nom_source** *str*: To name a source field with the `nom_source` keyword
- **source_reference** *str*
- **sources_reference** *list_nom_virgule* (7.1)
- **sources** *listchamp_generique* (7.2): sources { Champ_Post.... { ... } Champ_Post.. { ... } }

7.2 list_nom_virgule

Description: List of name.

See also: `listobj` (33.3)

Usage:
{ object1 , object2 }
list of *nom_anonyme* (24) separated with ,

7.3 listchamp_generique

Description: XXX

See also: `listobj` (33.3)

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

7.4 champ_post_operateur_base

Description: `not_set`

See also: `champ_post_de_champs_post` (7) `champ_post_operateur_gradient` (7.10) `champ_post_operateur_divergence` (7.7)

Usage:
champ_post_operateur_base obj Lire obj {

```

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

```

- **source** *champ_generique_base* (7) for inheritance: the source field.

- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

7.5 champ_post_operateur_eqn

Synonymous: **operateur_eqn**

Description: not_set

See also: champ_post_de_champs_post (7)

Usage:

```
champ_post_operateur_eqn obj Lire obj {
    [ numero_op int]
    [ numero_source int]
    [ sans_solveur_masse ]
    [ source champ_generique_base]
    [ nom_source str]
    [ source_reference str]
    [ sources_reference list_nom_virgule]
    [ sources listchamp_generique]
}
```

where

- **numero_op** *int*
- **numero_source** *int*
- **sans_solveur_masse**
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

7.6 champ_post_statistiques_base

Description: not_set

See also: champ_post_de_champs_post (7) correlation (7.6) moyenne (7.13) ecart_type (7.8)

Usage:

```
champ_post_statistiques_base obj Lire obj {
    t_deb float
    t_fin float
    [ source champ_generique_base]
    [ nom_source str]
    [ source_reference str]
    [ sources_reference list_nom_virgule]
```

```
[ sources listchamp_generique]
```

```
}
```

where

- **t_deb** *float*: Start of integration time
- **t_fin** *float*: End of integration time
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

7.7 correlation

Synonymous: **champ_post_statistiques_correlation**

Description: to calculate the correlation between the two fields.

See also: `champ_post_statistiques_base` (7.5)

Usage:

```
correlation obj Lire obj {
```

```
    t_deb float
```

```
    t_fin float
```

```
    [ source champ_generique_base]
```

```
    [ nom_source str]
```

```
    [ source_reference str]
```

```
    [ sources_reference list_nom_virgule]
```

```
    [ sources listchamp_generique]
```

```
}
```

where

- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

7.8 champ_post_operateur_divergence

Synonymous: **divergence**

Description: To calculate divergency of a given field.

See also: `champ_post_operateur_base` (7.3)

Usage:

```
champ_post_operateur_divergence obj Lire obj {
```

```

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

```

- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

7.9 ecart_type

Synonymous: **champ_post_statistiques_ecart_type**

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

See also: `champ_post_statistiques_base` (7.5)

Usage:

```

ecart_type obj Lire obj {
    t_deb float
    t_fin float
    [ source champ_generique_base]
    [ nom_source str]
    [ source_reference str]
    [ sources_reference list_nom_virgule]
    [ sources listchamp_generique]
}
where

```

- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

7.10 champ_post_extraction

Synonymous: **extraction**

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

See also: `champ_post_de_champs_post` (7)

Usage:

```
champ_post_extraction obj Lire obj {  
  
    domaine str  
    nom_frontiere str  
    [ methode str into ['trace', 'champ_frontiere']]  
    [ source champ_generique_base]  
    [ nom_source str]  
    [ source_reference str]  
    [ sources_reference list_nom_virgule]  
    [ sources listchamp_generique]  
  
}
```

where

- **domaine** *str*: name of the volume field
- **nom_frontiere** *str*: boundary name where the values of the volume field will be picked
- **methode** *str* into ['trace', 'champ_frontiere']: name of the extraction method (trace by_default or champ_frontiere)
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

7.11 champ_post_operateur_gradient

Synonymous: **gradient**

Description: To calculate gradient of a given field.

See also: champ_post_operateur_base (7.3)

Usage:

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

where

- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

7.12 champ_post_interpolation

Synonymous: **interpolation**

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

See also: champ_post_de_champs_post (7)

Usage:

```
champ_post_interpolation obj Lire obj {  
    localisation str  
    [ optimisation_sous_maillage str into ['default', 'yes', 'no']]  
    [ methode str]  
    [ domaine str]  
    [ source champ_generique_base]  
    [ nom_source str]  
    [ source_reference str]  
    [ sources_reference list_nom_virgule]  
    [ sources listchamp_generique]  
}
```

where

- **localisation** *str*: type_loc indicate where is done the interpolation (elem for element or som for node).
- **optimisation_sous_maillage** *str* into ['default', 'yes', 'no']
- **methode** *str*: The optional keyword methode is limited to calculer_champ_post for the moment.
- **domaine** *str*: the domain name where the interpolation is done (by default, the calculation domain)
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

7.13 champ_post_morceau_equation

Synonymous: **morceau_equation**

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

See also: champ_post_de_champs_post (7)

Usage:

```
champ_post_morceau_equation obj Lire obj {  
    type str  
    numero int  
    option str into ['stabilite', 'flux_bords']  
    [ compo int]  
    [ source champ_generique_base]  
    [ nom_source str]
```

```

[ source_reference str]
[ sources_reference list_nom_virgule]
[ sources listchamp_generique]
}

```

where

- **type** *str*: can only be `operateur` for equation operators.
- **numero** *int*: numero will be 0 (diffusive operator) or 1 (convective operator).
- **option** *str* into [`'stabilite'`, `'flux_bords'`]: option is stability for time steps or `flux_bords` for boundary fluxes.
- **compo** *int*: compo will specify the number component of the boundary flux (for boundary fluxes, in this case compo permits to specify the number component of the boundary flux choosen).
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: sources { `Champ_Post.... { ... } Champ_Post.. { ... }` }

7.14 moyenne

Synonymous: **champ_post_statistiques_moyenne**

Description: to calculate the average of the field over time

See also: **champ_post_statistiques_base** (7.5)

Usage:

```

moyenne obj Lire obj {
    [ moyenne_convergee champ_base]
    t_deb float
    t_fin float
    [ source champ_generique_base]
    [ nom_source str]
    [ source_reference str]
    [ sources_reference list_nom_virgule]
    [ sources listchamp_generique]
}

```

where

- **moyenne_convergee** *champ_base* (16): This option allows to read a converged time averaged field in a .xyz file in order to calculate, when restarting the calculation, the statistics fields (rms, correlation) which depend on this average. In that case, the time averaged field is not updated during the restarting calculation. In this case, the time averaged field must be fully converged to avoid errors when calculating high order statistics.
- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the `nom_source` keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: sources { `Champ_Post.... { ... } Champ_Post.. { ... }` }

7.15 predefini

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

See also: champ_generique_base (7)

Usage:

predefini obj Lire obj {

pb_champ *deuxmots*

}

where

- **pb_champ** *deuxmots* (4.24.25): { Pb_champ nom_pb nom_champ } : nom_pb is the problem name and nom_champ is the selected field name.

7.16 champ_post_reduction_0d

Synonymous: **reduction_0d**

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

See also: champ_post_de_champs_post (7)

Usage:

champ_post_reduction_0d obj Lire obj {

methode *str* into ['min', 'max', 'moyenne', 'somme', 'moyenne_ponderee', 'somme_ponderee', 'norme_l2']

 [**source** *champ_generique_base*]

 [**nom_source** *str*]

 [**source_reference** *str*]

 [**sources_reference** *list_nom_virgule*]

 [**sources** *listchamp_generique*]

}

where

- **methode** *str* into ['min', 'max', 'moyenne', 'somme', 'moyenne_ponderee', 'somme_ponderee', 'norme_l2']: name of the reduction method (min, max, somme for the sum, somme_ponderee for a weighted sum (integral), norme_L2 for the L2 norm, moyenne for a mean and moyenne_ponderee for a mean ponderated by integration volumes, e.g: cell volumes for temperature or pressure in VDF, volumes around faces for velocity and temperature in VEF)
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

7.17 champ_post_refchamp

Synonymous: **refchamp**

Description: Field of prolem

See also: `champ_generique_base` (7)

Usage:

champ_post_refchamp obj Lire obj {

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

}

where

- **pb_champ** *deuxmots* (4.24.25): { **Pb_champ** *nom_pb* *nom_champ* } : *nom_pb* is the problem name and *nom_champ* is the selected field name.
- **nom_source** *str*: The alias name for the field

7.18 champ_post_tparoi_vef

Synonymous: **tparoi_vef**

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

See also: `champ_post_de_champs_post` (7)

Usage:

champ_post_tparoi_vef obj Lire obj {

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

}

where

- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the *nom_source* keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: `sources { Champ_Post.... { ... } Champ_Post.. { ... } }`

7.19 champ_post_transformation

Synonymous: **transformation**

Description: To create a field with a transformation.

See also: `champ_post_de_champs_post` (7)

Usage:

```
champ_post_transformation obj Lire obj {  
    methode str into ['produit_scalaire', 'norme', 'vecteur', 'formule', 'composante']  
    [ expression n word1 word2 ... wordn]  
    [ numero int]  
    [ localisation str]  
    [ source champ_generique_base]  
    [ nom_source str]  
    [ source_reference str]  
    [ sources_reference list_nom_virgule]  
    [ sources listchamp_generique]  
}
```

where

- **methode** *str* into ['produit_scalaire', 'norme', 'vecteur', 'formule', 'composante']: methode norme : will calculate the norm of a vector given by a source field
methode produit_scalaire : will calculate the dot product of two vectors given by two sources fields
methode composante numero integer : will create a field by extracting the integer component of a field given by a source field
methode formule expression 1 : will create a scalar field located to elements using expressions with x,y,z,t parameters and field names given by a source field or several sources fields.
methode vecteur expression N f1(x,y,z,t) fN(x,y,z,t) : will create a vector field located to elements by defining its N components with N expressions with x,y,z,t parameters and field names given by a source field or several sources fields.
- **expression** *n word1 word2 ... wordn*: see methodes formule and vecteur
- **numero** *int*: see methode composante
- **localisation** *str*: type_loc indicate where is done the interpolation (elem for element or som for node). The optional keyword methode is limited to calculer_champ_post for the moment
- **source** *champ_generique_base* (7) for inheritance: the source field.
- **nom_source** *str* for inheritance: To name a source field with the nom_source keyword
- **source_reference** *str* for inheritance
- **sources_reference** *list_nom_virgule* (7.1) for inheritance
- **sources** *listchamp_generique* (7.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

8 chimie

Description: Keyword to describe the chmical reactions

See also: `objet_u` (35)

Usage:

```
chimie obj Lire obj {  
    reactions reactions
```

```

[ modele_micro_melange int]
[ constante_modele_micro_melange float]
[ espece_en_competition_micro_melange str]
}
where

```

- **reactions** *reactions* (8): list of reactions
- **modele_micro_melange** *int*: modele_micro_melange (0 by default)
- **constante_modele_micro_melange** *float*: constante of modele (1 by default)
- **espece_en_competition_micro_melange** *str*: espece in competition in reactions

8.1 reactions

Description: list of reactions

See also: listobj (33.3)

Usage:

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

list of *reaction* (8.1) separated with ,

8.1.1 reaction

Description: Keyword to describe reaction:

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

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

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

See also: objet_lecture (34)

Usage:

```

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

```

where

- **reactifs** *str*: LHS of equation (ex CH₄+2*O₂)
- **produits** *str*: RHS of equation (ex CO₂+2*H₂O)
- **constante_taux_reaction** *float*: constante of cinetic K
- **coefficients_activites** *bloc_lecture* (2.40): 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

9 class_generic

Description: not_set

See also: objet_u (35) dt_start (9.4) solveur_sys_base (9.11)

Usage:

9.1 cholesky

Description: Cholesky direct method.

See also: solveur_sys_base (9.11)

Usage:

cholesky obj Lire obj {

 [**impr**]

 [**quiet**]

}

where

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

9.2 dt_calc

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

See also: dt_start (9.4)

Usage:

dt_calc

9.3 dt_fixe

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

See also: dt_start (9.4)

Usage:

dt_fixe value

where

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

9.4 dt_min

Description: The first iteration is based on dt_min.

See also: dt_start (9.4)

Usage:

dt_min

9.5 dt_start

Description: not_set

See also: class_generic (9) dt_calc (9.1) dt_min (9.3) dt_fixe (9.2)

Usage:

dt_start

9.6 gcp_ns

Description: not_set

See also: gcp (9.10)

Usage:

```
gcp_ns obj Lire obj {  
    solveur0 solveur_sys_base  
    solveur1 solveur_sys_base  
    [ precond precond_base ]  
    [ precond_nul ]  
    seuil float  
    [ impr ]  
    [ quiet ]  
    [ save_matrix|save_matrice ]  
    [ optimized ]  
}
```

where

- **solveur0** *solveur_sys_base* (9.11): Solver type.
- **solveur1** *solveur_sys_base* (9.11): Solver type.
- **precond** *precond_base* (26) 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.

9.7 gen

Description: not_set

See also: solveur_sys_base (9.11)

Usage:

gen data

where

- **data** *bloc_lecture* (2.40)

9.8 gmres

Description: Gmres method (for non symmetric matrix).

See also: solveur_sys_base (9.11)

Usage:

gmres obj Lire obj {

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

}

where

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

9.9 optimal

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

See also: solveur_sys_base (9.11)

Usage:

optimal obj Lire obj {

```

    seuil float
    [ impr ]
    [ quiet ]
    [ save_matrix|save_matrice ]
    [ frequence_recalc int]
    [ nom_fichier_solveur str]
    [ fichier_solveur_non_recree ]
}
where

```

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

9.10 petsc

Description: Solveur via Petsc API

Usage:

```

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

```

Solver : Several solvers through PETSc API are available :

GCP : Conjugate Gradient

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

GMRES : Generalized Minimal Residual

BICGSTAB : Stabilized Bi-Conjugate Gradient

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

CHOLESKY : Parallelized version of Cholesky from MUMPS library. This solver accepts since the 1.6.7 version an option to select a different ordering than the automatic selected one by MUMPS (and printed by using the **impr** option). The possible choices are **Metis** | **Scotch** | **PT-Scotch** | **Parmetis**. The two last options can't only be used during a parallel calculation, whereas the two first are available for sequential or parallel calculations. It seems that the CPU cost of A=LU factorization but also of the backward/forward elimination steps may sometimes be reduced by selecting a different ordering than the default one. Notice that this solver requires a huge amount of memory compared to iterative methods. To know how many RAM you will need by core, then use the **impr** option to have detailed informations during the analysis phase and before the factorisation phase (in the following output, you will learn that the largest memory is taken by the 0th CPU with 108MB):

```

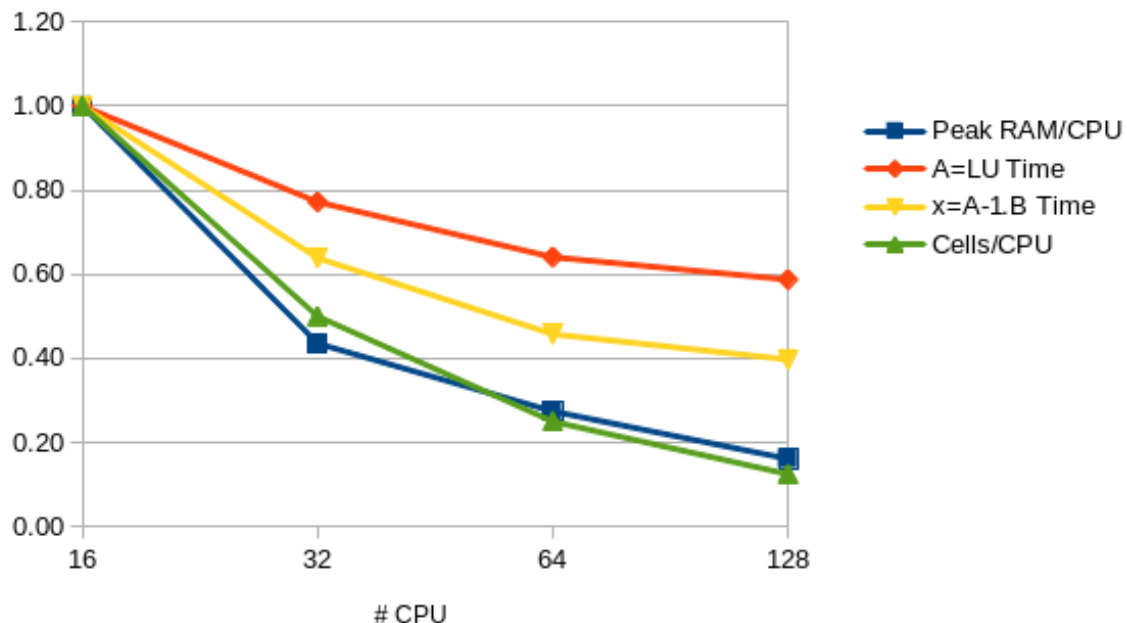
...
** Rank of proc needing largest memory in IC facto      :      0
** Estimated corresponding MBYTES for IC facto          :    108

```

...

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

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



CHOLESKY_OUT_OF_CORE : Same as the previous one but with a written LU decomposition of disc (save RAM memory but add an extra CPU cost during $Ax=B$ solve)

CHOLESKY_SUPERLU : Parallelized Cholesky from SUPERLU_DIST library (less CPU and RAM efficient than the previous one)

CHOLESKY_PASTIX : Parallelized Cholesky from PASTIX library

CHOLESKY_UMFPACK : Sequential Cholesky from UMFPACK library (seems fast).

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

trust datafile [N] -ksp_view -help

...

Preconditioner (PC) Options -----

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

HYPRE preconditioner options

-pc_hyre_type <pilut> (choose one of) pilut parasails boomeramg
HYPRE ParaSails Options
-pc_hyre_parasails_nlevels <1>: Number of number of levels (None)
-pc_hyre_parasails_thresh <0.1>: Threshold (None)
-pc_hyre_parasails_filter <0.1>: filter (None)
-pc_hyre_parasails_loadbal <0>: Load balance (None)
-pc_hyre_parasails_logging: <FALSE> Print info to screen (None)
-pc_hyre_parasails_reuse: <FALSE> Reuse nonzero pattern in preconditioner (None)
-pc_hyre_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_hyre_type boomeramg -ksp_atol 1.e-7 }

Precond : Several preconditioners are available :

NULL { } : No preconditioner used

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

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

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

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

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

DIAG { } : Diagonal (Jacobi) preconditioner.

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

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

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

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

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

save_matrix/read_matrix are the keywords to save/read into a file the constant matrix A of the linear system $Ax=B$ solved (eg: matrix from the pressure linear system for an incompressible flow). It is useful when you want to minimize the MPI communications on massive parallel calculation. Indeed, in VEF discretization, the overlapping width (generally 2, specified with the **largeur_joint** option in the partition keyword **partition**) can be reduced to 1, once the matrix has been properly assembled and saved. The cost of the MPI communications in TRUST itself (not in PETSc) will be reduced with length messages divided by 2. So the strategy is:

I) Partition your VEF mesh with a **largeur_joint** value of 2

II) Run your parallel calculation on 0 time step, to build and save the matrix with the **save_matrix** option. A file named *Matrix_NBROWS_rows_NCPUS_cpus.petsc* will be saved to the disc (where NBROWS is the number of rows of the matrix and NCPUS the number of CPUs used).

III) Partition your VEF mesh with a **largeur_joint** value of 1

IV) Run your parallel calculation completely now and substitute the **save_matrix** option by the **read_matrix** option. Some interesting gains have been noticed when the cost of linear system solve with PETSc is small compared to all the other operations.

TIPS:

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

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

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

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

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

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

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

See also: `solveur_sys_base` (9.11)

Usage:

petsc solveur option_solveur

where

- **solveur** *str*
- **option_solveur** *bloc_lecture* (2.40)

9.11 gcp

Description: Preconditioned conjugated gradient.

See also: `solveur_sys_base` (9.11) `gcp_ns` (9.5)

Usage:

```
gcp obj Lire obj {  
    [ precond precond_base ]  
    [ precond_nul ]  
    seuil float  
    [ impr ]  
    [ quiet ]  
    [ save_matrix|save_matrice ]  
    [ optimized ]  
}
```

where

- **precond** *precond_base* (26): 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.

9.12 solveur_sys_base

Description: Basic class to solve the linear system.

See also: `class_generic` (9) `optimal` (9.8) `gen` (9.6) `petsc` (9.9) `gcp` (9.10) `cholesky` (9) `gmres` (9.7)

Usage:

10 coeur

Description: `not_set`

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

Usage:

```
coeur obj Lire obj {  
    probleme str  
    type_probleme int into [0, 1]  
    nom_bord str  
    entreplat float  
    epaisseur_jeu float  
    nb_couronnes int  
    origine_numerotation int  
    [ test int ]  
}
```

where

- **probleme** *str*
- **type_probleme** *int into [0, 1]*
- **nom_bord** *str*
- **entreplat** *float*
- **epaisseur_jeu** *float*
- **nb_couronnes** *int*
- **origine_numerotation** *int*
- **test** *int*

11

11.1

Description: Comments in a data file.

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

Usage:

```
# comm
```

where

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

12 condlim_base

Description: Basic class of boundary conditions.

See also: [objet_u \(35\)](#) [paroi_fixe \(12.51\)](#) [symetrie \(12.67\)](#) [periodique \(12.64\)](#) [paroi_decalee_robin \(12.36\)](#) [paroi_adiabatique \(12.32\)](#) [dirichlet \(12.3\)](#) [neumann \(12.31\)](#) [paroi_couple \(12.35\)](#) [paroi_contact \(12.33\)](#) [paroi_contact_fictif \(12.34\)](#) [paroi_echange_contact_vdf \(12.42\)](#) [paroi_echange_externe_impose \(12.46\)](#) [paroi_echange_global_impose \(12.50\)](#) [Paroi \(12\)](#) [frontiere_ouverte_k_eps_impose \(12.17\)](#) [paroi_flux_impose \(12.53\)](#) [frontiere_ouverte_fraction_massique_imposee \(12.11\)](#) [paroi_echange_contact_correlation_vdf \(12.38\)](#) [paroi_echange_contact_correlation_vdf \(12.39\)](#) [paroi_ft_disc \(12.57\)](#) [flux_radiatif \(12.6\)](#) [contact_vdf_vdf \(12.1\)](#) [contact_vdf_vdf \(12.2\)](#) [sortie_libre_rho_variable \(12.65\)](#)

Usage:

```
condlim_base
```

12.1 Paroi

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

See also: `condlim_base` ([12](#))

Usage:

Paroi

12.2 contact_vdf_vef

Description: Boundary condition in the case of two problems (VDF -> VEF).

See also: `condlim_base` ([12](#))

Usage:

contact_vdf_vef champ

where

- **champ** `champ_front_base` ([17](#)): Boundary field type.

12.3 contact_vef_vdf

Description: Boundary condition in the case of two problems (VEF -> VDF).

See also: `condlim_base` ([12](#))

Usage:

contact_vef_vdf champ

where

- **champ** `champ_front_base` ([17](#)): Boundary field type.

12.4 dirichlet

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

See also: `condlim_base` ([12](#)) `paroi_defilante` ([12.37](#)) `paroi_knudsen_non_negligeable` ([12.59.2](#)) `paroi_rugueuse` ([12.60](#)) `frontiere_ouverte_vitesse_imposee` ([12.29](#)) `frontiere_ouverte_temperature_imposee` ([12.26](#)) `frontiere_ouverte_concentration_imposee` ([12.10](#)) `paroi_temperature_imposee` ([12.61](#))

Usage:

dirichlet

12.5 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` ([12.42](#))

Usage:

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

12.6 entree_temperature_imposee_h

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

See also: `frontiere_ouverte_temperature_imposee` (12.26)

Usage:

entree_temperature_imposee_h ch

where

- **ch** *champ_front_base* (17): Boundary field type.

12.7 flux_radiatif

Description: Boundary condition for radiation equation.

See also: `condlim_base` (12) `flux_radiatif_vdf` (12.7) `flux_radiatif_vdf` (12.8)

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): Wall emissivity, value between 0 and 1.

12.8 flux_radiatif_vdf

Description: Boundary condition for radiation equation in VDF.

See also: `flux_radiatif` (12.6)

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): Wall emissivity, value between 0 and 1.

12.9 flux_radiatif_vef

Description: Boundary condition for radiation equation in VEF.

See also: flux_radiatif (12.6)

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): Wall emissivity, value between 0 and 1.

12.10 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 (12.31) frontiere_ouverte_rayo_transp (12.22) frontiere_ouverte_rayo_semi_transp (12.21)

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): Boundary field type.

12.11 frontiere_ouverte_concentration_imposee

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

See also: dirichlet (12.3)

Usage:

frontiere_ouverte_concentration_imposee ch

where

- **ch** *champ_front_base* (17): Boundary field type.

12.12 `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](#)): Boundary field type.

12.13 `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` ([12.31](#))

Usage:

`frontiere_ouverte_gradient_pression_impose` **`ch`**

where

- **`ch`** `champ_front_base` ([17](#)): Boundary field type.

12.14 `frontiere_ouverte_gradient_pression_impose_vef`

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

See also: `frontiere_ouverte_pression_imposee` ([12.18](#)) `frontiere_ouverte_gradient_pression_impose_vefprep1b` ([12.14](#))

Usage:

`frontiere_ouverte_gradient_pression_impose_vef` **`ch`**

where

- **`ch`** `champ_front_base` ([17](#)): Boundary field type.

12.15 `frontiere_ouverte_gradient_pression_impose_vefprep1b`

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

See also: `frontiere_ouverte_gradient_pression_impose_vef` ([12.13](#))

Usage:

`frontiere_ouverte_gradient_pression_impose_vefprep1b` **`ch`**

where

- **`ch`** `champ_front_base` ([17](#)): Boundary field type.

12.16 `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` ([12.31](#))

Usage:

`frontiere_ouverte_gradient_pression_libre_vef`

12.17 `frontiere_ouverte_gradient_pression_libre_vefprep1b`

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

See also: `neumann` ([12.31](#))

Usage:

`frontiere_ouverte_gradient_pression_libre_vefprep1b`

12.18 `frontiere_ouverte_k_eps_impose`

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

See also: `condlim_base` ([12](#))

Usage:

`frontiere_ouverte_k_eps_impose ch`

where

- `ch` `champ_front_base` ([17](#)): Boundary field type.

12.19 `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` ([12.31](#)) `frontiere_ouverte_gradient_pression_impose_vef` ([12.13](#))

Usage:

`frontiere_ouverte_pression_imposee ch`

where

- `ch` `champ_front_base` ([17](#)): Boundary field type.

12.20 `frontiere_ouverte_pression_imposee_orlansky`

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

See also: `neumann` ([12.31](#))

Usage:

frontiere_ouverte_pression_imposee_orlansky

12.21 **frontiere_ouverte_pression_moyenne_imposee**

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

See also: `neumann` ([12.31](#))

Usage:

frontiere_ouverte_pression_moyenne_imposee pext

where

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

12.22 **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` ([12.9](#))

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](#)): Boundary field type.

12.23 **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` ([12.9](#)) `frontiere_ouverte_rayo_transp_vdf` ([12.23](#)) `frontiere_ouverte_rayo_transp-_vef` ([12.24](#))

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](#)): Boundary field type.

12.24 **frontiere_ouverte_rayo_transp_vdf**

Description: doit disparaitre

See also: `frontiere_ouverte_rayo_transp` ([12.22](#))

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](#)): Boundary field type.

12.25 **frontiere_ouverte_rayo_transp_vdf**

Description: doit disparaitre

See also: `frontiere_ouverte_rayo_transp` ([12.22](#))

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](#)): Boundary field type.

12.26 **frontiere_ouverte_rho_u_impose**

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

See also: `frontiere_ouverte_vitesse_imposee_sortie` ([12.30](#))

Usage:

frontiere_ouverte_rho_u_impose **ch**

where

- **ch** *champ_front_base* ([17](#)): Boundary field type.

12.27 **frontiere_ouverte_temperature_imposee**

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

See also: `dirichlet` ([12.3](#)) `entree_temperature_imposee_h` ([12.5](#)) `frontiere_ouverte_temperature_imposee_rayo_semi_transp` ([12.27](#)) `frontiere_ouverte_temperature_imposee_rayo_transp` ([12.28](#))

Usage:

frontiere_ouverte_temperature_imposee **ch**

where

- **ch** *champ_front_base* (17): Boundary field type.

12.28 **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* (12.26)

Usage:

frontiere_ouverte_temperature_imposee_rayo_semi_transp **ch**
where

- **ch** *champ_front_base* (17): Boundary field type.

12.29 **frontiere_ouverte_temperature_imposee_rayo_transp**

Description: Imposed temperature condition for a radiation problem with transparent gas.

See also: *frontiere_ouverte_temperature_imposee* (12.26)

Usage:

frontiere_ouverte_temperature_imposee_rayo_transp **ch**
where

- **ch** *champ_front_base* (17): Boundary field type.

12.30 **frontiere_ouverte_vitesse_imposee**

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

See also: *dirichlet* (12.3) *frontiere_ouverte_vitesse_imposee_sortie* (12.30)

Usage:

frontiere_ouverte_vitesse_imposee **ch**
where

- **ch** *champ_front_base* (17): Boundary field type.

12.31 **frontiere_ouverte_vitesse_imposee_sortie**

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

See also: *frontiere_ouverte_vitesse_imposee* (12.29) *frontiere_ouverte_rho_u_impose* (12.25)

Usage:

frontiere_ouverte_vitesse_imposee_sortie **ch**
where

- **ch** *champ_front_base* (17): Boundary field type.

12.32 **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 \(12.15\)](#) [frontiere_ouverte_gradient_pression_libre_vefprep1b \(12.16\)](#) [frontiere_ouverte_gradient_pression_impose \(12.12\)](#) [frontiere_ouverte_pression_imposee \(12.18\)](#) [frontiere_ouverte_pression_imposee_orlansky \(12.19\)](#) [frontiere_ouverte_pression_moyenne_imposee \(12.20\)](#) [frontiere_ouverte \(12.9\)](#) [sortie_libre_temperature_imposee_h \(12.66\)](#)

Usage:

neumann

12.33 **paroi_adiabatique**

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

See also: [condlim_base \(12\)](#)

Usage:

paroi_adiabatique

12.34 **paroi_contact**

Description: Thermal condition between two domains. Important: the name of the boundaries in the two domains should be the same. (Warning: there is also an old limitation not yet fixed on the sequential algorithm in VDF to detect the matching faces on the two boundaries: faces should be ordered in the same way). The kind of condition depends on the discretization. In VDF, it is a heat exchange condition, and in VEF, a temperature condition.

Such a coupling requires coincident meshes for the moment. In case of non-coincident meshes, run is stopped and two external files are automatically generated in VEF ([connectivity_failed_boundary_name](#) and [connectivity_failed_pb_name.med](#)). In 2D, the keyword [Decouper_bord_coincident](#) associated to the [connectivity_failed_boundary_name](#) file allows to generate a new coincident mesh.

In 3D, for a first preliminary cut domain with HOMARD (fluid for instance), the second problem associated to [pb_name](#) (solide in a fluid/solid coupling problem) has to be submitted to HOMARD cutting procedure with [connectivity_failed_pb_name.med](#).

Such a procedure works as while the primary refined mesh (fluid in our example) impacts the fluid/solid interface with a compact shape as described below (values 2 or 4 indicates the number of division from primary faces obtained in fluid domain at the interface after HOMARD cutting):

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

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

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

2-2

OK

2-2 2-2-2

2-4-2 2-2

2-2 2-2

NOT OK

See also: [condlim_base \(12\)](#)

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

12.35 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

12.36 paroi_couple

Description: `not_set`

See also: `condlim_base` ([12](#))

Usage:

paroi_couple autrepb

where

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

12.37 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*

12.38 paroi_defilante

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

See also: [dirichlet \(12.3\)](#)

Usage:

paroi_defilante **ch**
where

- **ch** *champ_front_base* ([17](#)): Boundary field type.

12.39 paroi_echange_contact_correlation_vdf

Description: Class to define a thermohydraulic 1D model which will apply to a boundary of 2D or 3D domain.

Warning : For parallel calculation, the only possible partition will be according the axis of the model with the keyword Tranche.

See also: [condlim_base \(12\)](#)

Usage:

paroi_echange_contact_correlation_vdf obj Lire obj {

dir *int*
tin *float*
tsup *float*
lambda *str*
rho *str*
cp *float*
dt_impr *float*
mu *str*
debit *float*
dh *float*
volume *str*
nu *str*
[**reprise_correlation**]

}
where

- **dir** *int*: Direction (0 : axis X, 1 : axis Y, 2 : axis Z) of the 1D model.
- **tin** *float*: Inlet fluid temperature of the 1D model (oC or K).
- **tsup** *float*: Outlet fluid temperature of the 1D model (oC or K).
- **lambda** *str*: Thermal conductivity of the fluid (W.m-1.K-1).
- **rho** *str*: Mass density of the fluid (kg.m-3) which may be a function of the temperature T.
- **cp** *float*: Calorific capacity value at a constant pressure of the fluid (J.kg-1.K-1).
- **dt_impr** *float*: Printing period in name_of_data_file_time.dat files of the 1D model results.

- **mu** *str*: Dynamic viscosity of the fluid (kg.m-1.s-1) which may be a function of the temperature T.
- **debit** *float*: Surface flow rate (kg.s-1.m-2) of the fluid into the channel.
- **dh** *float*: Hydraulic diameter may be a function f(x) with x position along the 1D axis (xinf <= x <= xsup)
- **volume** *str*: Exact volume of the 1D domain (m3) which may be a function of the hydraulic diameter (Dh) and the lateral surface (S) of the meshed boundary.
- **nu** *str*: Nusselt number which may be a function of the Reynolds number (Re) and the Prandtl number (Pr).
- **reprise_correlation** : Keyword in the case of a restarting calculation with this correlation.

12.40 paroi_echange_contact_correlation_vef

Description: Class to define a thermohydraulic 1D model which will apply to a boundary of 2D or 3D domain.

Warning : For parallel calculation, the only possible partition will be according the axis of the model with the keyword Tranche_geom.

See also: condlim_base ([12](#))

Usage:

```
paroi_echange_contact_correlation_vef obj Lire obj {
    dir int
    tinf float
    tsup float
    lambda str
    rho str
    cp float
    dt_impr float
    mu str
    debit float
    dh float
    n int
    surface str
    nu str
    xinf float
    xsup float
    [ emissivite_pour_rayonnement_entre_deux_plaques_quasi_infinies float]
    [ reprise_correlation ]
}
```

where

- **dir** *int*: Direction (0 : axis X, 1 : axis Y, 2 : axis Z) of the 1D model.
- **tinf** *float*: Inlet fluid temperature of the 1D model (oC or K).
- **tsup** *float*: Outlet fluid temperature of the 1D model (oC or K).
- **lambda** *str*: Thermal conductivity of the fluid (W.m-1.K-1).
- **rho** *str*: Mass density of the fluid (kg.m-3) which may be a function of the temperature T.
- **cp** *float*: Calorific capacity value at a constant pressure of the fluid (J.kg-1.K-1).
- **dt_impr** *float*: Printing period in name_of_data_file_time.dat files of the 1D model results.
- **mu** *str*: Dynamic viscosity of the fluid (kg.m-1.s-1) which may be a function of the temperature T.
- **debit** *float*: Surface flow rate (kg.s-1.m-2) of the fluid into the channel.
- **dh** *float*: Hydraulic diameter may be a function f(x) with x position along the 1D axis (xinf <= x <= xsup)

- **n** *int*: Number of 1D cells of the 1D mesh.
- **surface** *str*: Section surface of the channel which may be function $f(D_h, x)$ of the hydraulic diameter (D_h) 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 restarting calculation with this correlation.

12.41 paroi_echange_contact_odvm_vdf

Description: not_set

See also: paroi_echange_contact_vdf ([12.42](#))

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 $W.K^{-1}m^{-2}$) 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 :
$$q = h (T_1 - T_2) \text{ where } 1/h = d_1/\lambda_{a1} + 1/val_h_contact + d_2/\lambda_{a2}$$
where d_i : distance between the node where T_i and the wall is found.

12.42 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 ([12.42](#))

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 $W.K^{-1}m^{-2}$) 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 :
$$q = h (T_1 - T_2) \text{ where } 1/h = d_1/\lambda_{a1} + 1/val_h_contact + d_2/\lambda_{a2}$$
where d_i : distance between the node where T_i and the wall is found.

12.43 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 \(12\)](#) [exchange_contact_rayo_transp_vdf \(12.4\)](#) [paroi_echange_contact_rayo_semi_transp_vdf \(12.41\)](#) [paroi_echange_contact_vdf_ft \(12.43\)](#) [paroi_echange_contact_odvm_vdf \(12.40\)](#)

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 $1/h = d_1/\lambda_{d1} + 1/val_h_contact + d_2/\lambda_{d2}$
where d_i : distance between the node where T_i and the wall is found.

12.44 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 \(12.42\)](#)

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 $1/h = d_1/\lambda_{d1} + 1/val_h_contact + d_2/\lambda_{d2}$
where d_i : distance between the node where T_i and the wall is found.

12.45 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 \(12.46\)](#)

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): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* (17): Boundary field type.

12.46 paroi_echange_contact_vdf_zoom_grossier

Description: External type exchange condition with a heat exchange coefficient and an imposed external temperature in the case of zoom (coarse).

See also: [paroi_echange_externe_impose](#) (12.46)

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): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* (17): Boundary field type.

12.47 paroi_echange_externe_impose

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

See also: [condlim_base](#) (12) [paroi_echange_externe_impose_h](#) (12.47) [paroi_echange_externe_impose_rayo_transp](#) (12.49) [paroi_echange_externe_impose_rayo_semi_transp](#) (12.48) [paroi_echange_contact_vdf_zoom_grossier](#) (12.45) [paroi_echange_contact_vdf_zoom_fin](#) (12.44)

Usage:

paroi_echange_externe_impose h_imp himpc text ch
where

- **h_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ_front_base* (17): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* (17): Boundary field type.

12.48 paroi_echange_externe_impose_h

Description: Particular case of class [paroi_echange_externe_impose](#) for enthalpy equation.

See also: [paroi_echange_externe_impose](#) (12.46)

Usage:

paroi_echange_externe_impose_h h_imp himpc text ch
where

- **h_imp** *str*: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ_front_base* (17): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* (17): Boundary field type.

12.49 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` ([12.46](#))

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](#)): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* ([17](#)): Boundary field type.

12.50 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` ([12.46](#))

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](#)): Boundary field type.
- **text** *str*: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* ([17](#)): Boundary field type.

12.51 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](#)): Boundary field type.
- **text** *str*: External temperature value. The external temperature value is expressed in oC or K.
- **ch** *champ_front_base* ([17](#)): Boundary field type.

12.52 paroi_fixe

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

See also: `condlim_base` ([12](#)) `paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets` ([12.52](#))

Usage:

paroi_fixe

12.53 `paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets`

Description: CL to obtain iso Genepi2...

See also: `paroi_fixe` ([12.51](#))

Usage:

paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets

12.54 `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` ([12.56](#)) `paroi_flux_impose_rayo_semi_transp_vdf` ([12.54](#)) `paroi_flux_impose_rayo_semi_transp_vef` ([12.55](#))

Usage:

paroi_flux_impose ch

where

- **ch** `champ_front_base` ([17](#)): Boundary field type.

12.55 `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` ([12.53](#))

Usage:

paroi_flux_impose_rayo_semi_transp_vdf ch

where

- **ch** `champ_front_base` ([17](#)): Boundary field type.

12.56 `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` ([12.53](#))

Usage:

paroi_flux_impose_rayo_semi_transp_vef ch

where

- **ch** `champ_front_base` ([17](#)): Boundary field type.

12.57 **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` ([12.53](#))

Usage:

paroi_flux_impose_rayo_transp ch

where

- **ch** *champ_front_base* ([17](#)): Boundary field type.

12.58 **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* ([12.58](#)): Symetrie condition.

12.59 **paroi_ft_disc_deriv**

Description: `not_set`

See also: `objet_lecture` ([34](#)) `symetrie` ([12.59](#)) `constant` ([12.59.1](#))

Usage:

paroi_ft_disc_deriv

12.59.1 **symetrie**

Description: Symetrie condition in the case of two-phase flows

See also: `paroi_ft_disc_deriv` ([12.58](#))

Usage:

symetrie

12.59.2 **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` ([12.58](#))

Usage:

constant ch

where

- **ch** *champ_front_base* ([17](#)): Boundary field type.

12.60 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 \(12.3\)](#)

Usage:

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

where

- **name_champ_1** *str into ['vitesse_pari', 'k']*: Field name.
- **champ_1** *champ_front_base (17)*: Boundary field type.
- **name_champ_2** *str into ['vitesse_pari', 'k']*: Field name.
- **champ_2** *champ_front_base (17)*: Boundary field type.

12.61 paroi_rugueuse

Description: Rough wall boundary

See also: [dirichlet \(12.3\)](#)

Usage:

paroi_rugueuse **obj** Lire **obj** {

erugu *float*

}

where

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

12.62 paroi_temperature_imposee

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

See also: [dirichlet \(12.3\)](#) [temperature_imposee_pari \(12.68\)](#) [paroi_temperature_imposee_rayo_transp \(12.63\)](#) [paroi_temperature_imposee_rayo_semi_transp \(12.62\)](#)

Usage:

paroi_temperature_imposee **ch**

where

- **ch** *champ_front_base (17)*: Boundary field type.

12.63 `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` ([12.61](#))

Usage:

`paroi_temperature_imposee_rayo_semi_transp` **ch**

where

- **ch** *champ_front_base* ([17](#)): Boundary field type.

12.64 `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` ([12.61](#))

Usage:

`paroi_temperature_imposee_rayo_transp` **ch**

where

- **ch** *champ_front_base* ([17](#)): Boundary field type.

12.65 `periodique`

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

See also: `condlim_base` ([12](#))

Usage:

`periodique`

12.66 `sortie_libre_rho_variable`

Description: Class to define an outlet boundary condition at which the pressure is defined through the given field, whereas the density of the two-phase flow may varies (value of P/rho given in Pa/kg.m-3).

See also: `condlim_base` ([12](#))

Usage:

`sortie_libre_rho_variable` **ch**

where

- **ch** *champ_front_base* ([17](#)): Boundary field type.

12.67 sortie_libre_temperature_imposee_h

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

See also: [neumann \(12.31\)](#)

Usage:

sortie_libre_temperature_imposee_h **ch**

where

- **ch** *champ_front_base* ([17](#)): Boundary field type.

12.68 symetrie

Description: 1). For NAVIER STOKES equations, this keyword is used to designate a symmetry condition concerning the speed at the boundary called bord (edge) (normal speed at the edge equal to zero and tangential speed gradient at the edge equal to zero); 2). For scalar transport equation, this keyword is used to set a symmetry condition on scalar on the boundary named bord (edge).

See also: [condlim_base \(12\)](#)

Usage:

symetrie

12.69 temperature_imposee_paroi

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

See also: [paroi_temperature_imposee \(12.61\)](#)

Usage:

temperature_imposee_paroi **ch**

where

- **ch** *champ_front_base* ([17](#)): Boundary field type.

13 discretisation_base

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

See also: [objet_u \(35\)](#) [vdf \(13.1\)](#) [vef \(13.2\)](#) [ef \(13\)](#)

Usage:

13.1 ef

Description: Element Finite discretization.

See also: [discretisation_base \(13\)](#)

Usage:

13.2 vdf

Description: Finite difference volume discretization.

See also: [discretisation_base \(13\)](#)

Usage:

13.3 vef

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

Warning: it becomes an obsolete discretization.

See also: [discretisation_base \(13\)](#) [vefprep1b \(13.3\)](#)

Usage:

13.4 vefprep1b

Description: Finite element volume discretization (P1NC/P1-bubble element). Since the 1.5.5 version, several new discretizations are available thanks to the optional keyword Lire. By default, the VEFPreP1B keyword is equivalent to the former VEFPreP1B formulation (v1.5.4 and sooner). P0P1 (if used with the strong formulation for imposed pressure boundary) is equivalent to VEFPreP1B but the convergence is slower. VEFPreP1B dis is equivalent to VEFPreP1B dis Lire dis { P0 P1 Changement_de_base_P1Bulle 1 Cl_pression_sommet_faible 0 }

See also: [vef \(13.2\)](#)

Usage:

```
vefprep1b obj Lire obj {  
    [ p0 ]  
    [ p1 ]  
    [ pa ]  
    [ changement_de_base_p1bulle int into [0, 1]]  
    [ cl_pression_sommet_faible int into [0, 1]]  
    [ modif_div_face_dirichlet int into [0, 1]]  
}
```

where

- **p0** : Pressure nodes are added on element centres
- **p1** : Pressure nodes are added on vertices
- **pa** : Only available in 3D, pressure nodes are added on bones
- **changement_de_base_p1bulle** *int into [0, 1]*: This option may be used to have the P1NC/P0P1 formulation (value set to 0) or the P1NC/P1Bulle formulation (value set to 1, the default).
- **cl_pression_sommet_faible** *int into [0, 1]*: This option is used to specify a strong formulation (value set to 0, the default) or a weak formulation (value set to 1) for an imposed pressure boundary condition. The first formulation converges quicker and is stable in general cases. The second formulation should be used if there are several outlet boundaries with Neumann condition (see Ecoulement_Neumann test case for example).
- **modif_div_face_dirichlet** *int into [0, 1]*: This option (by default 0) is used to extend control volumes for the momentum equation.

14 domaine

Description: Keyword to create a domain.

See also: [objet_u \(35\)](#) [domaine_ale \(14\)](#)

Usage:

14.1 domaine_ale

Description: Domain with nodes at the interior of the domain are displaced in an arbitrarily prescribed way thanks to ALE description.

See also: [domaine \(14\)](#)

Usage:

15 espece

Description: `not_set`

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

Usage:

espece obj Lire obj {

cp *champ_base*
lambda *champ_base*
mu *champ_base*
masse_molaire *float*

}

where

- **cp** *champ_base* [\(16\)](#): Specific heat value (J.kg-1.K-1).
- **lambda** *champ_base* [\(16\)](#): Conductivity value (W.m-1.K-1).
- **mu** *champ_base* [\(16\)](#): 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 \(35\)](#) [champ_don_base \(16.1\)](#) [champ_ostwald \(16.14\)](#) [champ_input_base \(16.12\)](#) [champ_fonc_med \(16.5\)](#) [field_uniform_keps_from_ud \(16.22\)](#)

Usage:

16.2 champ_don_base

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

See also: `champ_base` (16) `uniform_field` (16.25) `champ_uniforme_morceaux` (16.18) `champ_fonc_xyz` (16.21) `champ_fonc_txyz` (16.20) `champ_don_lu` (16.2) `init_par_partie` (16.23) `champ_tabule_temps` (16.17) `champ_fonc_t` (16.8) `champ_fonc_tabule` (16.9) `champ_init_canal_sinal` (16.10) `champ_som_lu_vdf` (16.15) `champ_som_lu_vef` (16.16) `tayl_green` (16.24) `champ_fonc_reprise` (16.6)

Usage:

16.3 `champ_don_lu`

Description: Field to read a data field (values located at the center of the cells) in a file.

See also: `champ_don_base` (16.1)

Usage:

`champ_don_lu` **`dom`** **`nb_comp`** **`file`**

where

- **`dom`** *str*: Name of the domain.
- **`nb_comp`** *int*: Number of field components.
- **`file`** *str*: Name of the file.
This file has the following format:
`nb_val_lues` -> Number of values readen in th file
`Xi Yi Zi` -> Coordinates readen in the file
`Ui Vi Wi` -> Value of the field

16.4 `champ_fonc_fonction`

Description: Field that is a function of another field.

See also: `champ_fonc_tabule` (16.9) `champ_fonc_fonction_txyz` (16.4)

Usage:

`champ_fonc_fonction` **`dim`** **`inco`** **`bloc`**

where

- **`dim`** *int*: Number of field components.
- **`inco`** *str*: Name of the field (for example: temperature).
- **`bloc`** *bloc_lecture* (2.40): 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)).

16.5 `champ_fonc_fonction_txyz`

Description: this refers to a field that is a function of another field and time and/or space coordinates

See also: `champ_fonc_fonction` (16.3)

Usage:

`champ_fonc_fonction_txyz` **`dim`** **`inco`** **`bloc`**

where

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

- **bloc** *bloc_lecture* (2.40): Values (the table (the value of the field at any time is calculated by linear interpolation from this table) or the analytical expression (with keyword expression to use an analytical expression)).

16.6 champ_fonc_med

Description: Field to read a data field in a MED-format file .med at a specified time. It is very useful, for example, to restart a calculation with a new or refined geometry. The field post-processed on the new geometry at med format is used as initial condition for restarting.

See also: [champ_base](#) (16)

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.

16.7 champ_fonc_reprise

Description: This field is used to read a data field in a save file (.xyz or .sauv) at a specified time. It is very useful, for example, to run a thermohydraulic calculation with velocity initial condition read into a save file from a previous hydraulic calculation.

See also: [champ_don_base](#) (16.1)

Usage:

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

where

- **format** *str* into [*'binaire'*, *'formatte'*, *'xyz'*]: Type of file (the file format). If xyz format is activated, the .xyz file from the previous calculation will be given for filename, and if formatte or binaire is choosen, the .sauv file of the previous calculation will be specified for filename. In the case of a parallel calculation, if the mesh partition does not changed between the previous calculation and the next one, the binaire format should be preferred, because is faster than the xyz format.
- **filename** *str*: Name of the save file.
- **pb_name** *str*: Name of the problem.
- **champ** *str*: Name of the problem unknown. It may also be the temporal average of a problem unknown (like *moyenne_vitesse*, *moyenne_temperature*,...)
- **fonction** *fonction_champ_reprise* (16.7): Optional keyword to apply a function on the field being read in the save file (e.g. to read a temperature field in Celsius units and convert it for the calculation on Kelvin units, you will use: *fonction 1 273.+val*)
- **temps** *str*: Time of the saved field in the save file or last_time. If you give the keyword last_time instead, the last time saved in the save file will be used.

16.8 fonction_champ_reprise

Description: not_set

See also: objet_lecture (34)

Usage:

mot fonction

where

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

16.9 champ_fonc_t

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

See also: champ_don_base (16.1)

Usage:

champ_fonc_t val

where

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

16.10 champ_fonc_tabule

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

See also: champ_don_base (16.1) champ_fonc_fonction (16.3)

Usage:

champ_fonc_tabule dim inco bloc

where

- **dim** *int*: Number of field components.
- **inco** *str*: Name of the field (for example: temperature).
- **bloc** *bloc_lecture* (2.40): 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)).

16.11 champ_init_canal_sinal

Description: For a parabolic profile on U velocity with an unpredictable disturbance on V and W and a sinusoidal disturbance on V velocity.

See also: champ_don_base (16.1)

Usage:

champ_init_canal_sinal dim bloc

where

- **dim** *int*: Number of field components.
- **bloc** *bloc_lec_champ_init_canal_sinal* (16.11): Parameters for the class champ_init_canal_sinal.

16.12 bloc_lec_champ_init_canal_sinal

Description: Parameters for the class champ_init_canal_sinal.

in 2D:

$U = ucent * y(2h - y) / h / h$

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

rand: unpredictable value between -1 and 1.

in 3D:

$U = ucent * y(2h - y) / h / h$

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

$W = ampli_bruit * rand2$

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

See also: [objet_lecture \(34\)](#)

Usage:

```
{  
  
    ucent float  
    h float  
    ampli_bruit float  
    [ ampli_sin float]  
    omega float  
    [ dir_flow int into [0, 1, 2]]  
    [ dir_wall int into [0, 1, 2]]  
    [ min_dir_flow float]  
    [ min_dir_wall float]  
  
}
```

where

- **ucent float**: Velocity value at the center of the channel.
- **h float**: Half length of the channel.
- **ampli_bruit float**: Amplitude for the disturbance.
- **ampli_sin float**: Amplitude for the sinusoidal disturbance (by default equals to ucent/10).
- **omega float**: Value of pulsation for the of the sinusoidal disturbance.
- **dir_flow int into [0, 1, 2]**: Flow direction for the initialization of the flow in a channel.
 - if dir_flow=0, the flow direction is X
 - if dir_flow=1, the flow direction is Y
 - if dir_flow=2, the flow direction is ZDefault value for dir_flow is 0
- **dir_wall int into [0, 1, 2]**: Wall direction for the initialization of the flow in a channel.
 - if dir_wall=0, the normal to the wall is in X direction
 - if dir_wall=1, the normal to the wall is in Y direction
 - if dir_wall=2, the normal to the wall is in Z directionDefault value for dir_flow is 1
- **min_dir_flow float**: Value of the minimum coordinate in the flow direction for the initialization of the flow in a channel. Default value for dir_flow is 0.
- **min_dir_wall float**: Value of the minimum coordinate in the wall direction for the initialization of the flow in a channel. Default value for dir_flow is 0.

16.13 champ_input_base

Description: not_set

See also: `champ_base` ([16](#)) `champ_input_p0` ([16.13](#))

Usage:

```
champ_input_base obj Lire obj {  
    nb_comp int  
    nom str  
    [ initial_value n x1 x2 ... xn]  
    probleme str  
    [ sous_zone str]  
}
```

where

- **nb_comp** *int*
- **nom** *str*
- **initial_value** *n x1 x2 ... xn*
- **probleme** *str*
- **sous_zone** *str*

16.14 `champ_input_p0`

Description: `not_set`

See also: `champ_input_base` ([16.12](#))

Usage:

```
champ_input_p0 obj Lire obj {  
    nb_comp int  
    nom str  
    [ initial_value n x1 x2 ... xn]  
    probleme str  
    [ sous_zone str]  
}
```

where

- **nb_comp** *int* for inheritance
- **nom** *str* for inheritance
- **initial_value** *n x1 x2 ... xn* for inheritance
- **probleme** *str* for inheritance
- **sous_zone** *str* for inheritance

16.15 `champ_ostwald`

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

See also: `champ_base` ([16](#))

Usage:

```
champ_ostwald
```

16.16 champ_som_lu_vdf

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

See also: champ_don_base (16.1)

Usage:

champ_som_lu_vdf domain_name dim tolerance file

where

- **domain_name** *str*: Name of the domain.
- **dim** *int*: Value of the dimension of the field.
- **tolerance** *float*: Value of the tolerance to check the coordinates of the nodes.
- **file** *str*: name of the file

This file has the following format:

Xi Yi Zi -> Coordinates of the node

Ui Vi Wi -> Value of the field on this node

Xi+1 Yi+1 Zi+1 -> Next point

Ui+1 Vi+1 Zi+1 -> Next value ...

16.17 champ_som_lu_vdf

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

See also: champ_don_base (16.1)

Usage:

champ_som_lu_vdf domain_name dim tolerance file

where

- **domain_name** *str*: Name of the domain.
- **dim** *int*: Value of the dimension of the field.
- **tolerance** *float*: Value of the tolerance to check the coordinates of the nodes.
- **file** *str*: Name of the file.

This file has the following format:

Xi Yi Zi -> Coordinates of the node

Ui Vi Wi -> Value of the field on this node

Xi+1 Yi+1 Zi+1 -> Next point

Ui+1 Vi+1 Zi+1 -> Next value ...

16.18 champ_tabule_temps

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

See also: champ_don_base (16.1)

Usage:

champ_tabule_temps dim bloc

where

- **dim** *int*: Number of field components.
- **bloc** *bloc_lecture* (2.40): Values as a table. The value of the field at any time is calculated by linear interpolation from this table.

16.19 champ_uniforme_morceaux

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

See also: champ_don_base (16.1) champ_uniforme_morceaux_tabule_temps (16.19) valeur_totale_sur_volume (16.26)

Usage:

champ_uniforme_morceaux nom_dom nb_comp data

where

- **nom_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* (2.40): { 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.

16.20 champ_uniforme_morceaux_tabule_temps

Description: this type of field is constant in space on one or several sub_zones and tabulated as a function of time.

See also: champ_uniforme_morceaux (16.18)

Usage:

champ_uniforme_morceaux_tabule_temps nom_dom nb_comp data

where

- **nom_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* (2.40): { 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.

16.21 champ_fonc_txyz

Description: Field defined by analytical functions. It makes it possible the definition of a field that depends on the time and the space.

See also: champ_don_base (16.1)

Usage:

champ_fonc_txyz dom val

where

- **dom** *str*: Name of domain of calculation.
- **val** *n word1 word2 ... wordn*: List of functions on (t,x,y,z).

16.22 champ_fonc_xyz

Description: Field defined by analytical functions. It makes it possible the definition of a field that depends on (x,y,z).

See also: champ_don_base ([16.1](#))

Usage:

champ_fonc_xyz **dom** **val**

where

- **dom** *str*: Name of domain of calculation.
- **val** *n word1 word2 ... wordn*: List of functions on (x,y,z).

16.23 field_uniform_keps_from_ud

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

See also: champ_base ([16](#))

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

16.24 init_par_partie

Description: ne marche que pour n_comp=1

See also: champ_don_base ([16.1](#))

Usage:

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

where

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

16.25 tayl_green

Description: Class Tayl_green.

See also: champ_don_base ([16.1](#))

Usage:

tayl_green dim

where

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

16.26 uniform_field

Synonymous: **champ_uniforme**

Description: Field that is constant in space and stationary.

See also: **champ_don_base** ([16.1](#))

Usage:

uniform_field val

where

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

16.27 valeur_totale_sur_volume

Description: Similar as **Champ_Uniforme_Morceaux** with the same syntax. Used for source terms when we want to specify a source term with a value given for the volume (eg: heat in Watts) and not a value per volume unit (eg: heat in Watts/m3).

See also: **champ_uniforme_morceaux** ([16.18](#))

Usage:

valeur_totale_sur_volume nom_dom nb_comp data

where

- **nom_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* ([2.40](#)): { Defaut val_def sous_zone_1 val_1 ... sous_zone_i val_i } By default, the value val_def is assigned to the field. It takes the sous_zone_i identifier **Sous_Zone** (sub_area) type object value, val_i. **Sous_Zone** (sub_area) type objects must have been previously defined if the operator wishes to use a **Champ_Uniforme_Morceaux**(partly_uniform_field) type object.

17 champ_front_base

17.1 champ_front_base

Description: Basic class for fields at domain boundaries.

See also: **objet_u** ([35](#)) **champ_front_uniforme** ([17.23](#)) **champ_front_fonc_xyz** ([17.15](#)) **champ_front_fonc_txyz** ([17.14](#)) **champ_front_fonc_pois_ipsn** ([17.12](#)) **champ_front_fonc_pois_tube** ([17.13](#)) **champ_front_tabule** ([17.21](#)) **champ_front_fonction** ([17.16](#)) **champ_front_bruite** ([17.6](#)) **champ_front_tangentiel_vef** ([17.22](#)) **champ_front_lu** ([17.17](#)) **boundary_field_inward** ([17.1](#)) **champ_front_pression_from_u** ([17.19](#)) **champ_front_debit** ([17.11](#)) **champ_front_contact_vef** ([17.10](#)) **champ_front_calc** ([17.7](#)) **champ_front_recyclage** ([17.20](#)) **ch_front_input** ([17.3](#)) **boundary_field_uniform_keps_from_ud** ([17.2](#)) **champ_front_normal_vef** ([17.18](#)) **champ_front_vortex** ([17.24](#)) **champ_front_ale** ([17.5](#)) **champ_front_zoom** ([17.25](#))

Usage:

17.2 boundary_field_inward

Description: this field is used to define the normal vector field standard at the boundary in VDF or VEF discretization.

See also: [champ_front_base \(17\)](#)

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.

17.3 boundary_field_uniform_keps_from_ud

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

See also: [champ_front_base \(17\)](#)

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

17.4 ch_front_input

Description: not_set

See also: [champ_front_base \(17\)](#) [ch_front_input_uniforme \(17.4\)](#)

Usage:

ch_front_input obj Lire obj {

nb_comp *int*

nom *str*

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

probleme *str*

 [**sous_zone** *str*]

```
}  
where
```

- **nb_comp** *int*
- **nom** *str*
- **initial_value** *n x1 x2 ... xn*
- **probleme** *str*
- **sous_zone** *str*

17.5 ch_front_input_uniforme

Description: for coupling, you can use `ch_front_input_uniforme` which is a `champ_front_uniforme`, which use an external value. It must be used with `Problem.setInputField`.

See also: `ch_front_input` ([17.3](#))

Usage:

```
ch_front_input_uniforme obj Lire obj {
```

```
    nb_comp int  
    nom str  
    [ initial_value n x1 x2 ... xn ]  
    probleme str  
    [ sous_zone str ]
```

```
}  
where
```

- **nb_comp** *int* for inheritance
- **nom** *str* for inheritance
- **initial_value** *n x1 x2 ... xn* for inheritance
- **probleme** *str* for inheritance
- **sous_zone** *str* for inheritance

17.6 champ_front_ale

Description: Class to define a boundary condition on a moving boundary of a mesh.

See also: `champ_front_base` ([17](#))

Usage:

```
champ_front_ale val  
where
```

- **val** *n word1 word2 ... wordn*: Example:
2 20*0.3*SIN(6.28*y)*COS(20*t) 0.

17.7 champ_front_bruite

Description: Field which is variable in time and space in a random manner.

See also: `champ_front_base` ([17](#))

Usage:

champ_front_bruite **nb_comp** **bloc**

where

- **nb_comp** *int*: Number of field components.
- **bloc** *bloc_lecture* (2.40): { [N val L val] Moyenne $m_1.....[m_i]$ Amplitude $A_1.....[A_i]$ }:
Random noise: If N and L are not defined, the *i*th component of the field varies randomly around an average value m_i with a maximum amplitude A_i .
White noise: If N and L are defined, these two additional parameters correspond to L, the domain length and N, the number of nodes in the domain. Noise frequency will be between $2\pi/L$ and $2\pi N/(4L)$.
For example, formula for speed: $u=U_0(t)$ $v=U_1(t)$ $U_j(t)=M_j+2A_j*\text{bruit_blanc}$ where *bruit_blanc* (white_noise) is the formula given in the *mettre_a_jour* (update) method of the *Champ_front_bruite* (noise_boundary_field) (Refer to the *Ch_fr_bruite.cpp* file).

17.8 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)

Usage:

champ_front_calc **problem_name** **bord** **field_name**

where

- **problem_name** *str*: Name of the other problem to which *pb1* is coupled.
- **bord** *str*: Name of the side which is the boundary between the 2 domains in the domain object description associated with the *problem_name* object.
- **field_name** *str*: Name of the field containing the value that the user wishes to use at the boundary. The *field_name* object must be recognised by the *problem_name* object.

17.9 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* (17.10)

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.

17.10 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` ([17.10](#))

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.

17.11 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](#)) `champ_front_contact_rayo_transp_vef` ([17.9](#)) `champ_front_contact_rayo-semi_transp_vef` ([17.8](#))

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.

17.12 champ_front_debit

Description: This field is used to define a flow rate field instead of a velocity field for a Dirichlet boundary condition on Navier Stokes equation.

See also: `champ_front_base` ([17](#))

Usage:

champ_front_debit ch
where

- **ch** *champ_front_base* ([17](#)): field (`champ_front_uniforme`) to define the flow rate.

17.13 champ_front_fonc_pois_ipsn

Description: Boundary field `champ_front_fonc_pois_ipsn`.

See also: `champ_front_base` ([17](#))

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)*

17.14 champ_front_fonc_pois_tube

Description: Boundary field champ_front_fonc_pois_tube.

See also: champ_front_base ([17](#))

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)*

17.15 champ_front_fonc_txyz

Description: Boundary field which is not constant in space and in time.

See also: champ_front_base ([17](#))

Usage:

champ_front_fonc_txyz val

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

17.16 champ_front_fonc_xyz

Description: Boundary field which is not constant in space.

See also: champ_front_base ([17](#))

Usage:

champ_front_fonc_xyz val

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

17.17 champ_front_fonction

Description: boundary field that is function of another field

See also: champ_front_base ([17](#))

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.

17.18 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](#))

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

17.19 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](#))

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).

17.20 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](#))

Usage:

champ_front_pression_from_u expression

where

- **expression** *str*: value depending of a velocity (like $2 * u_{moy}^2$).

17.21 champ_front_recyclage

Description: This keyword is used on a boundary to get a field from another boundary. New keyword in the 1.6.1 version which replaces and generalizes several obsolete ones:

```

Champ_front_calc_intern
Champ_front_calc_recycl_fluct_pbperio
Champ_front_calc_recycl_champ
Champ_front_calc_intern_2pbs
Champ_front_calc_recycl_fluct
Champ_front_recyclage {
pb_champ_evaluateur pb field nb_comp
[ distance_plan dist0 dist1 [dist2] ]
[ moyenne_imposee methode_moy [fichier file [second_file] ]
[ moyenne_recyclee methode_recyc [fichier file [second_file] ]
[ direction_anisotrope 1|2|3 ]
[ ampli_moyenne_imposee 2|3 alpha(0) alpha(1) [alpha(2)] ]
[ ampli_moyenne_recyclee 2|3 beta(0) beta(1) [beta(2)] ]
[ ampli_fluctuation 2|3 gamma(0) gamma(1) [gamma(2)] ]
}

```

This keyword is to use, in a general way, on a boundary of a local_pb problem, a field calculated from a linear combination of an imposed field $g(x,y,z,t)$ with an instantaneous $f(x,y,z,t)$ and a spatial mean field $\langle f \rangle(t)$ or a temporal mean field $\langle f \rangle(x,y,z)$ field extracted from a plane of a problem named pb (pb may be local_pb itself) :

For each component i, the field F applied on the boundary will be:

$$F_i(x,y,z,t) = \alpha_i g_i(x,y,z,t) + \chi_i [f_i(x,y,z,t) - \beta_i \langle f_i \rangle]$$

The different options are:

pb_champ_evaluateur pb field nb_comp : To give the name of the pb problem, the name of the field of the problem and its number of components nb_comp.

distance_plan dist0 dist1 [dist2] : Vector which gives the distance between the boundary and the plane from where the field F will be extracted. By default, the vector is zero, that should imply the two domains have coincident boundaries.

ampli_moyenne_imposee 2|3 alpha(0) alpha(1) [alpha(2)] : α_i coefficients (by default =1)

ampli_moyenne_recyclee 2|3 beta(0) beta(1) [beta(2)] : β_i coefficients (by default =1)

ampli_fluctuation 2|3 gamma(0) gamma(1) [gamma(2)] : γ_i coefficients (by default =1)

direction_anisotrope direction : If an integer is given for direction (X:1, Y:2, Z:3, by default, direction is negative), the imposed field g will be 0 for the 2 other directions.

moyenne_imposee methode_moy : Value of the imposed g field. The methode_moy option can be :

profil [2|3] valx(x,y,z,t) valy(x,y,z,t) [valz(x,y,z,t)] : to specify analytic profile for the imposed g field.

interpolation fichier file : to create a imposed field built by interpolation of values read into a file. The imposed field is applied on the direction given by the keyword direction_anisotrope (the field is zero for the other directions). The format of the file is:

```
pos(1) val(1)
```

```
pos(2) val(2)
```

```
pos(N) val(N)
```

If direction given by direction_anisotrope is 1 (or 2 or 3), then pos will be X (or Y or Z) coordinate and val will be X value (or Y value, or Z value) of the imposed field.

connexion_approchee fichier file : to read the imposed field into a file where positions and values are given (it is not necessary that the coordinates of the points match the coordinates of the faces of the boundary, indeed, the nearest point of each face of the boundary will be used). The format of the file is:

```
N
```

x(1) y(1) [z(1)] valx(1) valy(1) [valz(1)]
 x(2) y(2) [z(2)] valx(2) valy(2) [valz(2)]

x(N) y(N) [z(N)] valx(N) valy(N) [valz(N)]

connection_exacte fichier file second_file : to read the imposed field into two files. The first file contains the points coordinates (which should be the same than the coordinates of each faces of the boundary) and the second_file contains the mean values. The format of the first file is:

N

1 x(1) y(1) [z(1)]

2 x(2) y(2) [z(2)]

N x(N) y(N) [z(N)]

The format of the second_file is:

N

1 valx(1) valy(1) [valz(1)]

2 valx(2) valy(2) [valz(2)]

...

N valx(N) valy(N) [valz(N)]

logarithmique diametre double u_tau double visco_cin double direction integer : to specify the imposed field (in this case, velocity) by an analytical logarithmic law of the wall :

$g(x,y,z) = u_tau * (\log(0.5*diametre*u_tau/visco_cin)/Kappa + 5.1)$

With $g(x,y,z)=u(x,y,z)$ if direction is set to 1 ($g=v(x,y,z)$ if direction is set to 2, and $g=w(w,y,z)$ if set to 3)

moyenne_recylee methode_recyc : Method used to do a spatial or a temporal averaging of f field to specify <f>. <f> can be the surface mean of f on the plane (surface option, see below) or it can be read from several files (for example generated by the chmoy_faceperio option of the Traitement_particulier keyword to obtain a temporal mean field). The option methode_recyc can be :

surfacique : surface mean for <f> from f values on the plane

Same options of methode_moy options but applied to read a temporal mean field <f>(x,y,z):

interpolation

connexion_approchee fichier file

connexion_exacte fichier file second_file

See also: champ_front_base ([17](#))

Usage:

champ_front_recyclage bloc

where

- **bloc** *str*

17.22 champ_front_tabule

Description: Constant field on the boundary, tabulated as a function of time.

See also: champ_front_base ([17](#))

Usage:

champ_front_tabule nb_comp bloc

where

- **nb_comp** *int*: Number of field components.

- **bloc** *bloc_lecture* (2.40): {nt1 t2 t3tn u1 [v1 w1 ...] u2 [v2 w2 ...] u3 [v3 w3 ...] ... un [vn wn ...]}

Values are entered into a table based on n couples (ti, ui) if nb_comp value is 1. The value of a field at a given time is calculated by linear interpolation from this table.

17.23 champ_front_tangentiel_vef

Description: Field to define the tangential speed vector field standard at the boundary in VEF discretisation.

See also: `champ_front_base` (17)

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].

17.24 champ_front_uniforme

Description: Boundary field which is constant in space and stationary.

See also: `champ_front_base` (17)

Usage:

champ_front_uniforme val

where

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

17.25 champ_front_vortex

Description: not_set

See also: `champ_front_base` (17)

Usage:

champ_front_vortex dom geom nu utau

where

- **dom** *str*: Name of domain.
- **geom** *str*
- **nu** *float*
- **utau** *float*

17.26 champ_front_zoom

Description: Basic class for fields at boundaries of two problems (global problem and local problem).

See also: `champ_front_base` (17)

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 \(35\)](#) [gaz_parfait \(18.2\)](#) [melange_gaz_parfait \(18.1\)](#) [gaz_reel_rhot \(18\)](#)

Usage:

18.1 gaz_reel_rhot

Description: Real gas.

See also: [loi_etat_base \(18\)](#)

Usage:

gaz_reel_rhot bloc
where

- **bloc** *bloc_lecture (2.40)*: 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*]
 Prandtl *float*
}
where

- **Sc** *float*: Schmidt number of the gas $Sc = \nu/D$ (D: diffusion coefficient of the mixing).
- **Prandtl** *float*: Prandtl number of the gas $Pr = \mu * Cp / \lambda$

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)

19 loi_horaire

Description: to define the movement with a time-dependant law for the solid interface.

See also: `objet_u` (35)

Usage:

```

loi_horaire obj Lire obj {
    position n word1 word2 ... wordn
    vitesse n word1 word2 ... wordn
    [ rotation n word1 word2 ... wordn]
    [ derivee_rotation n word1 word2 ... wordn]
}
where

```

- **position** *n word1 word2 ... wordn*
- **vitesse** *n word1 word2 ... wordn*
- **rotation** *n word1 word2 ... wordn*
- **derivee_rotation** *n word1 word2 ... wordn*

20 milieu_base

Description: Basic class for medium (physics properties of medium).

See also: `objet_u` (35) `solide` (20.5) `constituant` (20) `fluide_incompressible` (20.1)

Usage:

```

milieu_base obj Lire obj {
    [ rho champ_base]
    [ cp champ_base]
    [ lambda champ_base]
}
where

```

- **rho** *champ_base* (16): Density (kg.m-3).
- **cp** *champ_base* (16): Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (16): Conductivity (W.m-1.K-1).

20.1 constituent

Description: Constituent.

See also: milieu_base (20)

Usage:

```
constituant obj Lire obj {  
    [ coefficient_diffusion champ_base]  
    [ rho champ_base]  
    [ cp champ_base]  
    [ lambda champ_base]  
}  
where
```

- **coefficient_diffusion** champ_base (16): Constituent diffusion coefficient value (m².s⁻¹). If a multi-constituent problem is being processed, the diffusivity will be a vectorial and each component will be the diffusion of the constituent.
- **rho** champ_base (16) for inheritance: Density (kg.m⁻³).
- **cp** champ_base (16) for inheritance: Specific heat (J.kg⁻¹.K⁻¹).
- **lambda** champ_base (16) for inheritance: Conductivity (W.m⁻¹.K⁻¹).

20.2 fluide_incompressible

Description: This is an incompressible fluid.

See also: milieu_base (20) fluide_quasi_compressible (20.3) fluide_ostwald (20.2)

Usage:

```
fluide_incompressible obj Lire obj {  
    [ beta_th champ_base]  
    [ mu champ_base]  
    [ beta_co champ_base]  
    [ indice champ_base]  
    [ kappa champ_base]  
    [ rho champ_base]  
    [ cp champ_base]  
    [ lambda champ_base]  
}  
where
```

- **beta_th** champ_base (16): Thermal expansion (K⁻¹).
- **mu** champ_base (16): Dynamic viscosity (kg.m⁻¹.s⁻¹).
- **beta_co** champ_base (16): Volume expansion coefficient values in concentration.
- **indice** champ_base (16): Refractivity of fluid.
- **kappa** champ_base (16): Absorptivity of fluid (m⁻¹).
- **rho** champ_base (16) for inheritance: Density (kg.m⁻³).
- **cp** champ_base (16) for inheritance: Specific heat (J.kg⁻¹.K⁻¹).
- **lambda** champ_base (16) for inheritance: Conductivity (W.m⁻¹.K⁻¹).

20.3 fluide_ostwald

Description: Non-Newtonian fluids governed by Ostwald's law. The law applicable to stress tensor is:

$\tau = K(T) * (D:D/2)^{((n-1)/2)} * D$ Where:

D refers to the deformation speed tensor

K refers to fluid consistency (may be a function of the temperature T)

n refers to the fluid structure index n=1 for a Newtonian fluid, n<1 for a rheofluidifier fluid, n>1 for a rheothickening fluid.

See also: fluide_incompressible (20.1)

Usage:

fluide_ostwald obj Lire obj {

```
[ k champ_base]
[ n champ_base]
[ beta_th champ_base]
[ mu champ_base]
[ beta_co champ_base]
[ indice champ_base]
[ kappa champ_base]
[ rho champ_base]
[ cp champ_base]
[ lambda champ_base]
```

}

where

- **k** champ_base (16): Fluid consistency.
- **n** champ_base (16): Fluid structure index.
- **beta_th** champ_base (16) for inheritance: Thermal expansion (K-1).
- **mu** champ_base (16) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **beta_co** champ_base (16) for inheritance: Volume expansion coefficient values in concentration.
- **indice** champ_base (16) for inheritance: Refractivity of fluid.
- **kappa** champ_base (16) for inheritance: Absorptivity of fluid (m-1).
- **rho** champ_base (16) for inheritance: Density (kg.m-3).
- **cp** champ_base (16) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** champ_base (16) for inheritance: Conductivity (W.m-1.K-1).

20.4 fluide_quasi_compressible

Description: Compressible flow at low mach number.

See also: fluide_incompressible (20.1)

Usage:

fluide_quasi_compressible obj Lire obj {

```
[ sutherland bloc_sutherland]
[ pression float]
[ loi_etat loi_etat_base]
[ traitement_pth str into ['edo', 'constant', 'conservation_masse']]
[ traitement_rho_gravite str into ['standard', 'moins_rho_moyen']]
[ temps_debut_prise_en_compte_drho_dt float]
[ omega_relaxation_drho_dt float]
```



```

[ mu champ_base]
[ indice champ_base]
[ kappa champ_base]
[ rho champ_base]
[ cp champ_base]
[ lambda champ_base]
}
where

```

- **sutherland** *bloc_sutherland* (20.4): Sutherland law for viscosity and for conductivity.
- **pression** *float*: Initial pression.
- **loi_etat** *loi_etat_base* (18): State law.
- **traitement_pth** *str* into ['edo', 'constant', 'conservation_masse']: Particular treatment for the thermodynamic pressure Pth ; there are three possibilities:
 - 1) with the keyword 'edo' the code computes Pth solving an O.D.E. ; in this case, the mass is not strictly conserved (it is the default case for quasi compressible computation);
 - 2) the keyword 'conservation_masse' forces the conservation of the mass (closed geometry or with periodic boundaries condition)
 - 3) the keyword 'constant' makes it possible to have a constant Pth ; it's the good choice when the flow is open (e.g. with pressure boundary conditions).
- **traitement_rho_gravite** *str* into ['standard', 'moins_rho_moyen']: It may be :1) 'standard': the gravity term is evaluated with $\rho \cdot g$ (It is the default). 2) 'moins_rho_moyen': the gravity term is evaluated with $(\rho - \rho_{\text{moyen}}) \cdot g$.
- **temps_debut_prise_en_compte_drho_dt** *float*: While time < value, dRho/dt is set to zero (Rho, volumic mass). Useful for some calculation during the first time steps with big variation of temperature and volumic mass.
- **omega_relaxation_drho_dt** *float*: Optional option to have a relaxed algorithm to solve the mass equation. value is used (1 per default) to specify omega.
- **mu** *champ_base* (16) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **indice** *champ_base* (16) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (16) for inheritance: Absorptivity of fluid (m-1).
- **rho** *champ_base* (16) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (16) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (16) for inheritance: Conductivity (W.m-1.K-1).

20.5 bloc_sutherland

Description: Sutherland law for viscosity $\mu(T) = \mu_0 \cdot ((T_0 + C)/(T + C)) \cdot (T/T_0)^{1.5}$ and (optional) for conductivity $\lambda(T) = \mu_0 \cdot C_p / Prandtl \cdot ((T_0 + S\lambda)/(T + S\lambda)) \cdot (T/T_0)^{1.5}$

See also: *objet_lecture* (34)

Usage:

```

m mu0 t t0 [ ms ] [ s ] mc c
where

```

- **m** *str* into ['mu0']
- **mu0** *float*
- **t** *str* into ['T0']
- **t0** *float*
- **ms** *str* into ['Slambda']
- **s** *float*
- **mc** *str* into ['C']

- **c** *float*

20.6 solide

Description: Solid.

See also: `milieu_base` ([20](#))

Usage:

```
solide obj Lire obj {
    [ rho  champ_base]
    [ cp  champ_base]
    [ lambda champ_base]
}
```

where

- **rho** *champ_base* ([16](#)) for inheritance: Density (kg.m-3).
- **cp** *champ_base* ([16](#)) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* ([16](#)) for inheritance: Conductivity (W.m-1.K-1).

21 milieu_v2_base

Description: Basic class for medium (physics properties of medium) composed of constituents (fluids and solids).

See also: `objet_u` ([35](#)) `fluide_diphasique` ([21](#))

Usage:

21.1 fluide_diphasique

Description: Two-phase fluid.

See also: `milieu_v2_base` ([21](#))

Usage:

```
fluide_diphasique bloc
where
```

- **bloc** *bloc_lecture* ([2.40](#)): Two-phase fluid description.

22 modele_rayonnement_base

Description: Basic class for wall thermal radiation model.

See also: `objet_u` ([35](#)) `modele_rayonnement_milieu_transparent` ([22](#))

Usage:

22.1 modele_rayonnement_milieu_transparent

Description: Wall thermal radiation model for a transparent gas and resolving a radiation-conduction-thermohydraulics coupled problem in VDF or VEF.

Modele_Rayonnement_Milieu_Transparent mod

Read mod {

nom_pb_rayonnant

problem_name

fichier_fij

file_name

fichier_face_rayo

file_name

[fichier_matrice | fichier_matrice_binaire file_name]

}

nom_pb_rayonnant problem_name : problem_name is the name of the radiating fluid problem

fichier_fij file_name : file_name is the name of the file which contains the shape factor matrix between all the faces.

fichier_face_rayo file_name : file_name is the name of the file which contains the radiating faces characteristics (area, emission value ...)

fichier_matrice|fichier_matrice_binaire file_name : file_name is the name of the ASCII (or binary) file which contains the inverted shape factor matrix. It is an optional keyword, if not defined, the inverted shape factor matrix will be calculated and written in a file.

The two first files can be generated by a preprocessor, they allow the radiating face characteristics to be entered (set of faces considered to be uniform with respect to radiation for emission value, flux, etc.) and the form factors for these various faces. These files have the following format:

File on radiating faces:

N M -> N nombre de faces rayonnantes (=bords) et

(N is the number of radiating faces (=edges) and

-> M nombre de faces rayonnantes a emissivitee non nulle

M equals the number of non-zero emission radiating faces

Nom(i) S(i) E(i) -> Nom du bord i, surface du bord i, valeur de

(Name of the edge i, surface area of the edge i)

-> l'emissivite (comprise entre 0 et 1) (emission value (between 0 and 1))

Exemple:

13 4

Gauche 50.0 0.0

Droit1 50.0 0.5

Bas 10.0 0.0

Haut 10.0 0.0

Arriere 5.0 0.0

Avant 5.0 0.0

Droit2 30.0 0.5

Bas1 40.0 0.0

Haut1 20.0 0.0

Avant1 20.0 0.0

Arriere1 20.0 0.0

Entree 20.0 0.5

Sortie 20.0 0.5

File on form factors:

N -> Nombre de faces rayonnantes (Number of radiating faces)

Fij -> Matrice des facteurs de formes avec i,j entre 1 et N (Matrix of form factors where i, j between 1 and N)

Example:

13

```

1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.24 0.20 0.10 0.10 0.10 0.10 0.16
0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.40 0.00 0.00 0.00 0.00 0.00 0.20 0.10 0.10 0.10 0.10 0.00
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.00 0.15 0.10 0.10 0.15 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.30 0.00 0.10 0.10 0.00 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.20 0.10 0.00 0.10 0.10 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.20 0.10 0.10 0.00 0.10 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.30 0.00 0.10 0.10 0.00 0.10
0.00 0.40 0.00 0.00 0.00 0.00 0.00 0.20 0.10 0.10 0.10 0.10 0.00

```

Caution:

- a) The radiation model's precision is decided by the user when he/she names the domain edges. In fact, a radiating face is recognised by the preprocessor as the set of domain edges faces bearing the same name. Thus, if the user subdivides the edge into two edges which are named differently, he/she thus creates two radiating faces instead of one.
- b) The form factors are entered by the user, the preprocessor carries out no calculations other than checking preservation relationships on form factors.
- c) The fluid is considered to be a transparent gas.

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

See also: `modele_rayonnement_base` (22)

Usage:

modele_rayonnement_milieu_transparent bloc

where

- **bloc** *bloc_lecture* (2.40): See description.

23 modele_turbulence_scal_base

Description: Basic class for turbulence model for energy equation.

See also: `objet_u` (35) `prandtl` (23.1) `schmidt` (23.2) `sous_maille_dyn` (23.3) `fluctuation_temperature_w-_bas_re` (23)

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* (32): 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».

23.1 fluctuation_temperature_w_bas_re

Description: a_reprendre

See also: modele_turbulence_scal_base (23)

Usage:

```
fluctuation_temperature_w_bas_re obj Lire obj {
    [ transport_fluctuation_temperature_w_bas_re bloc_lecture]
    [ modele_fonc_bas_reynolds_thermique deuxmots]
    [ turbulence_paro turbulence_paro_scalaire_base]
    [ dt_impr_nusselt float]
}
```

where

- **transport_fluctuation_temperature_w_bas_re** *bloc_lecture* (2.40): Transport equation for the temperature fluctuation.
- **modele_fonc_bas_reynolds_thermique** *deuxmots* (4.24.25): Choice of the coefficient (Jones Lauder).
- **turbulence_paro** *turbulence_paro_scalaire_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_nusselt** *float* for inheritance: Keyword to print local values of Nusselt number and temperature near a wall during a turbulent calculation. The values will be printed in the _Nusselt.face file each dt_impr_nusselt time period. The local Nusselt expression is as follows : $Nu = ((\lambda + \lambda_t)/\lambda) * d_{wall}/d_{eq}$ where d_{wall} is the distance from the first mesh to the wall and d_{eq} is given by the wall law. This option also gives the value of d_{eq} and $h = (\lambda + \lambda_t)/d_{eq}$ and the fluid temperature of the first mesh near the wall.
For the Neumann boundary conditions (flux_impose), the «equivalent» wall temperature given by the wall law is also printed (Tparoi equiv.) preceded for VEF calculation by the edge temperature «T face de bord».

23.2 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 (23)

Usage:

```
prandtl obj Lire obj {
    [ prdt str]
    [ prandtl_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
- **prandtl_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* (32) 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».

23.3 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` (23)

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* (32) 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».

23.4 sous_maille_dyn

Description: Dynamic sub-grid turbulence modele.

Warning : Available in VDF only. Not coded in VEF yet.

See also: modele_turbulence_scal_base (23)

Usage:

```
sous_maille_dyn obj Lire obj {  
    [ stabilise str into ['6_points', 'moy_euler', 'plans_paralleles']]  
    [ nb_points int]  
    [ turbulence_paroit turbulence_paroit_scalaire_base]  
    [ dt_impr_nusselt float]  
}
```

where

- **stabilise** *str into* ['6_points', 'moy_euler', 'plans_paralleles']
- **nb_points** *int*
- **turbulence_paroit** *turbulence_paroit_scalaire_base* (32) for inheritance: Keyword to set the wall law.
- **dt_impr_nusselt** *float* for inheritance: Keyword to print local values of Nusselt number and temperature near a wall during a turbulent calculation. The values will be printed in the _Nusselt.face file each dt_impr_nusselt time period. The local Nusselt expression is as follows : $Nu = ((\lambda + \lambda_t)/\lambda) * d_{wall}/d_{eq}$ where d_{wall} is the distance from the first mesh to the wall and d_{eq} is given by the wall law. This option also gives the value of d_{eq} and $h = (\lambda + \lambda_t)/d_{eq}$ and the fluid temperature of the first mesh near the wall.
For the Neumann boundary conditions (flux_impose), the «equivalent» wall temperature given by the wall law is also printed (Tparoi equiv.) preceded for VEF calculation by the edge temperature «T face de bord».

24 nom

Description: Class to name the TRUST objects.

See also: objet_u (35) nom_anonyme (24)

Usage:

```
nom [ mot ]
```

where

- **mot** *str*: Chain of characters.

24.1 nom_anonyme

Description: not_set

See also: nom (24)

Usage:

```
[ mot ]
```

where

- **mot** *str*: Chain of characters.

25 partitionneur_deriv

Description: not_set

See also: objet_u (35) metis (25.1) sous_zones (25.3) tranche (25.4) partition (25.2) fichier_decoupage (25)

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).

25.1 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 (25)

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).

25.2 metis

Description: Metis is an external partitionning library. It is a general algorithm that will generate a partition of the domain.

See also: `partitionneur_deriv` (25)

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).

25.3 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` (25)

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).

25.4 sous_zones

Description: This algorithm will create one part for each specified subzone. All elements contained in the first subzone are put in the first part, all remaining elements contained in the second subzone in the second part, etc...

If all elements of the domain are contained in the specified subzones, then N parts are created, otherwise, a

supplemental part is created with the remaining elements.

If no subzone is specified, all subzones defined in the domain are used to split the mesh.

See also: `partitionneur_deriv` (25)

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).

25.5 tranche

Description: This algorithm will create a geometrical partitionning by slicing the mesh in the two or three axis directions, based on the geometric center of each mesh element. *nz* must be given if *dimension=3*. Each slice contains the same number of elements (slices don't have the same geometrical width, and for VDF meshes, slice boundaries are generally not flat except if the number of mesh elements in each direction is an exact multiple of the number of slices). First, *nx* slices in the X direction are created, then each slice is split in *ny* slices in the Y direction, and finally, each part is split in *nz* slices in the Z direction. The resulting number of parts is *nx*ny*nz*. If one particular direction has been declared periodic, the default slicing (0, 1, 2, ..., *n-1*) is replaced by (0, 1, 2, ..., *n-1*, 0), each of the two '0' slices having twice less elements than the other slices.

See also: `partitionneur_deriv` (25)

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 precondition_base

Description: Basic class for preconditioning.

See also: `objet_u` (35) `ssor` (26.2) `ssor_bloc` (26.3) `precondsolv` (26.1) `precond_local` (26)

Usage:

26.1 **precond_local**

Description: This keyword can be used with the conjugate gradient (GCP) to choose a local preconditionment for parallel calculation (ie: Cholesky, SSOR,...).

See also: [precond_base \(26\)](#)

Usage:

precond_local **solveur**

where

- **solveur** *solveur_sys_base* ([9.11](#)): Solver type.

26.2 **precondsolv**

Description: not_set

See also: [precond_base \(26\)](#)

Usage:

precondsolv **solveur**

where

- **solveur** *solveur_sys_base* ([9.11](#)): Solver type.

26.3 **ssor**

Description: Symmetric successive over-relaxation algorithm.

See also: [precond_base \(26\)](#)

Usage:

ssor obj Lire obj {

omega *float*

}

where

- **omega** *float*: Over-relaxation facteur (between 1 and 2, optimal value around 1.5-1.6).

26.4 **ssor_bloc**

Description: not_set

See also: [precond_base \(26\)](#)

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* ([26](#))
- **alpha_1** *float*
- **precond1** *precond_base* ([26](#))
- **alpha_a** *float*
- **preconda** *precond_base* ([26](#))

27 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](#) ([35](#)) [scheme_euler_explicit](#) ([27.2](#)) [schema_predictor_corrector](#) ([27.17](#)) [Sch_CN_iteratif](#) ([27.1](#)) [runge_kutta_ordre_3](#) ([27.5](#)) [runge_kutta_ordre_4_d3p](#) ([27.6](#)) [leap_frog](#) ([27.3](#)) [runge_kutta_rationnel_ordre_2](#) ([27.7](#)) [schema_implicite_base](#) ([27.15](#)) [schema_adams_bashforth_order_2](#) ([27.8](#)) [schema_adams_bashforth_order_3](#) ([27.9](#)) [schema_phase_field](#) ([27.16](#))

Usage:

```

schema_temps_base obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicite int into [0, 1]]
    [ niter_max_diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
where

```

- **tinit** *float*: Value of initial calculation time (0 by default).
- **tmax** *float*: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float*: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).

- **dt_min** *float*: Minimum calculation time step (1e-16s by default).
- **dt_max** *float*: Maximum calculation time step (1e30s by default).
- **facsec** *float*: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3
- **nb_pas_dt_max** *int*: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float*: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt Impr** *float*: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4): dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float*: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]*
- **diffusion_implicit** *int into [0, 1]*: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int*: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float*: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]*: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision Impr** *int*: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]*
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]*
- **periode_sauvegarde_securite_en_heures** *int*: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** : To disable the check of the available amount of disk space during the calculation.

27.1 Sch_CN_EX_iteratif

Description: This keyword also describes a Crank-Nicholson method of second order accuracy but here, for scalars, because of instabilities encountered when $dt > dt_{CFL}$, the Crank Nicholson scheme is not applied to scalar quantities. Scalars are treated according to Euler-Explicite scheme at the end of the CN treatment for velocity flow fields (by doing p Euler explicite under-iterations at $dt \leq dt_{CFL}$). Parameters

are the same (but default values may change) compare to the Sch_CN_iterative scheme plus a relaxation keyword: niter_min (2), niter_max (6), niter_avg (3), facsec_max (20), seuil (0.05)

See also: Sch_CN_iteratif ([27.1](#))

Usage:

Sch_CN_EX_iteratif obj Lire obj {

```
[ omega float]
[ niter_min int]
[ niter_max int]
[ niter_avg int]
[ facsec_max float]
[ seuil float]
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

where

- **omega** *float*: relaxation factor (0.1)
- **niter_min** *int* for inheritance: minimal number of p-iterations to satisfy convergence criteria (2 by default)
- **niter_max** *int* for inheritance: number of maximum p-iterations allowed to satisfy convergence criteria (6)
- **niter_avg** *int* for inheritance: threshold of p-iterations (3). If the number of p-iterations is greater than niter_avg, facsec is reduced, if lesser than niter_avg, facsec is increased (but limited by the facsec_max value).
- **facsec_max** *float* for inheritance: maximum ratio allowed between dynamical time step returned by iterative process and stability time returned by CFL condition (2).
- **seuil** *float* for inheritance: criteria for ending iterative process ($\text{Max}(\|u(p) - u(p-1)\|/\text{Max} \|u(p)\|) < \text{seuil}$) (0.001)
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).

- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.2 Sch_CN_iteratif

Description: The Crank-Nicholson method of second order accuracy. A mid-point rule formulation is used (Euler-centered scheme). The basic scheme is: $u(t+1) = u(t) + du/dt(t+1/2)*dt$. The estimation of the time

derivative du/dt at the level $(t+1/2)$ is obtained either by iterative process. The time derivative du/dt at the level $(t+1/2)$ is calculated iteratively with a simple under-relaxations method. Since the method is implicit, neither the cfl nor the fourier stability criteria must be respected. The time step is calculated in a way that the iterative procedure converges with the less iterations as possible.

Remark : for stationary or RANS calculations, no limitation can be given for time step through high value of `facsec_max` parameter (for instance : `facsec_max 1000`). In counterpart, for LES calculations, high values of `facsec_max` may engender numerical instabilities.

See also: `schema_temps_base` (27) `Sch_CN_EX_iteratif` (27)

Usage:

Sch_CN_iteratif obj Lire obj {

```
[ niter_min  int]
[ niter_max  int]
[ niter_avg  int]
[ facsec_max float]
[ seuil      float]
[ tinit      float]
[ tmax       float]
[ tcpumax    float]
[ dt_min     float]
[ dt_max     float]
[ facsec     float]
[ nb_pas_dt_max int]
[ dt_sauv    float]
[ dt_impr    float]
[ dt_start   dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

where

- **niter_min** *int*: minimal number of p-iterations to satisfy convergence criteria (2 by default)
- **niter_max** *int*: number of maximum p-iterations allowed to satisfy convergence criteria (6)
- **niter_avg** *int*: threshold of p-iterations (3). If the number of p-iterations is greater than `niter_avg`, `facsec` is reduced, if lesser than `niter_avg`, `facsec` is increased (but limited by the `facsec_max` value).
- **facsec_max** *float*: maximum ratio allowed between dynamical time step returned by iterative process and stability time returned by CFL condition (2).
- **seuil** *float*: criteria for ending iterative process ($\text{Max}(\|u(p) - u(p-1)\|/\text{Max}\|u(p)\|) < \text{seuil}$) (0.001)
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).

- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.3 scheme_euler_explicit

Synonymous: **schema_euler_explicite**

Description: This is the Euler explicite scheme.

See also: `schema_temps_base` (27)

Usage:

```
scheme_euler_explicit obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max float]  
    [ facsec float]  
    [ nb_pas_dt_max int]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ dt_start dt_start]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int into [0, 1]]  
    [ diffusion_implicite int into [0, 1]]  
    [ niter_max_diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int into [0, 1]]  
    [ precision_impr int]  
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]  
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example `Schema_Adams_Bashforth_order_3`
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every `dt_sauv`, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the `.out` file.
- **dt_start** *dt_start* (9.4) for inheritance: `dt_min` : the first iteration is based on `dt_min`
`dt_start dt_calc` : the time step at first iteration is calculated in agreement with CFL condition.
`dt_start dt_fixe` value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on `dt_calc`.

- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/G_i of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, *vrel* should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_convection$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a *facsec* that is too large. Start with a *facsec* of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_max$.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.4 leap_frog

Description: This is the leap-frog scheme.

See also: [schema_temps_base \(27\)](#)

Usage:

```
leap_frog obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicit int into [0, 1]]
    [ niter_max_diffusion_implicit int]
```

```

[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
where

```

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.

- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicite** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.5 rk3_ft

Description: Keyword for Runge Kutta time scheme for Front_Tracking calculation.

See also: `runge_kutta_ordre_3` ([27.5](#))

Usage:

```
rk3_ft obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicite int into [0, 1]]
    [ niter_max_diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth_order_3

- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.6 runge_kutta_ordre_3

Description: This is the Runge-Kutta scheme of third order.

See also: schema_temps_base (27) rk3_ft (27.4)

Usage:

runge_kutta_ordre_3 obj Lire obj {

[**tinit** *float*]

[**tmax** *float*]

```

[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
where

```

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth_order_3
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection

on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a `facsec` that is too large. Start with a `facsec` of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore `dt=facsec*dt_max`.

- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergence criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into `.out` files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in `.sauv` file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.7 runge_kutta_ordre_4_d3p

Description: `not_set`

See also: `schema_temps_base` ([27](#))

Usage:

```
runge_kutta_ordre_4_d3p obj Lire obj {
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicit int into [0, 1]]
    [ niter_max_diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicit int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.8 runge_kutta_rationnel_ordre_2

Description: This is the Runge-Kutta rational scheme of second order.

See also: `schema_temps_base` (27)

Usage:

```
runge_kutta_rationnel_ordre_2 obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max float]  
    [ facsec float]  
    [ nb_pas_dt_max int]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ dt_start dt_start]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int into [0, 1]]  
    [ diffusion_implicite int into [0, 1]]  
    [ niter_max_diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int into [0, 1]]  
    [ precision_impr int]  
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]  
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
}  
where
```

- **tinit** float for inheritance: Value of initial calculation time (0 by default).
- **tmax** float for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** float for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** float for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** float for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** float for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example `Schema_Adams_Bashforth_order_3`
- **nb_pas_dt_max** int for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** float for inheritance: Save time step value (1e30s by default). Every `dt_sauv`, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt_impr** float for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the `.out` file.
- **dt_start** dt_start (9.4) for inheritance: `dt_min` : the first iteration is based on `dt_min`
`dt_start dt_calc` : the time step at first iteration is calculated in agreement with CFL condition.
`dt_start dt_fixe` value : the first time step is fixed by the user (recommended when restarting calcula-

tion with Crank Nicholson temporal scheme to ensure continuity).

By default, the first iteration is based on `dt_calc`.

- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/G_i of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, `vrel` should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_convection$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a `facsec` that is too large. Start with a `facsec` of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_max$.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into `.out` files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in `.sauv` file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.9 schema_adams_bashforth_order_2

Description: `not_set`

See also: `schema_temps_base` ([27](#))

Usage:

schema_adams_bashforth_order_2 obj Lire obj {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
```

```

[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]

```

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int* into [0, 1] for inheritance
- **diffusion_implicit** *int* into [0, 1] for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.

- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.10 schema_adams_bashforth_order_3

Description: not_set

See also: schema_temps_base (27)

Usage:

schema_adams_bashforth_order_3 obj Lire obj {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.

Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3

- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.11 schema_adams_moulton_order_2

Description: not_set

See also: schema_implicit_base (27.15)

Usage:

```
schema_adams_moulton_order_2 obj Lire obj {
    [ facsec_max float]
```

```

[ max_iter_implicit int]
solveur solveur_implicit_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}

```

where

- **facsec_max** *float* for inheritance: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by **facsec** keyword is changed during the calculation with the implicit scheme but it couldn't be higher than **facsec_max** value.
Warning: Some implicit schemes do not permit high **facsec_max**, example Schema_Adams_Moulton_order_3 needs **facsec=facsec_max=1**.
Advice:
The calculation may start with a **facsec** specified by the user and increased by the algorithm up to the **facsec_max** limit. But the user can also choose to specify a constant **facsec** (**facsec_max** will be set to **facsec** value then). Faster convergence has been seen and depends on the kind of calculation:
-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), **facsec** between 20-30
-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), **facsec** between 90-100
-Thermohydraulic with natural convection, **facsec** around 300
-Conduction only, **facsec** can be set to a very high value (1e8) as if the scheme was unconditionally stable
These values can also be used as rule of thumb for initial **facsec** with a **facsec_max** limit higher.
- **max_iter_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (28) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. **solver** is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicit (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
Advice: Since the 1.6.0 version, we recommend to use first the 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 Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicite** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicite** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.

- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.12 schema_adams_moulton_order_3

Description: not_set

See also: schema_implicite_base ([27.15](#))

Usage:

```
schema_adams_moulton_order_3 obj Lire obj {
    [ facsec_max float]
    [ max_iter_implicite int]
    solveur solveur_implicite_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicite int into [0, 1]]
    [ niter_max_diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
where
```

- **facsec_max** *float* for inheritance: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.

Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.

Advice:

The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:

- Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
- Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
- Thermohydraulic with natural convection, facsec around 300

-Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.

- **max_iter_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (28) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solveur* is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every *dt_sauv*, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: *dt_min* : the first iteration is based on *dt_min*
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on *dt_calc*.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/G_i of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, *vrel* should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.

- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicite** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.13 schema_backward_differentiation_order_2

Description: not_set

See also: schema_implicite_base (27.15)

Usage:

schema_backward_differentiation_order_2 obj Lire obj {

```
[ facsec_max float]
[ max_iter_implicite int]
solveur solveur_implicite_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicite int into [0, 1]]
[ niter_max_diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
[ no_conv_subiteration_diffusion_implicite int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
```

}

where

- **facsec_max** *float* for inheritance: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation

with the implicit scheme but it couldn't be higher than `facsec_max` value.

Warning: Some implicit schemes do not permit high `facsec_max`, example `Schema_Adams_Moulton_order_3` needs `facsec=facsec_max=1`.

Advice:

The calculation may start with a `facsec` specified by the user and increased by the algorithm up to the `facsec_max` limit. But the user can also choose to specify a constant `facsec` (`facsec_max` will be set to `facsec` value then). Faster convergence has been seen and depends on the kind of calculation:

-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value `beta` low), `facsec` between 20-30

-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value `beta` high), `facsec` between 90-100

-Thermohydraulic with natural convection, `facsec` around 300

-Conduction only, `facsec` can be set to a very high value ($1e8$) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial `facsec` with a `facsec_max` limit higher.

- **max_iter_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (28) 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 ($1e30$ s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped ($1e30$ s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step ($1e30$ s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the `facsec` to 0.5.
Warning: Some schemes needs a `facsec` lower than 1 (0.5 is a good start), for example `Schema_Adams_Bashforth_order_3`
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps ($1e9$).
- **dt_sauv** *float* for inheritance: Save time step value ($1e30$ s by default). Every `dt_sauv`, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time ($1e30$ s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the `.out` file.
- **dt_start** *dt_start* (9.4) for inheritance: `dt_min` : the first iteration is based on `dt_min`
`dt_start dt_calc` : the time step at first iteration is calculated in agreement with CFL condition.
`dt_start dt_fixe` value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on `dt_calc`.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold ($1e-12$ by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported

values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.

- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, *vrel* should be set to 1). The stability time step is then only based on the convection time step ($dt = facsec * dt_convection$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a *facsec* that is too large. Start with a *facsec* of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore $dt = facsec * dt_max$.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value ($1e-6$) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.14 schema_backward_differentiation_order_3

Description: not_set

See also: [schema_implicit_base \(27.15\)](#)

Usage:

schema_backward_differentiation_order_3 obj Lire obj {

```
[ facsec_max float]
[ max_iter_implicit int]
solveur solveur_implicit_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
```

```

[ niter_max_diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
[ no_conv_subiteration_diffusion_implicite int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
where

```

- **facsec_max** *float* for inheritance: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.

Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.

Advice:

The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:

- Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
- Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
- Thermohydraulic with natural convection, facsec around 300
- Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.

- **max_iter_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (28) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solveur* is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).

- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.15 scheme_euler_implicit

Synonymous: **schema_euler_implicit**

Description: This is the Euler implicite scheme.

See also: **schema_implicit_base** (27.15)

Usage:

```
scheme_euler_implicit obj Lire obj {  
    [ facsec_max float]
```



```

[ max_iter_implicit int]
solveur solveur_implicit_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max float]
[ facsec float]
[ nb_pas_dt_max int]
[ dt_sauv float]
[ dt_impr float]
[ dt_start dt_start]
[ seuil_statio float]
[ seuil_statio_relatif_deconseille int into [0, 1]]
[ diffusion_implicit int into [0, 1]]
[ niter_max_diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int into [0, 1]]
[ precision_impr int]
[ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
[ no_conv_subiteration_diffusion_implicit int into [0, 1]]
[ periode_sauvegarde_securite_en_heures int]
[ no_check_disk_space ]
}
where

```

- **facsec_max** *float* for inheritance: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by **facsec** keyword is changed during the calculation with the implicit scheme but it couldn't be higher than **facsec_max** value.
Warning: Some implicit schemes do not permit high **facsec_max**, example Schema_Adams_Moulton_order_3 needs **facsec=facsec_max=1**.
Advice:
The calculation may start with a **facsec** specified by the user and increased by the algorithm up to the **facsec_max** limit. But the user can also choose to specify a constant **facsec** (**facsec_max** will be set to **facsec** value then). Faster convergence has been seen and depends on the kind of calculation:
-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), **facsec** between 20-30
-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), **facsec** between 90-100
-Thermohydraulic with natural convection, **facsec** around 300
-Conduction only, **facsec** can be set to a very high value (1e8) as if the scheme was unconditionally 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* (28) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. **solver** is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicit (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
Advice: Since the 1.6.0 version, we recommend to use first the 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 Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicite** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicite** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.

- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.16 schema_implicite_base

Description: Basic class for implicate time scheme.

See also: [schema_temps_base \(27\)](#) [scheme_euler_implicit \(27.14\)](#) [schema_adams_moulton_order_2 \(27.10\)](#) [schema_adams_moulton_order_3 \(27.11\)](#) [schema_backward_differentiation_order_2 \(27.12\)](#) [schema_backward_differentiation_order_3 \(27.13\)](#)

Usage:

```
schema_implicite_base obj Lire obj {
    [ facsec_max float]
    [ max_iter_implicite int]
    solveur solveur_implicite_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicite int into [0, 1]]
    [ niter_max_diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
```

where

- **facsec_max float**: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.

Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.

Advice:

The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:

- Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
- Thermal hydraulic with forced convection and strong coupling between velocity and temperature

(Boussinesq value beta high), facsec between 90-100

-Thermohydraulic with natural convection, facsec around 300

-Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.

- **max_iter_implicit** *int*: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (28): This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solveur* is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.

Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.

Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3

- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every *dt_sauv*, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: *dt_min* : the first iteration is based on *dt_min*
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).

By default, the first iteration is based on *dt_calc*.

- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/G_i of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, *vrel* should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore

dt=facsec*dt_max.

- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.17 schema_phase_field

Description: Keyword for the only available Scheme for time discretization of the Phase Field problem.

See also: [schema_temps_base \(27\)](#)

Usage:

```

schema_phase_field obj Lire obj {
    [ schema_ch schema_temps_base]
    [ schema_ns schema_temps_base]
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max float]
    [ facsec float]
    [ nb_pas_dt_max int]
    [ dt_sauv float]
    [ dt_impr float]
    [ dt_start dt_start]
    [ seuil_statio float]
    [ seuil_statio_relatif_deconseille int into [0, 1]]
    [ diffusion_implicit int into [0, 1]]
    [ niter_max_diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ impr_diffusion_implicit int into [0, 1]]
    [ precision_impr int]
    [ no_error_if_not_converged_diffusion_implicit int into [0, 1]]
    [ no_conv_subiteration_diffusion_implicit int into [0, 1]]
    [ periode_sauvegarde_securite_en_heures int]
    [ no_check_disk_space ]
}
where

```

- **schema_ch** *schema_temps_base* ([27](#)): Time scheme for the Cahn-Hilliard equation.

- **schema_ns** *schema_temps_base* (27): Time scheme for the Navier-Stokes equation.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *float* for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** *float* for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** *dt_start* (9.4) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision Impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

27.18 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` ([27](#))

Usage:

```
schema_predictor_corrector obj Lire obj {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max float]  
    [ facsec float]  
    [ nb_pas_dt_max int]  
    [ dt_sauv float]  
    [ dt_impr float]  
    [ dt_start dt_start]  
    [ seuil_statio float]  
    [ seuil_statio_relatif_deconseille int into [0, 1]]  
    [ diffusion_implicite int into [0, 1]]  
    [ niter_max_diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int into [0, 1]]  
    [ precision_impr int]  
    [ no_error_if_not_converged_diffusion_implicite int into [0, 1]]  
    [ no_conv_subiteration_diffusion_implicite int into [0, 1]]  
    [ periode_sauvegarde_securite_en_heures int]  
    [ no_check_disk_space ]  
}
```

where

- **tinit** float for inheritance: Value of initial calculation time (0 by default).
- **tmax** float for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** float for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** float for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** float for inheritance: Maximum calculation time step (1e30s by default).
- **facsec** float for inheritance: Value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth_order_3
- **nb_pas_dt_max** int for inheritance: Maximum number of calculation time steps (1e9).
- **dt_sauv** float for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion.
- **dt_impr** float for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **dt_start** dt_start ([9.4](#)) for inheritance: dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.

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

By default, the first iteration is based on dt_calc.

- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/Gi of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **seuil_statio_relatif_deconseille** *int into [0, 1]* for inheritance
- **diffusion_implicit** *int into [0, 1]* for inheritance: Keyword to make the diffusion term in the Navier Stokes equation implicit (in this case, vrel should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int into [0, 1]* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **no_error_if_not_converged_diffusion_implicit** *int into [0, 1]* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int into [0, 1]* for inheritance
- **periode_sauvegarde_securite_en_heures** *int* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

28 solveur_implicit_base

Description: Class for solver in the situation where the time scheme is the implicit scheme. Solver allows equation diffusion and convection operators to be set as implicit terms.

See also: objet_u (35) solveur_lineaire_std (28.4) simplr (28.3)

Usage:

28.1 implicit

Description: similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.

See also: piso (28.1)

Usage:

```
implicit obj Lire obj {
```

```
    [ seuil_convergence_implicit float]
```



```

[ nb_corrections_max int]
[ seuil_convergence_solveur float]
[ seuil_generation_solveur float]
[ seuil_verification_solveur float]
[ seuil_test_preliminaire_solveur float]
[ solveur solveur_sys_base]
[ no_qdm ]
[ nb_it_max int]
[ controle_residu ]

```

}

where

- **seuil_convergence_implicit** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections than nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier-Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error $\|Ax-B\|$ is lesser than vrel after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than vrel.
- **solveur** *solveur_sys_base* (9.11) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

28.2 piso

Description: Piso (Pressure Implicit with Split Operator) - method to solve N_S.

See also: simpler (28.3) implicite (28) simple (28.2)

Usage:

piso obj Lire obj {

```

[ seuil_convergence_implicit float]
[ nb_corrections_max int]
[ seuil_convergence_solveur float]
[ seuil_generation_solveur float]
[ seuil_verification_solveur float]
[ seuil_test_preliminaire_solveur float]
[ solveur solveur_sys_base]
[ no_qdm ]

```



```

[ nb_it_max int]
[ controle_residu ]

```

```

}

```

where

- **seuil_convergence_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* (9.11) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

28.3 simple

Description: SIMPLE type algorithm

See also: piso (28.1)

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_implicite** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections than nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error $\|Ax-B\|$ is lesser than vrel after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than vrel.
- **solveur** *solveur_sys_base* (9.11) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

28.4 simpler

Description: Simpler method for incompressible systems.

See also: solveur_implicite_base (28) piso (28.1)

Usage:

```
simpler obj Lire obj {
    seuil_convergence_implicite float
    [ seuil_convergence_solveur float]
    [ seuil_generation_solveur float]
    [ seuil_verification_solveur float]
    [ seuil_test_preliminaire_solveur float]
    [ solveur solveur_sys_base]
    [ no_qdm ]
    [ nb_it_max int]
    [ controle_residu ]
}
```

where

- **seuil_convergence_implicite** *float*: Keyword to set the value of the convergence criteria for the resolution of the implicit system build to solve either the Navier_Stokes equation (only for Simple and Simpler algorithms) or a scalar equation. It is advised to use the default value (1e6) to solve the implicit system only once by time step. This value must be decreased when a coupling between problems is considered.
- **seuil_convergence_solveur** *float*: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).

- **seuil_generation_solveur** *float*: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than vrel).
- **seuil_verification_solveur** *float*: Option to check if residual error $\|Ax-B\|$ is lesser than vrel after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float*: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than vrel.
- **solveur** *solveur_sys_base* (9.11): Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** : Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** : Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

28.5 solveur_lineaire_std

Description: not_set

See also: solveur_implicite_base (28)

Usage:

solveur_lineaire_std obj Lire obj {

 [**solveur** *solveur_sys_base*]

}

where

- **solveur** *solveur_sys_base* (9.11)

29 source_base

Description: Basic class of source terms introduced in the equation.

See also: objet_u (35) source_generique (29.20) boussinesq_temperature (29.3) boussinesq_concentration (29.2) dirac (29.7) puissance_thermique (29.16) source_qdm_lambdaup (29.23) source_th_tdivu (29.29) source_robin (29.26) source_robin_scalaire (29.27) canal_perio (29.4) source_constituant (29.18) source_transport_k_eps (29.31) acceleration (29.1) coriolis (29.5) source_qdm (29.22) perte_charge_singuliere (29.15.2) perte_charge_directionnelle (29.11) perte_charge_isotrope (29.12) perte_charge_anisotrope (29.9) perte_charge_circulaire (29.10) darcy (29.6) forchheimer (29.8) perte_charge_reguliere (29.13) source_transport_k_eps_bas_reynolds (29.34) source_qdm_phase_field (29.24) source_con_phase_field (29.17) source_rayo_semi_transp (29.25) trainee (29.30) flottabilite (29.19) masse_ajoutee (29.21)

Usage:

29.1 Source_Transport_K_Eps_anisotherme

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transportation equation. By default, these constants are set to: C1_eps=1.44 C2_eps=1.92 C3_eps=1.0

See also: source_transport_k_eps (29.31)

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.

29.2 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 \(29\)](#)

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): 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): 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): 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): 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): Keyword to specify the centre of rotation (expressed in R' coordinates) of R' into R (if the domain rotates with the R' referential, the centre of rotation is $0'=(0,0,0)$). The `time_field` should have 2 or 3 components according the dimension 2 or 3.
- **option** str into ['terme_complet', 'coriolis_seul', 'entrainement_seul']: Keyword to specify the kind of calculation: `terme_complet` (default option) will calculate both the Coriolis and centrifugal forces, `coriolis_seul` will calculate the first one only, `entrainement_seul` will calculate the second one only.

29.3 boussinesq_concentration

Description: Class to describe a source term that couples the movement quantity equation and constituent transportation equation with the Boussinesq hypothesis.

See also: [source_base \(29\)](#)

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.

29.4 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 \(29\)](#)

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.

29.5 canal_perio

Description: Momentum source term to maintain flow rate. The expression of the source term is:

$$S(t) = (2*(Q(0) - Q(t)) - (Q(0) - Q(t-dt)))/(coeff*dt*area)$$

Where:

coeff=damping coefficient

area=area of the periodic boundary

Q(t)=flow rate at time t

dt=time step

Three files will be created during calculation on a datafile named DataFile.data. The first file contains the flow rate evolution. The second file is useful for restarting a calculation with the flow rate of the previous stopped calculation, and the last one contains the pressure gradient evolution:

-DataFile_Channel_Flow_Rate_ProblemName_BoundaryName

-DataFile_Channel_Flow_Rate_repr_ProblemName_BoundaryName

-DataFile_Pressure_Gradient_ProblemName_BoundaryName

See also: [source_base \(29\)](#)

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.

29.6 coriolis

Description: Keyword for a Coriolis term in hydraulic equation. Warning: Only available in VDF.

See also: [source_base \(29\)](#)

Usage:

coriolis **omega**

where

- **omega** *str*: Value of omega.

29.7 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 \(29\)](#)

Usage:

darcy **bloc**

where

- **bloc** *bloc_lecture (2.40)*: Description.

29.8 dirac

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: [source_base \(29\)](#)

Usage:

dirac position ch

where

- **position** *n x1 x2 ... xn*
- **ch** *champ_base (16)*: 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.

29.9 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 \(29\)](#)

Usage:

forchheimer bloc

where

- **bloc** *bloc_lecture (2.40)*: Description.

29.10 perte_charge_anisotrope

Description: Anisotropic pressure loss.

See also: [source_base \(29\)](#)

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 (16.1)*: Hydraulic diameter value.
- **direction** *champ_don_base (16.1)*: Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

29.11 perte_charge_circulaire

Description: New pressure loss.

See also: [source_base \(29\)](#)

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* (16.1): Hydraulic diameter value.
- **diam_hydr_ortho** *champ_don_base* (16.1): Transverse hydraulic diameter value.
- **direction** *champ_don_base* (16.1): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

29.12 perte_charge_directionnelle

Description: Directional pressure loss.

See also: [source_base \(29\)](#)

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* (16.1): Hydraulic diameter value.
- **direction** *champ_don_base* (16.1): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

29.13 perte_charge_isotrope

Description: Isotropic pressure loss.

See also: [source_base \(29\)](#)

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* (16.1): Hydraulic diameter value.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

29.14 perte_charge_reguliere

Description: Source term modelling the presence of a bundle of tubes in a flow.

See also: [source_base \(29\)](#)

Usage:

```

perte_charge_reguliere spec zone_name
where

```

- **spec** *spec_pdc_base* (29.14): Description of longitudinale or transversale type.
- **zone_name** *str*: Name of the sub-area occupied by the tube bundle. A Sous_Zone (Sub-area) type object called *zone_name* should have been previously created.

29.15 spec_pdc_base

Description: Class to read the source term modelling the presence of a bundle of tubes in a flow. $C_f = A \text{Re} \cdot B$.

See also: [objet_lecture \(34\)](#) [longitudinale \(29.15\)](#) [transversale \(29.15.1\)](#)

Usage:

```

spec_pdc_base ch_a a [ ch_b ] [ b ]
where

```

- **ch_a** *str into* [*'a'*, *'cf'*]: Keyword to be used to set law coefficient values for the coefficient of regular pressure losses.
- **a** *float*: Value of a law coefficient for regular pressure losses.
- **ch_b** *str into* [*'b'*]: Keyword to be used to set law coefficient values for regular pressure losses.
- **b** *float*: Value of a law coefficient for regular pressure losses.

29.15.1 longitudinale

Description: Class to define the pressure loss in the direction of the tube bundle.

See also: [spec_pdc_base \(29.14\)](#)

Usage:

```

longitudinale dir dd ch_a a [ ch_b ] [ b ]
where

```

- **dir** *str into* [*'x'*, *'y'*, *'z'*]: Direction.

- **dd** *float*: Tube bundle hydraulic diameter value. This value is expressed in m.
- **ch_a** *str into ['a', 'cf']*: Keyword to be used to set law coefficient values for the coefficient of regular pressure losses.
- **a** *float*: Value of a law coefficient for regular pressure losses.
- **ch_b** *str into ['b']*: Keyword to be used to set law coefficient values for regular pressure losses.
- **b** *float*: Value of a law coefficient for regular pressure losses.

29.15.2 transversale

Description: Class to define the pressure loss in the direction perpendicular to the tube bundle.

See also: `spec_pdcr_base` (29.14)

Usage:

transversale dir dd chaine_d d ch_a a [ch_b] [b]

where

- **dir** *str into ['x', 'y', 'z']*: Direction.
- **dd** *float*: Value of the tube bundle step.
- **chaine_d** *str into ['d']*: Keyword to be used to set the value of the tube external diameter.
- **d** *float*: Value of the tube external diameter.
- **ch_a** *str into ['a', 'cf']*: Keyword to be used to set law coefficient values for the coefficient of regular pressure losses.
- **a** *float*: Value of a law coefficient for regular pressure losses.
- **ch_b** *str into ['b']*: Keyword to be used to set law coefficient values for regular pressure losses.
- **b** *float*: Value of a law coefficient for regular pressure losses.

29.16 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` (29)

Usage:

perte_charge_singuliere dir coeff bloc_definition_surface

where

- **dir** *str into ['kx', 'ky', 'kz']*: KX, KY or KZ designate directional pressure loss coefficients for respectively X, Y or Z direction.
- **coeff** *float*: Value of friction coefficient (KX, KY, KZ).
- **bloc_definition_surface** *bloc_lecture* (2.40): Surface definition block : In VDF, the surface area definition syntax is identical to that used to define sides (edges) in the Block, for example { X = x0 y0 <= Y <= y1 } for a line perpendicular to the Ox axis in a two-dimensional domain, or { Y = y0 x0 <= X <= x1 z0 <= Z <= z1 } for a surface perpendicular to the Oy axis in a 3D domain.
example : `sources { Perte_Charge_Singuliere KX 0.5 { X = 1. 0. <= Y <= 1. } }`

VEF : the surface area definition syntax relies on sub-areas definition (see 4.3.22). First value (X=0.35 in the example below, in regard to KX keyword) allows to determine the faces of elements in sub-area for which the pressure loss is applied.

example : `sources { Perte_Charge_Singuliere KX 0.5 { 0.35 sous_zone_toto } }`

Observations :

- If the surface area is not included in the calculation domain or if (in VDF) it is not perpendicular to the space direction in accordance with which the pressure loss is being calculated, Trio-U exists in error.
- The surface area may be diminished at only one side if a sudden shrinking or widening occurs.

29.17 puissance_thermique

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: `source_base` (29)

Usage:

puissance_thermique **ch**

where

- **ch** *champ_base* (16): 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).

29.18 source_con_phase_field

Description: Keyword to define the source term of the Cahn-Hilliard equation.

See also: `source_base` (29)

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
    implication_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 beta of the model.
- **kappa** *float*: Mobility coefficient kappa0.
- **kappa_variable** *str into ['oui', 'non']*: To define a mobility which depends on concentration C.
- **moyenne_de_kappa** *str*: To define how mobility kappa is calculated on faces of the mesh according to cell-centered values (chaîne is arithmetique/harmonique/geometrique).
- **multiplicateur_de_kappa** *float*: To define the parameter of the mobility expression when mobility depends on C.
- **couplage_NS_CH** *str*: Evaluating time choosen for the term source calculation into the Navier Stokes equation (chaîne is mutilde(n+1/2)/mutilde(n), in order to be conservative, the first choice seems better).
- **implication_CH** *str into ['oui', 'non']*: To define if the Cahn-Hilliard will be solved using a implicit algorithm or not.
- **gmres_non_lineaire** *str into ['oui', 'non']*: To define the algorithm to solve Cahn-Hilliard equation (oui: Newton-Krylov method, non: fixed point method).
- **seuil_cv_iterations_ptfixe** *float*: Convergence threshold (an option of the fixed point method).
- **seuil_residu_ptfixe** *float*: Threshold for the matrix inversion used in the method (an option of the fixed point method).
- **seuil_residu_gmresnl** *float*: Convergence threshold (an option of the Newton-Krylov method).
- **dimension_espace_de_krylov** *int*: Vector numbers used in the method (an option of the Newton-Krylov method).
- **nb_iterations_gmresnl** *int*: Maximal iteration (an option of the Newton-Krylov method).
- **residu_min_gmresnl** *float*: Minimal convergence threshold (an option of the Newton-Krylov method).
- **residu_max_gmresnl** *float*: Maximal convergence threshold (an option of the Newton-Krylov method).

29.19 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 \(29\)](#)

Usage:

source_constituant ch

where

- **ch** *champ_base (16)*: Field type.

29.20 flottabilite

Description: buoyancy effect

See also: [source_base \(29\)](#)

Usage:

flottabilite

29.21 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 \(29\)](#)

Usage:

source_generique champ

where

- **champ** *champ_generique_base (7)*: the source field

29.22 masse_ajoutee

Description: weight added effect

See also: [source_base \(29\)](#)

Usage:

masse_ajoutee

29.23 source_qdm

Description: Momentum source term in the Navier Stokes equation.

See also: [source_base \(29\)](#)

Usage:

source_qdm ch

where

- **ch** *champ_base (16)*: Field type.

29.24 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 \(29\)](#)

Usage:

source_qdm_lambdaup obj Lire obj {

```
    lambda float  
    [ lambda_min float]  
    [ lambda_max float]  
    [ ubar_umprim_cible float]
```

}

where

- **lambda** *float*: value of lambda
- **lambda_min** *float*: value of lambda_min
- **lambda_max** *float*: value of lambda_max
- **ubar_umprim_cible** *float*: value of ubar_umprim_cible

29.25 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 \(29\)](#)

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).

29.26 source_rayo_semi_transp

Description: Radiative term source in energy equation.

See also: [source_base \(29\)](#)

Usage:

```
source_rayo_semi_transp
```

29.27 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 \(29\)](#)

Usage:

```
source_robin bords
```

where

- **bords** *vect_nom* ([2.103](#))

29.28 source_robin_scalaire

Description: This source term should be used when a Paroi_decalee_Robin boundary condition is set in an energy equation. The source term will be applied on the N specified boundaries. The values temp_wall_valueI are the temperature specified on the Ith boundary. The last value dt_impr is a printing period which is mandatory to specify in the data file but has no effect yet.

See also: [source_base \(29\)](#)

Usage:

```
source_robin_scalaire bords
```

where

- **bords** *listdeuxmots_sacc* ([29.28](#))

29.29 listdeuxmots_sacc

Description: List of groups of two words (without accodances).

See also: listobj ([33.3](#))

Usage:

n object1 object2

list of *deuxmots* ([4.24.25](#))

29.30 source_th_tdivu

Description: This term source is dedicated for any scalar (called T) transportation. Coupled with upwind (amont) or muscl scheme, this term gives for final expression of convection : $\text{div}(\mathbf{U} \cdot \mathbf{T}) - \mathbf{T} \cdot \text{div}(\mathbf{U}) = \mathbf{U} \cdot \text{grad}(\mathbf{T})$. This ensures, in incompressible flow when divergence free is badly resolved, to stay in a better way in the physical boundaries.

Warning: Only available in VEF discretization.

See also: source_base ([29](#))

Usage:

source_th_tdivu

29.31 trainee

Description: drag effect

See also: source_base ([29](#))

Usage:

trainee

29.32 source_transport_k_eps

Description: Keyword to alter the source term constants in the standard k-eps model epsilon transportation equation. By default, these constants are set to: C1_eps=1.44 C2_eps=1.92

See also: source_base ([29](#)) Source_Transport_K_Eps_anisotherme ([29](#)) source_transport_k_eps_aniso_concen ([29.32](#)) source_transport_k_eps_aniso_therm_concen ([29.33](#))

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.

29.33 source_transport_k_eps_aniso_concen

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transportation equation. By default, these constants are set to: C1_eps=1.44 C2_eps=1.92 C3_eps=1.0

See also: source_transport_k_eps ([29.31](#))

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.

29.34 source_transport_k_eps_aniso_therm_concen

Description: Keywords to modify the source term constants in the anisotherm standard k-eps model epsilon transportation equation. By default, these constants are set to: C1_eps=1.44 C2_eps=1.92 C3_eps=1.0

See also: source_transport_k_eps ([29.31](#))

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.

29.35 source_transport_k_eps_bas_reynolds

Description: Keywords to modify the source term constants in the model's epsilon transportation equation. By default, these constants are set to: C1_eps=1.55 C2_eps=2.

See also: source_base ([29](#))

Usage:

source_transport_k_eps_bas_reynolds obj Lire obj {

[**c1_eps** *float*]

[**c2_eps** *float*]

}
where

- **c1_eps** *float*: First constant.
- **c2_eps** *float*: Second constant.

30 sous_zone

Description: It is an object type describing a domain sub-set.

A Sous_Zone (Sub-area) type object must be associated with a Domaine type object. The Lire (Read) interpreter is used to define the items comprising the sub-area.

Caution: The Domain type object nom_domaine must have been meshed (and triangulated or tetrahedralised in VEF) prior to carrying out the Associer (Associate) nom_sous_zone nom_domaine instruction; this instruction must always be preceded by the read instruction.

See also: objet_u (35)

Usage:

```
sous_zone obj Lire obj {
    [ restriction str]
    [ rectangle bloc_origine_cotes]
    [ segment bloc_origine_cotes]
    [ boite bloc_origine_cotes]
    [ liste n n1 n2 ... nn]
    [ fichier str]
    [ intervalle deuxentiers]
    [ polynomes bloc_lecture]
    [ couronne bloc_couronne]
    [ tube bloc_tube]
    [ fonction_sous_zone str]
    [ union str]
}
```

where

- **restriction** *str*: The elements of the sub-area nom_sous_zone must be included into the other sub-area named nom_sous_zone2. This keyword should be used first in the Lire keyword.
- **rectangle** *bloc_origine_cotes* (30): The sub-area will include all the domain elements whose centre of gravity is within the Rectangle (in dimension 2).
- **segment** *bloc_origine_cotes* (30)
- **boite** *bloc_origine_cotes* (30): 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* (4.24.10): The sub-area will include domain items whose number is between n1 and n2 (where n1<=n2).
- **polynomes** *bloc_lecture* (2.40): A REPENDRE
- **couronne** *bloc_couronne* (30.1): In 2D case, to create a couronne.
- **tube** *bloc_tube* (30.2): In 3D case, to create a tube.
- **fonction_sous_zone** *str*: Keyword to build a sub-area with the the elements included into the area defined by fonction>0.
- **union** *str*: The elements of the sub-area nom_sous_zone3 will be added to the sub-area nom_sous_zone. This keyword should be used last in the Lire keyword.

30.1 bloc_origine_cotes

Description: Class to create a rectangle (or a box).

See also: [objet_lecture \(34\)](#)

Usage:

name origin name2 cotes

where

- **name** *str* into [*'Origine'*]: Keyword to define the origin of the rectangle (or the box).
- **origin** *x1 x2 (x3)*: Co-ordinates of the origin of the rectangle (or the box).
- **name2** *str* into [*'Cotes'*]: Keyword to define the length along the axes.
- **cotes** *x1 x2 (x3)*: Length along the axes.

30.2 bloc_couronne

Description: Class to create a couronne (2D).

See also: [objet_lecture \(34\)](#)

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.

30.3 bloc_tube

Description: Class to create a tube (3D).

See also: [objet_lecture \(34\)](#)

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.

31 turbulence_paroil_base

Description: Basic class for wall laws for NAVIER STOKES equations.

See also: objet_u (35) loi_standard_hydr_old (31.5) loi_standard_hydr (31.4) paroi_tble (31.8) negligible (31.7) utau_imp (31.12) loi_puissance_hydr (31.3) loi_paroil_2_couches (31.2)

Usage:

31.1 loi_ciofalo_hydr

Description: A Loi_ciofalo_hydr law for wall turbulence for NAVIER STOKES equations.

See also: loi_standard_hydr (31.4)

Usage:

loi_ciofalo_hydr

31.2 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 (31.4)

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)

31.3 loi_paroil_2_couches

Description: Standard law of the wall for turbulence model k-eps at two layers for a hydraulic problem.

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

Usage:

loi_paro_i_2_couches

31.4 loi_puissance_hydr

Description: A Loi_puissance_hydr law for wall turbulence for NAVIER STOKES equations.

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

Usage:

31.5 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_i_base \(31\)](#) [loi_expert_hydr \(31.1\)](#) [loi_ww_hydr \(31.6\)](#) [loi_ciofalo_hydr \(31\)](#)

Usage:

loi_standard_hydr

31.6 loi_standard_hydr_old

Description: not_set

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

Usage:

loi_standard_hydr_old

31.7 loi_ww_hydr

Description: laws have been qualified on channel calculation

See also: [loi_standard_hydr \(31.4\)](#)

Usage:

31.8 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_i_base \(31\)](#)

Usage:

negligeable

31.9 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 \(31\)](#)

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* ([31.9](#)): 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* ([31.10](#))
- **restart**
- **stationnaire** *entierfloat* ([31.11.1](#))
- **lambda** *str*
- **mu** *str*
- **sans_source_boussinesq**
- **alpha** *float*
- **kappa** *float*

31.10 twofloat

Description: two reals.

See also: [objet_lecture \(34\)](#)

Usage:

```
a b  
where
```

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

31.11 liste_sonde_tble

Description: not_set

See also: listobj ([33.3](#))

Usage:

n object1 object2

list of *sonde_tble* ([31.11](#))

31.11.1 sonde_tble

Description: not_set

See also: objet_lecture ([34](#))

Usage:

name point

where

- **name** *str*
- **point** *un_point* ([2.10.2](#))

31.12 entierfloat

Description: An integer and a real.

See also: objet_lecture ([34](#))

Usage:

the_int the_float

where

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

31.13 utau_imp

Description: Keyword to impose the friction velocity on the wall with a turbulence model for thermohydraulic problems. There are two possibilities to use this keyword :

1 - we can impose directly the value of the friction velocity u_{star} .

2 - we can also give the friction coefficient and hydraulic diameter. So, TRUST determines the friction velocity by : $u_{star} = U \cdot \sqrt{\lambda_c/8}$.

See also: turbulence_paro_base ([31](#))

Usage:

utau_imp obj Lire obj {

[**u_tau** *champ_base*]

[**lambda_c** *str*]

[**diam_hydr** *champ_base*]

}

where

- **u_tau** *champ_base* (16): 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): The hydraulic diameter.

32 turbulence_paroil_scalaire_base

Description: Basic class for wall laws for energy equation.

See also: [objet_u](#) (35) [loi_standard_hydr_scalaire](#) (32.6) [loi_analytique_scalaire](#) (32.1) [paroi_tble_scal](#) (32.8) [loi_paroil_nu_impose](#) (32.5) [negligeable_scalaire](#) (32.7) [loi_WW_scalaire](#) (32) [loi_odvm](#) (32.3) [loi_paroil_2_couches_scalaire](#) (32.4)

Usage:

32.1 loi_WW_scalaire

Description: not_set

See also: [turbulence_paroil_scalaire_base](#) (32)

Usage:

loi_WW_scalaire

32.2 loi_analytique_scalaire

Description: not_set

See also: [turbulence_paroil_scalaire_base](#) (32)

Usage:

loi_analytique_scalaire

32.3 loi_expert_scalaire

Description: Keyword similar to keyword [Loi_standard_hydr_scalaire](#) but with additional option.

See also: [loi_standard_hydr_scalaire](#) (32.6)

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.

32.4 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_ODVM_VDF). This law is also available with isothermal walls.

See also: turbulence_paro_scalaire_base (32)

Usage:

loi_odvm obj Lire obj {

```
    n    int
    gamma float
    [ stats floatfloat ]
    [ check_files ]
```

}

where

- **n** *int*: Number of points per face in the 1D uniform meshes. **n** should be chosen in order to have the first point situated near $\Delta y=1/3$.
- **gamma** *float*: Smoothing parameter of the signal between 10e-5 (no smoothing) and 10e-1 (high averaging).
- **stats** *floatfloat* (4.24.28): 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.

32.5 loi_paro_2_couches_scalaire

Description: Standard law of the wall for turbulence model k-eps at two layers for a thermohydraulic problem.

See also: turbulence_paro_scalaire_base (32)

Usage:

loi_paro_2_couches_scalaire

32.6 loi_paro_nu_impose

Description: Keyword to impose Nusselt numbers on the wall for the thermohydraulic problems. To use this option, it is necessary to give in the data file the value of the hydraulic diameter and the expression of the Nusselt number.

See also: turbulence_paro_scalaire_base (32)

Usage:

loi_paro_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): The hydraulic diameter.

32.7 loi_standard_hydr_scalaire

Description: Keyword for the law of the wall.

See also: turbulence_paroil_scalaire_base (32) loi_expert_scalaire (32.2)

Usage:

loi_standard_hydr_scalaire

32.8 negligeable_scalaire

Description: Keyword to suppress the calculation of a law of the wall with a turbulence model for thermo-hydraulic problems. The wall stress is directly calculated with the derivative of the velocity, in the direction perpendicular to the wall.

See also: turbulence_paroil_scalaire_base (32)

Usage:

negligeable_scalaire

32.9 paroi_tble_scal

Description: Keyword for the Thin Boundary Layer Equation thermal wall-model.

See also: turbulence_paroil_scalaire_base (32)

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* (32.9): 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* (31.10)
- **prandtl** *float*

32.10 fourfloat

Description: Four reals.

See also: `objet_lecture` (34)

Usage:

a b c d

where

- **a** *float*: First real.
- **b** *float*: Second real.
- **c** *float*: Third real.
- **d** *float*: Fourth real.

33 listobj_impl

Description: `not_set`

See also: `objet_u` (35) `listobj` (33.3)

Usage:

33.1 list_un_pb

Description: pour les groupes

See also: `listobj` (33.3)

Usage:

{ `object1` , `object2` }

list of `un_pb` (33.1) separated with ,

33.2 un_pb

Description: pour les groupes

See also: `objet_lecture` (34)

Usage:

mot

where

- **mot** *str*: la chaîne

33.3 listdeuxmots

Description: List of groups of two words.

See also: listobj (33.3)

Usage:

```
{ object1 object2 .... }  
list of deuxmots (4.24.25)
```

33.4 listobj

Description: List of objects.

See also: listobj_impl (33) champs_a_post (3.2.17) list_stat_post (3.2.20) listpoints (3.2.6) sondes (3.2.2) listchamp_generique (7.2) list_nom_virgule (7.1) definition_champs (3.2) post_processings (3.2.28) liste_post (3.4.4) liste_post_ok (3.3.1) condlims (3.10) sources (4.3.1) vect_nom (2.103) list_nom (2.88) list_bord (2.50.3) list_bloc_mailler (2.49) list_un_pb (33) list_list_nom (3.7) ecrire_fichier_xyz_valeur_param (4.4) pp (4.16) listdeuxmots_sacc (29.28) liste_sonde_tble (31.10) listeqn (3.11) list_info_med (3.36) listsous_zone_valeur (4.8.11) reactions (8) listdeuxmots (33.2)

Usage:

34 objet_lecture

Description: Auxiliary class for reading.

See also: objet_u (35) bloc_lecture (2.40) deuxmots (4.24.25) format_file (3.5.3) deuxentiers (4.24.10) floatfloat (4.24.28) entierfloat (31.11.1) champ_a_post (3.2.18) champs_posts (3.2.16) stat_post_deriv (3.2.21) stats_posts (3.2.19) stats_serie_posts (3.2.27) sonde_base (3.2.4) un_point (2.10.2) sonde (3.2.3) definition_champ (3.2.1) postraitement_base (3.4.1) un_postraitement (3.3) type_un_post (3.5.1) type_postraitement_ft_lata (3.5.2) un_postraitement_spec (3.5) nom_postraitement (3.4) condinit (4.3) condinits (4.2.9) condlimlu (3.10.1) mailler_base (2.50) bloc_pave (2.50.2) defbord (2.50.6) bord_base (2.50.4) parametre_equation_base (4.5.2) un_pb (33.1) bords_ecrire (4.5.1) ecrire_fichier_xyz_valeur_item (4.5) convection_deriv (4.8) bloc_convection (4.7) diffusion_deriv (4.2) op_implicite (4.2.8) bloc_diffusion (4.1) traitement_particulier_base (4.26) traitement_particulier (4.25) penalisation_l2_ftd_lec (4.17) dt_impr_ustar_mean_only (4.24) modele_turbulence_hyd_deriv (4.23) paroi_ft_disc_deriv (12.58) bloc_sutherland (20.4) form_a_nb_points (4.24.3) modele_fonction_bas_reynolds_base (4.24.19) fourfloat (32.9) twofloat (31.9) sonde_tble (31.11) remove_elem_bloc (2.76) lecture_bloc_moment_base (2.9) bloc_origine_cotes (30) bloc_couronne (30.1) bloc_tube (30.2) bloc_lecture_poro (2.60) bloc_lec_champ_init_canal_sinal (16.11) fonction_champ_reprise (16.7) bloc_decouper (2.57) troisi (2.34) spec_pdc_base (29.14) format_lata_to_med (2.45) info_med (3.37) methode_transport_deriv (4.32) bloc_ef (4.8.8) sous_zone_valeur (4.8.12) bloc_diffusion_standard (4.2.6) reaction (8.1) floatentier (4.24.11) floattantchaine (34) threefloat (34.1) eq_rayo_semi_transp (3.9) bloc_lecture_remaillage (4.33.3) objet_lecture_maintien_temperature (4.18) interpolation_champ_face_deriv (4.35) parcours_interface (4.34) injection_marqueur (4.38) penalisation_forcage (4.22) ceg_areva (4.26.11) ceg_cea_jaea (4.26.12)

Usage:

34.1 floattantchaine

Description: A real and a chain.

See also: `objet_lecture` ([34](#))

Usage:

the_float name

where

- **the_float** *float*: Real.
- **name** *str*: Chain.

34.2 threefloat

Description: Three reals.

See also: `objet_lecture` ([34](#))

Usage:

a b c

where

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

35 index

Index

/*, 183
#, 203

, 103, 106, 110, 158
associer, 17
champ_post_statistiques_correlation, 68, 186
champ_post_statistiques_ecart_type, 68, 187
champ_post_statistiques_moyenne, 67, 190
champ_uniforme, 235
decouper, 41, 257
discretiser, 23
divergence, 186
ecrire_fichier, 58
extraction, 187
fin, 30
gradient, 188
interpolation, 189
lire, 45
lire_fichier, 46
lire_fichier_bin, 46
lire_med, 47
morceau_equation, 189
operateur_eqn, 185
postraitement, 70
postraitements, 69
raffiner_simplexes, 45
rectify_mesh, 47
reduction_0d, 191
refchamp, 192
resoudre, 52
schema_euler_explicite, 265
schema_euler_implicite, 287
tparoi_vef, 192
transformation, 193
6_points, 150, 151, 255
≤, 36, 37
=, 36
A, 205, 206
a, 305, 306
amont, 114
analytique, 173, 175
ancien, 109, 116
antisym, 112
arrete, 134–151
avec_energie_cinetique, 123
avec_les_cl, 130, 131, 164–169, 171
avec_sources, 130, 131, 164–169, 171
avec_sources_et_operateurs, 130, 131, 164–169, 171
b, 305, 306
binaire, 23, 65, 72, 228
bords, 107

C, 249
C_ext, 206, 209, 210
centre, 114
cf, 305, 306
chakravarthy, 114
champ_frontiere, 188
chsom, 61
composante, 193
conservation_masse, 248, 249
constant, 248, 249
coriolis_seul, 300
Cotes, 314
d, 306
debit_total, 32
default, 189
defaut_bar, 105, 112
dir, 314
distant, 37
divrhout_moins_Tdivrhout, 109, 116
divuT_moins_Tdivu, 109, 116
dt_integr, 69
dt_post, 65, 66
edo, 248, 249
elem, 40, 65, 68, 228
emissivite, 205, 206
entrainement_seul, 300
faces, 65, 68
family_names_from_group_names, 47
filtrer_resu, 105, 106, 112
Fluctu_Temperature_ext, 206, 209, 210
flux_bords, 189, 190
Flux_Chaleur_Turb_ext, 206, 209, 210
fonction, 229
format_post_sup, 33
formatte, 23, 65, 72, 228
formule, 193
grad_i, 129, 130
grad_Ubar, 105, 106
grav, 61
hauteur, 314
homogene, 37
implicite, 106
initiale, 173, 175
integrale_en_z, 32
k, 222
K_Eps_ext, 206, 209, 210
kx, 306
ky, 306
kz, 306
last_time, 228

lata , 33, 44, 60, 70
lata_v1 , 33, 44, 60, 70
lata_v2 , 33, 44, 60, 70
lml , 33, 44, 60, 70
local , 37
max , 191
med , 33, 44, 60, 70
meshtv , 33, 44, 60, 70
min , 191
minmod , 114
modifiee , 173, 175
moins_rho_moyen , 248, 249
moy_euler , 150, 151, 255
moyenne , 191
moyenne_ponderee , 191
mu0 , 249
muscl , 114
nb_pas_dt_post , 65, 66
no , 180, 189
nodes , 61
non , 41, 164, 307, 308
norme , 193
norme_l2 , 191
nu , 105, 106
nu_transp , 105, 106
nut , 105, 106
nut_transp , 105, 106
one_way_coupling , 181
Origine , 314
oui , 41, 164, 307, 308
periode , 61
plans_paralleles , 150, 151, 255
post_processing , 72
postraitement , 72
postraitement_ft_lata , 72
postraitement_lata , 72
produit_scalaire , 193
que_les_faces_des_elts_dirichlet , 315
re , 314
rho_g , 129, 130
ri , 314
sans_energie_cinetique , 123
sans_rien , 130, 131, 164–169, 171
scotti , 134–151
short_family_names , 47
simplifiee , 173, 175
Slambda , 249
solveur , 106
som , 40, 61, 65, 68, 228
somme , 191
somme_ponderee , 191
stabilite , 189, 190
standard , 248, 249
suivi , 181
superbee , 114
T0 , 249
T_ext , 206, 209, 210
terme_complet , 300
toutes_les_faces_accrochees , 315
trace , 188
transportant_bar , 112
transporte_bar , 112
two_way_coupling , 181
uniforme , 173, 175
use_existing_domain , 228
V2_ext , 206, 209, 210
valeur_a_elem , 173, 174
valeur_normale , 241
vanalbada , 114
vanleer , 114
vdf_lineaire , 173, 174
vecteur , 193
vef , 47
vitesse_interpoele , 181
vitesse_paroie , 222
vitesse_particules , 181
vitesse_tangentielle , 244
volume , 134–151
volume_sans_lissage , 134–151
X , 36, 37, 51, 314
x , 305, 306
xyz , 72, 228
Y , 36, 37, 51, 314
y , 305, 306
yes , 180, 189
Z , 36, 37, 51, 314
z , 305, 306
, 103, 106, 110, 158
champs , 60, 70
conditions_initiales , 103, 109, 117–122, 124–126, 128, 129, 131, 165, 166, 168, 170, 172, 173, 180, 181
conditions_limites , 74, 103, 109, 117–122, 124–126, 128, 129, 131, 165, 166, 168, 170, 172, 175, 180, 182
fichier , 44
nom_zones , 42
partitionneur , 42
postraitement , 59, 73, 75, 76, 78–89, 91–98, 100, 101
postraitements , 59, 73, 75, 76, 78–88, 90–97, 99, 100, 102
save_matrice , 196–198, 202
sondes , 60, 70
1D , 160, 161
3D , 160, 161
A_plus , 315
acceleration , 300

alias , 118–120, 124
 alpha , 112, 113, 307, 317
 alpha_0 , 260
 alpha_1 , 260
 alpha_a , 260
 alpha_sous_zone , 113
 amont_sous_zone , 113
 ampli_bruit , 230
 ampli_sin , 230
 approximation_de_boussinesq , 164
 areva , 163
 ascii , 16, 53
 avec_certains_bords , 27
 avec_certains_bords_pour_extraire_surface , 27
 avec_les_bords , 27
 beta , 307
 beta_co , 247, 248
 beta_th , 247, 248
 binaire , 22, 44
 boite , 313
 bord , 20, 158, 302
 bords_a_decouper , 22
 boundaries , 133
 boundary_conditions , 74, 103, 109, 117–122, 124–126, 128, 129, 131, 165, 166, 168, 170, 172, 175, 180, 182
 boundary_xmax , 39
 boundary_xmin , 39
 boundary_ymax , 39
 boundary_ymin , 39
 boundary_zmax , 39
 boundary_zmin , 39
 btd , 116
 c , 163
 c0 , 301
 c1_eps , 300, 311–313
 c2_eps , 300, 311–313
 c3_eps , 300, 312
 calc_spectre , 160, 161
 calcul_ldp_en_flux_impose , 319
 canal , 143
 canalx , 140
 cea_jaea , 163
 centre_rotation , 300
 champ_med , 32
 changement_de_base_p1bulle , 225
 check_files , 320
 cl_pression_sommet_faible , 225
 clipping_courbure_interface , 130
 cmu , 152
 coeff , 302
 coefficient_diffusion , 247
 coefficients_activites , 194
 collisions , 174
 compo , 190
 concmoy , 161
 condition_elements , 26, 27
 condition_faces , 27
 condition_geometrique , 22
 conduction , 76
 conservation_Ec , 160, 161
 constante_modele_micro_melange , 194
 constante_taux_reaction , 194
 contre_energie_activation , 194
 contre_reaction , 194
 contribution_one_way , 181
 controle_residu , 197, 296–299
 convection , 109, 116, 118–122, 124–126, 128, 129, 131, 165, 166, 168, 170, 172, 175, 180, 181
 convection_diffusion_chaleur_qc , 92, 93
 convection_diffusion_chaleur_turbulent_qc , 96, 97
 convection_diffusion_concentration , 79, 80, 87, 88
 convection_diffusion_concentration_turbulent , 81, 82, 89, 91
 convection_diffusion_phase_field , 84
 convection_diffusion_temperature , 86–88, 94
 convection_diffusion_temperature_turbulent , 89, 91, 95, 98
 correction_parcours_thomas , 179
 correction_visco_turb_pour_controle_pas_de_temps , 133, 135, 137–139, 141–143, 145–147, 149–152, 154, 156, 157
 correction_visco_turb_pour_controle_pas_de_temps_parametre , 133, 135, 137–139, 141–143, 145–147, 149–152, 155–157
 corriger_partition , 256
 couplage_NS_CH , 308
 couronne , 313
 Cp , 246
 cp , 214, 215, 226, 246–250
 crank , 108
 critere_absolu , 29
 critere_arete , 178
 critere_longueur_fixe , 178
 critere_remaillage , 178
 cs , 138
 Cv , 246
 cw , 136
 d , 234, 236
 debit , 215
 debit_impose , 302
 debug , 163
 debut_stat , 159
 definition_champs , 60, 70
 delta , 214

derivee_rotation , 246
 dh , 215
 diag , 197
 diam_hydr , 303–305, 319, 321
 diam_hydr_ortho , 304
 diffusion , 103, 109, 117–122, 124–126, 128, 129, 131, 165, 166, 168, 170, 172, 175, 180, 181
 diffusion_implicite , 261, 263, 265, 267, 268, 270, 271, 273, 275, 276, 278, 280, 282, 285, 287, 289, 291, 293, 295
 dimension_espace_de_krylov , 308
 dir , 214, 215
 dir_flow , 230
 dir_wall , 230
 direction , 20, 28–30, 158, 303, 304
 distance_projete_faces , 175
 dmax , 140
 domain , 38
 domaine , 20, 21, 26, 27, 29, 30, 44, 60, 70, 188, 189, 257
 domaine_final , 20, 28
 domaine_flottant_fluide , 132
 domaine_grossier , 21
 domaine_init , 20, 28
 domaines , 44
 domegadt , 300
 dt_impr , 133, 214, 215, 261, 263, 265, 266, 268, 270, 271, 273, 274, 276, 278, 280, 282, 284, 287, 289, 291, 293, 294
 dt_impr_moy_spat , 159
 dt_impr_moy_temp , 159
 dt_impr_nusselt , 252–255
 dt_impr_ustar , 133, 135, 137–139, 141, 142, 144–146, 148–152, 155–157
 dt_impr_ustar_mean_only , 133, 135, 137–139, 141, 142, 144–146, 148–152, 155–157
 dt_injection , 182
 dt_max , 261, 263, 264, 266, 268, 269, 271, 273, 274, 276, 277, 280, 282, 284, 286, 289, 291, 293, 294
 dt_min , 260, 262, 264, 266, 268, 269, 271, 273, 274, 276, 277, 280, 282, 284, 286, 289, 291, 293, 294
 dt_post , 163
 dt_projection , 131, 164, 166, 168, 170, 171
 dt_sauv , 261, 263, 265, 266, 268, 270, 271, 273, 274, 276, 278, 280, 282, 284, 286, 289, 291, 293, 294
 dt_start , 261, 263, 265, 266, 268, 270, 271, 273, 274, 276, 278, 280, 282, 284, 287, 289, 291, 293, 294
 dt_uniforme , 183
 Ec , 159
 Ec_dans_repere_fixe , 159
 ecrire_decoupage , 42
 ecrire_fichier_xyz_valeur , 103, 109, 117–122, 124–126, 128, 129, 132, 165, 167, 168, 170, 172, 175, 180, 182
 ecrire_fichier_xyz_valeur_bin , 103, 109, 117–122, 124, 125, 127–129, 132, 165, 167, 168, 170, 172, 175, 180, 182
 ecrire_frontiere , 44
 ecrire_lata , 42
 emissivite_pour_rayonnement_entre_deux_plaques-quasi_infinies , 216
 energie_activation , 194
 ensemble_points , 182
 enthalpie_reaction , 194
 entreplat , 203
 epaisseur , 27, 29
 epaisseur_jeu , 203
 eps_min , 133, 135, 137, 138, 140–142, 144, 145, 147–151, 153, 155–157
 eq_rayo_semi_transp , 73
 eqnf22 , 156
 equation_frequence_resolue , 109
 equation_interface , 119, 126
 equation_interfaces_proprietes_fluide , 130
 equation_interfaces_vitesse_imposee , 130
 equation_navier_stokes , 126
 equation_non_resolue , 103, 109, 110, 117–119, 121–125, 127–129, 132, 165, 167, 168, 170, 172, 176, 180, 182
 equation_temperature_mpoint , 130
 equations_interfaces_vitesse_imposee , 130
 equations_scalaires_passifs , 75, 80, 82, 88, 91, 93, 94, 97, 98
 Erugu , 315
 erugu , 222
 espece , 121, 122
 espece_en_competition_micro_melange , 194
 exposant_beta , 194
 expression , 193
 facon_init , 160, 161
 facsec , 261, 263, 265, 266, 268, 269, 271, 273, 274, 276, 277, 280, 282, 284, 286, 289, 291, 293, 294
 facsec_max , 262, 264, 279, 281, 283, 286, 288, 290
 facteur , 115, 116, 317, 321
 facteur_longueur_ideale , 178
 facteurs , 35
 fichier , 60, 71, 140, 256, 313
 fichier_distance_paroie , 153
 fichier_ecriture_K_Eps , 140
 fichier_matrice , 53
 fichier_post , 20

fichier_secmem , 53
 fichier_solution , 53
 fichier_solveur , 53
 fichier_solveur_non_recree , 198
 fichier_sortie , 32
 fields , 60, 70
 file , 44
 file_coord_x , 39
 file_coord_y , 39
 file_coord_z , 39
 fin_stat , 159
 fonction , 49, 139
 fonction_filtre , 40
 fonction_sous_zone , 313
 format , 44, 60, 70
 format_post , 40
 formate , 42
 forme_du_terme_source , 310
 formulation_a_nb_points , 135, 136, 138, 139, 141–143, 145–148, 150, 151
 frequence_recale , 198
 frontiere , 162
 function_coord_x , 39
 function_coord_y , 39
 function_coord_z , 39
 gamma , 246, 320
 genere_fichier_solveur , 53
 ghost_thickness , 38
 gmres_non_lineaire , 308
 gravite , 164
 groupes , 73, 77, 101
 h , 230, 302
 haspi , 163
 hexa_old , 28
 implication_CH , 308
 implicite , 181
 impr , 53, 178, 195–198, 202
 impr_diffusion_implicite , 261, 263, 265, 267, 268, 270, 272, 273, 275, 276, 278, 280, 283, 285, 287, 289, 292, 293, 295
 indic_faces_modifiee , 175
 indice , 247–249
 info , 104
 init_Ec , 160, 161
 initial_conditions , 103, 109, 117–122, 124–126, 128, 129, 131, 165, 166, 168, 170, 172, 173, 180, 181
 initial_value , 231, 237
 injecteur_interfaces , 175
 injection , 181
 interfaces , 60, 71
 interpolation_champ_face , 175
 interpolation_repere_local , 175
 intervalle , 313
 inverse_condition_element , 27
 iterations_correction_volume , 174
 joints_non_postraites , 44
 k , 248
 k_min , 133, 135, 137, 138, 140–142, 144, 145, 147–151, 153, 155–157
 kappa , 247–249, 308, 315, 317
 kappa_variable , 308
 kmetis , 257
 lambda , 214, 215, 226, 246–250, 303–305, 309, 317
 lambda_c , 319
 lambda_max , 309
 lambda_min , 309
 lambda_ortho , 303, 304
 larg_joint , 42
 lissage_courbure_coeff , 178
 lissage_courbure_iterations , 178
 lissage_courbure_iterations_si_remaillage , 178
 lissage_courbure_iterations_systematique , 178
 liste , 49, 313
 liste_cas , 25
 liste_de_postraitements , 59, 73, 75, 76, 78–88, 90–97, 99, 100, 102
 liste_postraitements , 59, 73, 75, 76, 78–88, 90–97, 99, 100, 102
 localisation , 40, 189, 193
 loi_etat , 249
 longueur_boite , 160, 161
 longueur_maille , 135, 136, 138, 139, 141–143, 145–148, 150, 151
 longueurs , 35
 maillage , 174
 main , 43
 maintien_temperature , 126
 masse_molaire , 118–120, 123, 226
 matrice_pression_invariante , 131
 max_iter_implicite , 279, 282, 284, 286, 288, 291
 methode , 32, 188, 189, 191, 193
 methode_calcul_face_keps_impose , 315
 methode_calcul_pression_initiale , 131, 164, 166, 167, 169, 171
 methode_couplage , 181
 methode_interpolation_v , 174
 methode_transport , 174, 181
 min_critere_q_sur_max_critere_q , 163
 min_dir_flow , 230
 min_dir_wall , 230
 mode_calcul_convection , 109, 116
 modele_fonc_bas_reynolds , 152, 154
 modele_fonc_bas_reynolds_thermique , 253
 modele_micro_melange , 194
 modele_turbulence , 116, 120, 122, 128, 131, 169, 171

modele_visco , 317, 321
modif_div_face_dirichlet , 225
moyenne_convergee , 190
moyenne_de_kappa , 308
mu , 214, 215, 226, 247–249, 317
mu_1 , 123
mu_2 , 123
multiplicateur_de_kappa , 308
n , 215, 248, 317, 320, 321
n_iterations_distance , 174
n_iterations_interpolation_ibc , 175
name_of_initial_zones , 16
name_of_new_zones , 16
navier_stokes_phase_field , 84
navier_stokes_qc , 92, 93
navier_stokes_standard , 78–80, 86–88, 94
navier_stokes_turbulent , 81–83, 89, 90, 95, 98
navier_stokes_turbulent_qc , 96, 97
nb_comp , 231, 237, 321
nb_corrections_max , 296–298
nb_couronnes , 203
nb_it_max , 197, 296–299
nb_iter_barycentrage , 177
nb_iter_correction_volume , 178
nb_iter_remaillage , 177
nb_iteration_max_uzawa , 175
nb_iterations , 181
nb_iterations_gmresnl , 308
nb_mailles_mini , 163
nb_nodes , 38
nb_parts , 256–258
nb_parts_geom , 22
nb_parts_naif , 21
nb_parts_tot , 42
nb_pas_dt_max , 261, 263, 265, 266, 268, 269, 271, 273, 274, 276, 278, 280, 282, 284, 286, 289, 291, 293, 294
nb_points , 151, 255
nb_points_par_phase , 159
nb_procs , 25
nb_test , 53
nb_tranche , 32
nb_tranches , 28–30
nb_var , 139
new_jacobian , 104
niter_avg , 262, 264
niter_max , 262, 264
niter_max_diffusion_implicit , 108, 261, 263, 265, 267, 268, 270, 272, 273, 275, 276, 278, 280, 282, 285, 287, 289, 292, 293, 295
niter_min , 262, 264
no_check_disk_space , 261, 263, 265, 267, 269, 270, 272, 273, 275, 277, 278, 280, 283, 285, 287, 289, 292, 293, 295
no_conv_subiteration_diffusion_implicit , 261, 263, 265, 267, 269, 270, 272, 273, 275, 277, 278, 280, 283, 285, 287, 289, 292, 293, 295
no_error_if_not_converged_diffusion_implicit , 261, 263, 265, 267, 269, 270, 272, 273, 275, 277, 278, 280, 283, 285, 287, 289, 292, 293, 295
no_qdm , 296–299
nom , 231, 237
nom_bord , 28, 29, 203
nom_cl_derriere , 30
nom_cl_devant , 30
nom_domaine , 40
nom_fichier_post , 40
nom_fichier_solveur , 198
nom_fichier_sortie , 22
nom_frontiere , 188
nom_inconnue , 118–120, 123
nom_pb , 40
nom_source , 184–193
nombre_de_noeuds , 35
nombre_facettes_retenues_par_cellule , 175
noms_champs , 40
non_perio , 28
normal_value , 236
normalise , 163
nu , 104, 215, 216
nu_transp , 105
numero , 190, 193
numero_op , 185
numero_source , 185
nusselt , 321
nut , 105
nut_max , 133, 135, 137, 138, 140–142, 144, 145, 147–152, 155–157
nut_transp , 105
old , 113
omega , 230, 259, 262, 300
omega_relaxation_drho_dt , 249
optimisation_sous_maillage , 189
optimized , 196, 202
option , 119, 190, 300
Origine , 35
origine , 27
origine_numerotation , 203
p0 , 225
p1 , 225
p_imposee_aux_faces , 41
pa , 225
par_sous_zone , 20
parametre_equation , 103, 110, 117–119, 121–125, 127–129, 132, 165, 167, 168, 170, 172, 176, 180, 182

parcours_interface , 175
 Partition_tool , 42
 pas , 177
 pas_de_solution_initiale , 53
 pas_lissage , 177
 pb_champ , 191, 192
 pb_name , 43
 penalisation_forcage , 131
 penalisation_l2_ftd , 124, 126
 perio_x , 39
 perio_y , 39
 perio_z , 39
 periode , 159, 161
 periode_calc_spectre , 160, 161
 periode_sauvegarde_securite_en_heures , 261, 263, 265, 267, 269, 270, 272, 273, 275, 277, 278, 280, 283, 285, 287, 289, 292, 293, 295
 periodique , 42
 phase , 119, 126
 phase_marquee , 181
 point1 , 27
 point2 , 27
 point3 , 27
 polynomes , 313
 position , 246
 Post_processing , 59, 73, 75, 76, 78–89, 91–98, 100, 101
 Post_processings , 59, 73, 75, 76, 78–88, 90–97, 99, 100, 102
 potentiel_chimique_generalise , 123
 prandtl_turbulent_fonction_nu_t_alpha , 254
 Prandtl , 245, 246
 prandtl , 322
 prandtl_eps , 133, 135, 137, 138, 140–142, 144, 145, 147–151, 153, 155–157
 prandtl_k , 133, 135, 137, 138, 140–142, 144, 145, 147–151, 153, 155–157
 prdt , 254
 prdt_sur_kappa , 319
 precision_impr , 261, 263, 265, 267, 268, 270, 272, 273, 275, 277, 278, 280, 283, 285, 287, 289, 292, 293, 295
 precondition , 196, 202
 precondition0 , 260
 precondition1 , 260
 precondition_nul , 196, 202
 preconda , 260
 preconditionnement_diag , 108
 pression , 249
 pression_reference , 132
 Probes , 60, 70
 probleme , 26, 27, 203, 231, 237
 produits , 194
 projection_initiale , 131, 164, 166, 168, 169, 171
 projection_normale_bord , 29
 proprietes_particules , 182
 pulsation_w , 159
 quiet , 195–198, 202
 reactifs , 194
 reactions , 194
 rectangle , 313
 relax_barycentrage , 177
 relax_pression , 297
 remaillage , 174
 reorder , 42
 reprise , 59, 74, 75, 77–87, 89–96, 98–100, 102, 159
 reprise_correlation , 215, 216
 residu_max_gmresnl , 308
 residu_min_gmresnl , 308
 resolution_explicite , 109
 restart , 317
 restriction , 313
 resume_last_time , 60, 74, 76–83, 85–94, 96–99, 101, 102
 reynolds_stress_isotrope , 153
 rho , 214, 215, 246–250
 rho_1 , 123
 rho_2 , 123
 rho_constant_pour_debug , 246
 rotation , 246
 sans_passer_par_le2D , 28
 sans_solveur_masse , 185
 sans_source_boussinesq , 317
 sauvegarde , 59, 73, 75, 76, 78–87, 89–97, 99, 100, 102
 sauvegarde_simple , 59, 74, 75, 77–87, 89–96, 98–100, 102
 save_matrix , 196–198, 202
 Sc , 245
 schema_ch , 292
 schema_ns , 292
 scturb , 254
 segment , 313
 seuil , 196–198, 202, 262, 264
 seuil_convergence_implicit , 108, 296–298
 seuil_convergence_solveur , 109, 296–298
 seuil_convergence_uzawa , 175
 seuil_cv_iterations_ptfixe , 308
 seuil_diffusion_implicit , 108, 261, 263, 265, 267, 268, 270, 272, 273, 275, 276, 278, 280, 283, 285, 287, 289, 292, 293, 295
 seuil_divU , 131, 164, 166, 168, 170, 171
 seuil_dvolume_residuel , 178
 seuil_generation_solveur , 296–298
 seuil_residu_gmresnl , 308
 seuil_residu_ptfixe , 308

seuil_statio , 261, 263, 265, 266, 268, 270, 271, 273, 275, 276, 278, 280, 282, 284, 287, 289, 291, 293, 295
 seuil_statio_relatif_deconseille , 261, 263, 265, 267, 268, 270, 271, 273, 275, 276, 278, 280, 282, 285, 287, 289, 291, 293, 295
 seuil_test_preliminaire_solveur , 296–299
 seuil_verification , 53
 seuil_verification_solveur , 296–299
 solveur , 53, 74, 109, 279, 282, 284, 286, 288, 291, 296–299
 solveur0 , 196
 solveur1 , 196
 solveur_bar , 131, 164, 166, 168, 169, 171
 solveur_pression , 131, 164, 166, 168, 169, 171
 sonde_tble , 317, 322
 source , 184–193
 source_reference , 184–193
 sources , 103, 109, 117–122, 124–126, 128, 129, 131, 165, 166, 168, 170, 172, 175, 180, 182, 184–193
 sources_reference , 184–193
 sous_zone , 26, 231, 237, 303–305
 sous_zones , 258
 splitting , 38
 stabilise , 151, 255
 standard , 104
 stationnaire , 317
 statistiques , 60, 71
 statistiques_en_serie , 60, 71
 stats , 317, 320, 321
 stencil_width , 126
 surface , 216
 surfacique , 43
 sutherland , 249
 symx , 35
 symy , 35
 symz , 35
 t0 , 301
 t_deb , 162, 186, 187, 190
 t_debut_injection , 182
 t_fin , 163, 186, 187, 190
 tanh , 35
 tanh_dilatation , 35
 tanh_taille_premiere_maille , 35
 tcpumax , 260, 262, 264, 266, 268, 269, 271, 273, 274, 276, 277, 280, 282, 284, 286, 289, 291, 293, 294
 tdivu , 113
 temps_d_affichage , 307
 temps_debut_prise_en_compte_drho_dt , 249
 terme_gravite , 130
 test , 113, 203
 thi , 143
 tinf , 214, 215
 tinit , 260, 262, 264, 266, 268, 269, 271, 272, 274, 276, 277, 280, 282, 284, 286, 289, 291, 293, 294
 tmax , 260, 262, 264, 266, 268, 269, 271, 273, 274, 276, 277, 280, 282, 284, 286, 289, 291, 293, 294
 traitement_coins , 41
 traitement_particulier , 131, 165, 166, 168, 170, 172
 traitement_pth , 249
 traitement_rho_gravite , 249
 tranches , 258
 transformation_bulles , 181
 transport_fluctuation_temperature_w_bas_re , 253
 transport_k_epsilon , 152
 transport_k_epsilon_bas_reynolds , 154
 transport_k_epsilon_v2 , 156
 transport_k_kepsilon , 157
 transport_v2 , 156
 triangle , 27
 trois_tetra , 28
 tsup , 214, 215
 tube , 313
 turbulence_parois , 133, 135, 137–139, 141–143, 145, 146, 148–152, 155–157, 252–255
 tuyauz , 140
 tx1 , 162
 tx2 , 162
 tx3 , 162
 type , 190
 type_probleme , 203
 type_vitesse_imposee , 175
 u , 234, 236
 u_star_impose , 315
 u_tau , 318
 ubar_umprim_cible , 309
 ucent , 230
 union , 313
 use_weights , 257
 val_Ec , 160, 161
 verif_boussinesq , 301
 verif_dparois , 140
 via_extraire_surface , 27
 vintg_tetra , 28
 viscosite_dynamique_constante , 164
 vitesse , 246, 300
 vitesse_fluide_explicite , 179
 vitesse_imposee_regularisee , 175
 volume , 215
 volume_impose_phase_1 , 174
 volumes_etendus , 113
 volumes_non_etendus , 113
 volumique , 43

with_nu , 180
xinf , 216
xsup , 216
zmax , 32
zmin , 32
zones_name , 42

acceleration, 300
 ale, 116
 algo_base, 182
 algo_couple_1, 183
 amont, 110
 amont_old, 110
 analyse_angle, 16
 associate, 17
 associer_algo, 17
 associer_pbm_g_pbf, 17
 associer_pbm_g_pbgglobal, 18
 axi, 18

base, 179
 bidim_axi, 18
 bord, 35
 bord_base, 35
 boundary_field_inward, 236
 boundary_field_uniform_keps_from_ud, 236
 boussinesq_concentration, 300
 boussinesq_temperature, 301
 brech, 162
 btd, 115

calcul, 19
 calculer_moments, 18
 canal, 158
 canal_perio, 301
 ceg, 162
 centre, 111
 centre4, 111
 centre_de_gravite, 19
 centre_old, 111
 ch_front_input, 236
 ch_front_input_uniforme, 237
 champ_base, 226
 champ_don_base, 226
 champ_don_lu, 227
 champ_fonc_fonction, 227
 champ_fonc_fonction_txyz, 227
 champ_fonc_med, 228
 champ_fonc_reprise, 228
 champ_fonc_t, 229
 champ_fonc_tabule, 229
 champ_fonc_txyz, 233
 champ_fonc_xyz, 233
 champ_front_ale, 237
 champ_front_base, 235
 champ_front_bruite, 237
 champ_front_calc, 238
 champ_front_contact_rayo_semi_transp_vef, 238
 champ_front_contact_rayo_transp_vef, 238
 champ_front_contact_vef, 239
 champ_front_debit, 239
 champ_front_fonc_pois_ipsn, 239
 champ_front_fonc_pois_tube, 240
 champ_front_fonc_txyz, 240
 champ_front_fonc_xyz, 240
 champ_front_fonction, 240
 champ_front_lu, 241
 champ_front_normal_vef, 241
 champ_front_pression_from_u, 241
 champ_front_recyclage, 242
 champ_front_tabule, 243
 champ_front_tangentiel_vef, 244
 champ_front_uniforme, 244
 champ_front_vortex, 244
 champ_front_zoom, 244
 champ_generique_base, 183
 champ_init_canal_sinal, 229
 champ_input_base, 230
 champ_input_p0, 231
 champ_ostwald, 231
 champ_post_de_champs_post, 183
 champ_post_extraction, 187
 champ_post_interpolation, 188
 champ_post_morceau_equation, 189
 champ_post_operateur_base, 184
 champ_post_operateur_divergence, 186
 champ_post_operateur_eqn, 185
 champ_post_operateur_gradient, 188
 champ_post_reduction_0d, 191
 champ_post_refchamp, 191
 champ_post_statistiques_base, 185
 champ_post_tparoi_vef, 192
 champ_post_transformation, 192
 champ_som_lu_vdf, 231
 champ_som_lu_vef, 232
 champ_tabule_temps, 232
 champ_uniforme_morceaux, 232
 champ_uniforme_morceaux_tabule_temps, 233
 Champ_front_fonc_txyz, 15
 chimie, 193
 chmoy_faceperio, 161
 Cholesky, 198–200
 cholesky, 195
 circle, 64
 circle_3, 64
 class_generic, 195
 coeur, 202
 combinaison, 138

Concentration, 66, 68
 concmoy, 161
 condlim_base, 203
 condlims, 74
 conduction, 102
 constant, 221
 constituant, 246
 contact_vdf_vdf, 204
 contact_vdf_vdf, 204
 convection_deriv, 110
 convection_diffusion_chaleur_qc, 109
 convection_diffusion_chaleur_turbulent_qc, 116
 convection_diffusion_concentration, 117
 convection_diffusion_concentration_ft_disc, 118
 convection_diffusion_concentration_turbulent, 119
 convection_diffusion_fraction_massique_qc, 121
 convection_diffusion_fraction_massique_turbulent_qc, 122
 convection_diffusion_phase_field, 123
 convection_diffusion_temperature, 124
 convection_diffusion_temperature_ft_disc, 125
 convection_diffusion_temperature_turbulent, 127
 coriolis, 302
 Correlation, 66
 correlation, 68, 186
 corriger_frontiere_periodique, 19
 create_domain_from_sous_zone, 20

 darcy, 302
 debug, 20
 decoupebord_pour_rayonnement, 21
 decouper_bord_coincident, 22
 di_l2, 111
 diffusion_deriv, 103
 dilate, 22
 dimension, 22
 dirac, 302
 dirichlet, 204
 discretisation_base, 224
 discretiser_domaine, 22
 discretize, 23
 distance_parois, 23
 domain, 38
 domaine, 226
 domaine_ale, 226
 dt_calc, 195
 dt_fixe, 195
 dt_min, 195
 dt_start, 195
 Dt_post, 66

 ec, 159
 ecart_type, 68, 187
 Ecart_type, 66, 68

 echange_contact_rayo_transp_vdf, 204
 ecrire, 58
 ecrire_champ_med, 23
 ecrire_fichier_bin, 58
 ecrire_fichier_formatte, 24
 ecrire_med, 58
 ecriturelecturespecial, 24
 ef, 111, 224
 ef_stab, 112
 end, 30
 entree_temperature_imposee_h, 205
 epsilon, 37
 eqn_base, 128
 execute_parallel, 24
 export, 25
 extract_2d_from_3d, 25
 extract_2daxi_from_3d, 25
 extraire_domaine, 25
 extraire_plan, 26
 extraire_surface, 27
 extrudebord, 27
 extrudeparoi, 28
 extruder, 29
 extruder_en20, 29
 extruder_en3, 30

 fichier_decoupage, 256
 field_uniform_keps_from_ud, 234
 flottabilite, 308
 fluctuation_temperature_w_bas_re, 253
 fluide_diphasique, 250
 fluide_incompressible, 247
 fluide_ostwald, 247
 fluide_quasi_compressible, 248
 flux_radiatif, 205
 flux_radiatif_vdf, 205
 flux_radiatif_vdf, 206
 forchheimer, 303
 frontiere_ouverte, 206
 frontiere_ouverte_concentration_imposee, 206
 frontiere_ouverte_fraction_massique_imposee, 206
 frontiere_ouverte_gradient_pression_impose, 207
 frontiere_ouverte_gradient_pression_impose_vdf, 207
 frontiere_ouverte_gradient_pression_impose_vdfprep1b, 207
 frontiere_ouverte_gradient_pression_libre_vdf, 207
 frontiere_ouverte_gradient_pression_libre_vdfprep1b, 208
 frontiere_ouverte_k_eps_impose, 208
 frontiere_ouverte_pression_imposee, 208
 frontiere_ouverte_pression_imposee_orlansky, 208
 frontiere_ouverte_pression_moyenne_imposee, 209
 frontiere_ouverte_rayo_semi_transp, 209
 frontiere_ouverte_rayo_transp, 209

frontiere_ouverte_rayo_transp_vdf, 209
 frontiere_ouverte_rayo_transp_vef, 210
 frontiere_ouverte_rho_u_impose, 210
 frontiere_ouverte_temperature_imposee, 210
 frontiere_ouverte_temperature_imposee_rayo_semi-
 _transp, 211
 frontiere_ouverte_temperature_imposee_rayo_transp, 211
 frontiere_ouverte_vitesse_imposee, 211
 frontiere_ouverte_vitesse_imposee_sortie, 211

 gaz_parfait, 245
 gaz_reel_rhot, 245
 GCP, 198, 201
 gcp, 201
 gcp_ns, 196
 gen, 197
 generic, 114
 gmres, 197
 Gradient, 198

 IBICGSTAB, 198
 implicite, 295
 imposer_vit_bords_ale, 31
 imprimer_flux, 31
 imprimer_flux_sum, 31
 init_par_partie, 234
 integrer_champ_med, 32
 Interface, 199
 internes, 37
 interpolation_champ_face_deriv, 179
 interpreter, 16

 Jones_Launders, 154

 k_epsilon, 152
 k_epsilon_2_couches, 156
 k_epsilon_bas_reynolds, 154
 k_epsilon_v2, 155
 kquick, 114

 Lam_Bremhorst, 153
 lata_to_med, 32
 lata_to_other, 33
 Launder_Sharma, 153
 leap_frog, 267
 lineaire, 179
 lire_ideas, 33
 lire_tgrid, 46
 list_bloc_mailler, 34
 list_bord, 35
 list_nom, 52
 list_nom_virgule, 184
 liste_post, 71
 liste_post_ok, 69

 listobj, 323
 listobj_impl, 322
 local, 200
 loi_analytique_scalaire, 319
 loi_ciofalo_hydr, 315
 loi_etat_base, 245
 loi_expert_hydr, 315
 loi_expert_scalaire, 319
 loi_horaire, 176, 246
 loi_odvm, 319
 loi_paro_2_couches, 315
 loi_paro_2_couches_scalaire, 320
 loi_paro_nu_impose, 320
 loi_puissance_hydr, 316
 loi_standard_hydr, 316
 loi_standard_hydr_old, 316
 loi_standard_hydr_scalaire, 321
 loi_ww_hydr, 316
 loi_WW_scalaire, 319
 longitudinale, 305
 longueur_melange, 140

 mailler, 33
 mailler_base, 34
 maillerparallel, 38
 masse_ajoutee, 309
 melange_gaz_parfait, 245
 methode_transport_deriv, 176
 metis, 256
 milieu_base, 246
 milieu_v2_base, 250
 mod_turb_hyd_ss_maille, 134
 modele_fonction_bas_reynolds_base, 153
 modele_rayo_semi_transp, 73
 modele_rayonnement_base, 250
 modele_rayonnement_milieu_transparent, 250
 modele_turbulence_hyd_deriv, 132
 modele_turbulence_scal_base, 252
 modif_bord_to_raccord, 39
 mor_eqn, 102
 Moyenne, 66, 68, 69
 moyenne, 67, 190
 moyenne_volumique, 39
 muscl, 114
 muscl3, 112
 muscl_new, 115
 muscl_old, 114

 N, 199
 navier_stokes_ft_disc, 129
 navier_stokes_phase_field, 163
 navier_stokes_qc, 165
 navier_stokes_standard, 167
 navier_stokes_turbulent, 169

navier_stokes_turbulent_qc, 170
 negligeable, 104, 115, 316
 negligeable_scalaire, 321
 nettoiepasnoeuds, 40
 neumann, 211
 nom, 255
 NUL, 133
 NULL, 200
 numero_elem_sur_maitre, 63

 objet_lecture, 323
 optimal, 197
 option, 106
 option_vdf, 40
 orientefacesbord, 41
 orienter_simplexes, 47

 plb, 104
 plncplb, 104
 parametre_diffusion_implicit, 108
 parametre_equation_base, 107
 parametre_implicit, 108
 Paroi, 203
 paroi_adiabatique, 212
 paroi_contact, 212
 paroi_contact_fictif, 213
 paroi_couple, 213
 paroi_decalee_robin, 213
 paroi_defilante, 214
 paroi_echange_contact_correlation_vdf, 214
 paroi_echange_contact_correlation_vef, 215
 paroi_echange_contact_odvm_vdf, 216
 paroi_echange_contact_rayo_semi_transp_vdf, 216
 paroi_echange_contact_vdf, 216
 paroi_echange_contact_vdf_ft, 217
 paroi_echange_contact_vdf_zoom_fin, 217
 paroi_echange_contact_vdf_zoom_grossier, 218
 paroi_echange_externe_impose, 218
 paroi_echange_externe_impose_h, 218
 paroi_echange_externe_impose_rayo_semi_transp, 218
 paroi_echange_externe_impose_rayo_transp, 219
 paroi_echange_global_impose, 219
 paroi_fixe, 219
 paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets, 220
 paroi_flux_impose, 220
 paroi_flux_impose_rayo_semi_transp_vdf, 220
 paroi_flux_impose_rayo_semi_transp_vef, 220
 paroi_flux_impose_rayo_transp, 220
 paroi_ft_disc, 221
 paroi_ft_disc_deriv, 221
 paroi_knudsen_non_negligeable, 221
 paroi_rugueuse, 222
 paroi_tble, 316
 paroi_tble_scal, 321
 paroi_temperature_imposee, 222
 paroi_temperature_imposee_rayo_semi_transp, 222
 paroi_temperature_imposee_rayo_transp, 223
 partition, 41, 257
 partitionneur_deriv, 256
 pave, 34
 pb_avec_passif, 75
 Pb_base, 59
 pb_conduction, 76
 pb_couple_rayo_semi_transp, 77
 pb_couple_rayonnement, 101
 pb_gen_base, 59
 pb_hydraulique, 77
 pb_hydraulique_concentration, 78
 pb_hydraulique_concentration_scalaires_passifs, 79
 pb_hydraulique_concentration_turbulent, 80
 pb_hydraulique_concentration_turbulent_scalaires_passifs, 81
 pb_hydraulique_turbulent, 83
 pb_mg, 83
 pb_phase_field, 84
 pb_thermohydraulique, 86
 pb_thermohydraulique_concentration, 87
 pb_thermohydraulique_concentration_scalaires_passifs, 88
 pb_thermohydraulique_concentration_turbulent, 89
 pb_thermohydraulique_concentration_turbulent_scalaires_passifs, 90
 pb_thermohydraulique_qc, 91
 pb_thermohydraulique_qc_fraction_massique, 92
 pb_thermohydraulique_scalaires_passifs, 93
 pb_thermohydraulique_turbulent, 95
 pb_thermohydraulique_turbulent_qc, 96
 pb_thermohydraulique_turbulent_qc_fraction_massique, 97
 pb_thermohydraulique_turbulent_scalaires_passifs, 98
 pbc_med, 99
 periodique, 223
 perte_charge_anisotrope, 303
 perte_charge_circulaire, 303
 perte_charge_directionnelle, 304
 perte_charge_isotrope, 304
 perte_charge_reguliere, 305
 perte_charge_singuliere, 306
 Petsc, 198, 200
 petsc, 198
 pilote_icoco, 42
 piso, 296
 plan, 63
 point, 62
 points, 62
 porosites, 43
 porosites_champ, 43

position_like, 63
 post_processing, 70
 post_processings, 69
 postraitement_base, 70
 postraitement_ft_lata, 71
 postraiter_domaine, 44
 pp, 125
 prandtl, 253
 precisiongeom, 44
 Precond, 198, 200
 precondition_base, 258
 precondition_local, 258
 precondsolv, 259
 predefini, 190
 Pression, 66, 68, 69
 Print, 200
 problem_read_generic, 100
 probleme_couple, 72
 probleme_ft_disc_gen, 101
 profils_thermo, 162
 puissance_thermique, 307

 quick, 115

 raccord, 37
 raffiner_anisotrope, 45
 raffiner_isotrope, 45
 Raffiner_isotrope_parallele, 16
 read, 45
 read_file, 45
 read_file_binary, 46
 read_med, 47
 read_unsupported_ascii_file_from_icem, 46
 redresser_hexaedres_vdf, 48
 regroupebord, 48
 remove_elem, 48
 remove_invalid_internal_boundaries, 49
 reordonner, 50
 reordonner_faces_periodiques, 49
 reorienter_tetraedres, 50
 reorienter_triangles, 50
 rk3_ft, 269
 rotation, 51
 runge_kutta_ordre_3, 270
 runge_kutta_ordre_4_d3p, 272
 runge_kutta_rationnel_ordre_2, 273

 scatter, 51
 scatterformatte, 51
 scattermed, 51
 Sch_CN_EX_iteratif, 261
 Sch_CN_iteratif, 263
 schema_adams_bashforth_order_2, 275
 schema_adams_bashforth_order_3, 277
 schema_adams_moulton_order_2, 278
 schema_adams_moulton_order_3, 281
 schema_backward_differentiation_order_2, 283
 schema_backward_differentiation_order_3, 285
 schema_implicite_base, 290
 schema_phase_field, 292
 schema_predictor_corrector, 293
 schema_temps_base, 260
 scheme_euler_explicit, 265
 scheme_euler_implicit, 287
 schmidt, 254
 segment, 63
 segmentpoints, 62
 simple, 297
 simplifier, 298
 solide, 250
 solve, 52
 Solver, 198, 201
 Solveur, 198, 200
 solveur_implicite_base, 295
 solveur_lineaire_std, 299
 solveur_sys_base, 202
 Solveur_pression, 198, 200
 sonde_base, 61
 sortie_libre_rho_variable, 223
 sortie_libre_temperature_imposee_h, 223
 source_base, 299
 source_con_phase_field, 307
 source_constituant, 308
 source_generique, 308
 source_qdm, 309
 source_qdm_lambdaup, 309
 source_qdm_phase_field, 309
 source_rayo_semi_transp, 310
 source_robin, 310
 source_robin_scalaire, 310
 source_th_tdivu, 311
 source_transport_k_eps, 311
 source_transport_k_eps_aniso_concen, 311
 source_transport_k_eps_aniso_therm_concen, 312
 Source_Transport_K_Eps_anisotherme, 299
 source_transport_k_eps_bas_reynolds, 312
 sources, 106
 sous_maille, 141
 sous_maille_1elt, 145
 sous_maille_1elt_selectif_mod, 147
 sous_maille_axi, 148
 sous_maille_dyn, 254
 sous_maille_selectif, 144
 sous_maille_selectif_mod, 142
 sous_maille_smago, 137
 sous_maille_smago_dyn, 150
 sous_maille_smago_filtre, 149
 sous_maille_wale, 136

- sous_zone, [313](#)
- sous_zones, [257](#)
- Spai, [200](#)
- spec_pdc_r_base, [305](#)
- SSOR, [200](#), [201](#)
- ssor, [259](#)
- ssor_bloc, [259](#)
- stab, [104](#)
- standard, [105](#)
- standard_KEps, [153](#)
- stat_post_deriv, [67](#)
- Statistiques, [66](#), [68](#), [69](#)
- Statistiques_en_serie, [68](#), [69](#)
- supg, [115](#)
- supprime_bord, [52](#)
- symetrie, [221](#), [224](#)
- system, [52](#)

- t_deb, [67](#)
- t_fin, [67](#)
- tayl_green, [234](#)
- Temperature, [66](#), [68](#)
- temperature, [158](#)
- temperature_imposee_paro_i, [224](#)
- test_solveur, [53](#)
- testeur, [53](#)
- testeur_medcoupling, [54](#)
- tetraedriser, [54](#)
- tetraedriser_homogene, [54](#)
- tetraedriser_homogene_compact, [54](#)
- tetraedriser_homogene_fin, [55](#)
- tetraedriser_par_prisme, [55](#)
- thi, [159](#)
- thi_thermo, [160](#)
- trainee, [311](#)
- traitement_particulier_base, [158](#)
- tranche, [258](#)
- transformer, [55](#)
- transport_interfaces_ft_disc, [172](#)
- transport_k_epsilon, [179](#)
- transport_marqueur_ft, [180](#)
- transversale, [306](#)
- triangler, [56](#)
- triangler_fin, [56](#)
- triangler_h, [56](#)
- turbulence_paro_i_base, [315](#)
- turbulence_paro_i_scalaire_base, [319](#)
- type, [66](#), [68](#), [199](#), [200](#)

- uniform_field, [235](#)
- utau_imp, [318](#)

- valeur_totale_sur_volume, [235](#)
- vdf, [224](#)

- vect_nom, [57](#)
- vef, [225](#)
- vefprep1b, [225](#)
- verifier_qualite_raffinements, [57](#)
- verifier_simplexes, [57](#)
- verifiercoin, [57](#)
- Vitesse, [66](#), [68](#)
- vitesse_imposee, [176](#)
- vitesse_interpolee, [176](#)
- volume, [64](#)

- xyz, [15](#)