

TRUST Reference Manual V1.9.7

Support team: trust@cea.fr

November 27, 2025

Contents

1	Syntax to define a mathematical function	16
2	Existing & predefined fields names	17
3	interprete	19
3.1	Create_domain_from_sub_domain	20
3.2	Write_med	20
3.3	Link_cgns_files	21
3.4	Merge_med	21
3.5	Multiplefiles	21
3.6	My_comm_group	22
3.7	Op_conv_ef_stab_polymac_face	22
3.8	Op_conv_ef_stab_polymac_p0p1nc_elem	22
3.9	Op_conv_ef_stab_polymac_p0p1nc_face	22
3.10	Op_conv_ef_stab_polymac_p0_face	23
3.11	Option_cgns	23
3.12	Option_dg	23
3.13	Option_ijk	24
3.14	Option_interpolation	24
3.15	Option_polymac	25
3.16	Parallel_io_parameters	25
3.17	Raffiner_isotrope_parallele	26
3.18	Read_med	26
3.19	Test_sse_kernels	27
3.20	Analyse_angle	27
3.21	Associate	27
3.22	Axi	28
3.23	Bidim_axi	28
3.24	Calculer_moments	28
3.25	Lecture_bloc_moment_base	28
3.25.1	Calcul	29
3.25.2	Centre_de_gravite	29
3.25.3	Un_point	29
3.26	Corriger_frontiere_periodique	29
3.27	Criteres_convergence	30
3.28	Debog	30
3.29	{	30
3.30	Decoupebord_pour_rayonnement	31
3.31	Decouper_bord_coincident	31
3.32	Dilate	32
3.33	Dimension	32
3.34	Disable_tu	32
3.35	Discretiser_domaine	32
3.36	Discretize	33
3.37	Distance_parois	33
3.38	Ecrire_champ_med	33
3.39	Ecrire_fichier_formatte	34
3.40	Ecrire_fichier_xyz_valeur	34
3.41	Ecriturelecturespecial	34
3.42	Espece	35
3.43	Execute_parallel	35
3.44	Export	35

3.45	Extract_2d_from_3d	36
3.46	Extract_2daxi_from_3d	36
3.47	Extraire_domaine	36
3.48	Extraire_plan	37
3.49	Extraire_surface	37
3.50	Extrudebord	38
3.51	Extrudeparoi	39
3.52	Extruder	39
3.53	Troisf	40
3.54	Extruder_en20	40
3.55	Extruder_en3	40
3.56	Facsec_expert	41
3.57	End	42
3.58	}	42
3.59	Imprimer_flux	42
3.60	Bloc_lecture	42
3.60.1	Bloc_criteres_convergence	42
3.60.2	Solveur_petsc_option_cli	43
3.61	Imprimer_flux_sum	43
3.62	Integrer_champ_med	43
3.63	Interprete_geometrique_base	44
3.64	Lata_to_cgns	44
3.65	Format_lata_to_cgns	44
3.66	Lata_2_med	45
3.67	Format_lata_to_med	45
3.68	Lata_2_other	45
3.69	Lire_ideas	46
3.70	Lml_2_lata	46
3.71	Mailler	46
3.72	List_bloc_mailler	46
3.72.1	Mailler_base	47
3.72.2	Pave	47
3.72.3	Bloc_pave	47
3.72.4	List_bord	48
3.72.5	Bord_base	48
3.72.6	Raccord	49
3.72.7	Defbord	49
3.72.8	Defbord_2	49
3.72.9	Defbord_3	49
3.72.10	Internes	50
3.72.11	Bord	50
3.72.12	Epsilon	50
3.72.13	Domain	51
3.73	Maillerparallel	51
3.74	Mass_source	52
3.75	Mkdir	52
3.76	Modif_bord_to_raccord	53
3.77	Modifydomaineaxi1d	53
3.78	Moyenne_volumique	53
3.79	Multigrid_solver	54
3.80	Coarsen_operators	55
3.80.1	Coarsen_operator_uniform	55
3.81	Nettoiepasnoeuds	56
3.82	Option_vdf	56

3.83	Orientefacesbord	57
3.84	Partition	57
3.85	Bloc_decouper	57
3.86	Partition_multi	58
3.87	Pilote_icoco	59
3.88	Polyedriser	59
3.89	Postraiter_domaine	59
3.90	Precisiongeom	60
3.91	Raffiner_anisotrope	60
3.92	Raffiner_isotrope	61
3.93	Read	62
3.94	Read_file	62
3.95	Read_file_binary	63
3.96	Lire_tgrid	63
3.97	Read_unsupported_ascii_file_from_icem	63
3.98	Orienter_simplexes	64
3.99	Redresser_hexaedres_vdf	64
3.100	Refine_mesh	64
3.101	Regroupebord	64
3.102	Remove_elem	65
3.103	Remove_elem_bloc	65
3.104	Remove_invalid_internal_boundaries	66
3.105	Reorienter_tetraedres	66
3.106	Reorienter_triangles	66
3.107	Reordonner	66
3.108	Residuals	67
3.109	Rotation	67
3.110	Scatter	67
3.111	Scatteredmed	68
3.112	Solve	68
3.113	Stat_per_proc_perf_log	68
3.114	Supprime_bord	68
3.115	List_nom	69
3.116	System	69
3.117	Test_solveur	69
3.118	Testeur	70
3.119	Testeur_medcoupling	70
3.120	Tetraedriser	70
3.121	Tetraedriser_homogene	71
3.122	Tetraedriser_homogene_compact	71
3.123	Tetraedriser_homogene_fin	72
3.124	Tetraedriser_par_prisme	73
3.125	Transformer	73
3.126	Trianguler	74
3.127	Trianguler_fin	74
3.128	Trianguler_h	75
3.129	Verifier_qualite_raffinements	75
3.130	Vect_nom	75
3.131	Verifier_simplexes	76
3.132	Verifiercoin	76
3.133	Verifiercoin_bloc	76
3.134	Ecrire	76
3.135	Ecrire_fichier_bin	77

4	pb_gen_base	77
4.1	Pb_conduction	77
4.2	Corps_postraitement	78
4.2.1	Interface_posts	80
4.2.2	Champs_a_post	80
4.2.3	Champ_a_post	80
4.2.4	Definition_champs	81
4.2.5	Definition_champ	81
4.2.6	Definition_champs_fichier	81
4.2.7	Sondes	81
4.2.8	Sonde	81
4.2.9	Sonde_base	82
4.2.10	Points	82
4.2.11	Listpoints	82
4.2.12	Point	83
4.2.13	Segmentpoints	83
4.2.14	Segment	83
4.2.15	Segmentfacesx	83
4.2.16	Segmentfacesy	84
4.2.17	Segmentfacesz	84
4.2.18	Radius	84
4.2.19	Numero_elem_sur_maitre	85
4.2.20	Position_like	85
4.2.21	Plan	85
4.2.22	Volume	85
4.2.23	Circle	86
4.2.24	Circle_3	86
4.2.25	Sondes_fichier	86
4.2.26	Champs_posts	87
4.2.27	Champs_posts_fichier	87
4.2.28	Bloc_fichier	87
4.2.29	Stats_posts	87
4.2.30	List_stat_post	88
4.2.31	Stat_post_deriv	88
4.2.32	T_deb	88
4.2.33	T_fin	89
4.2.34	Moyenne	89
4.2.35	Ecart_type	89
4.2.36	Correlation	90
4.2.37	Stats_posts_fichier	90
4.2.38	Stats_serie_posts	91
4.2.39	Stats_serie_posts_fichier	91
4.3	Post_processings	92
4.3.1	Un_postraitement	92
4.4	Liste_post_ok	92
4.4.1	Nom_postraitement	92
4.4.2	Postraitement_base	92
4.4.3	Post_processing	93
4.5	Liste_post	94
4.5.1	Un_postraitement_spec	94
4.5.2	Type_un_post	95
4.5.3	Type_postraitement_ft_lata	95
4.6	Format_file_base	95
4.6.1	Binaire	95

4.6.2	Formatte	96
4.6.3	Xyz	96
4.6.4	Single_hdf	96
4.6.5	Pdi	96
4.6.6	Pdi_expert	97
4.7	Pb_conduction_ibm	97
4.8	Pb_hydraulique_cloned_concentration	98
4.9	Pb_hydraulique_cloned_concentration_turbulent	99
4.10	Pb_hydraulique_ibm_turbulent	100
4.11	Pb_hydraulique_list_concentration	101
4.12	Listeqn	103
4.13	Pb_hydraulique_list_concentration_turbulent	103
4.14	Pb_multiphase	104
4.15	Pb_multiphase_h	106
4.16	Pb_hem	107
4.17	Pb_thermohydraulique_cloned_concentration	109
4.18	Pb_thermohydraulique_cloned_concentration_turbulent	110
4.19	Pb_thermohydraulique_ibm_turbulent	111
4.20	Pb_thermohydraulique_list_concentration	113
4.21	Pb_thermohydraulique_list_concentration_turbulent	114
4.22	Pb_base	115
4.23	Probleme_couple	116
4.24	List_list_nom	117
4.25	Pb_avec_liste_conc	117
4.26	Pb_avec_passif	118
4.27	Pb_hydraulique	119
4.28	Pb_hydraulique_concentration	121
4.29	Pb_hydraulique_concentration_scalaires_passifs	122
4.30	Pb_hydraulique_concentration_turbulent	123
4.31	Pb_hydraulique_concentration_turbulent_scalaires_passifs	124
4.32	Pb_hydraulique_ibm	126
4.33	Pb_hydraulique_melange_binaire_qc	127
4.34	Pb_hydraulique_melange_binaire_wc	128
4.35	Pb_hydraulique_melange_binaire_turbulent_qc	129
4.36	Pb_hydraulique_turbulent	130
4.37	Pb_post	131
4.38	Pb_thermohydraulique	132
4.39	Pb_thermohydraulique_qc	134
4.40	Pb_thermohydraulique_wc	135
4.41	Pb_thermohydraulique_concentration	136
4.42	Pb_thermohydraulique_concentration_scalaires_passifs	138
4.43	Pb_thermohydraulique_concentration_turbulent	139
4.44	Pb_thermohydraulique_concentration_turbulent_scalaires_passifs	140
4.45	Pb_thermohydraulique_especes_qc	142
4.46	Pb_thermohydraulique_especes_wc	143
4.47	Pb_thermohydraulique_especes_turbulent_qc	144
4.48	Pb_thermohydraulique_ibm	146
4.49	Pb_thermohydraulique_scalaires_passifs	147
4.50	Pb_thermohydraulique_turbulent	148
4.51	Pb_thermohydraulique_turbulent_qc	149
4.52	Pb_thermohydraulique_turbulent_scalaires_passifs	150
4.53	Pbc_med	152
4.54	List_info_med	152
4.54.1	Info_med	152

4.55	Problem_read_generic	152
5	mor_eqn	153
5.1	Conduction	154
5.2	Bloc_convection	154
5.2.1	Convection_deriv	155
5.2.2	Ale	155
5.2.3	Muscl_old	155
5.2.4	Muscl3	155
5.2.5	Ef	156
5.2.6	Bloc_ef	156
5.2.7	Di_l2	157
5.2.8	Amont_old	157
5.2.9	Generic	157
5.2.10	Ef_stab	157
5.2.11	Listsous_zone_valeur	158
5.2.12	Sous_zone_valeur	158
5.2.13	Kquick	158
5.2.14	Muscl	159
5.2.15	Muscl_new	159
5.2.16	Quick	159
5.2.17	Centre_old	159
5.2.18	Negligeable	159
5.2.19	Amont	160
5.2.20	Centre	160
5.2.21	Centre4	160
5.2.22	Btd	160
5.2.23	Supg	160
5.3	Bloc_diffusion	161
5.3.1	Diffusion_deriv	161
5.3.2	Turbulente	161
5.3.3	Type_diffusion_turbulente_multiphase_deriv	161
5.3.4	Interfacial_area	162
5.3.5	Wale	162
5.3.6	L_melange	162
5.3.7	Smago	163
5.3.8	Prandtl	163
5.3.9	Sgdh	163
5.3.10	Stab	164
5.3.11	Standard	164
5.3.12	Bloc_diffusion_standard	165
5.3.13	P1ncplb	165
5.3.14	P1b	165
5.3.15	Negligeable	165
5.3.16	Option	166
5.3.17	Op_implicite	166
5.4	Condlims	166
5.4.1	Condlimlu	166
5.5	Condinit	167
5.5.1	Condinit	167
5.6	Sources	167
5.7	Parametre_equation_base	167
5.7.1	Parametre_implicite	167
5.7.2	Parametre_diffusion_implicite	168

5.8	Conduction_ibm	168
5.9	Convection_diffusion_espece_binaire_turbulent_qc	169
5.10	Echelle_temporelle_turbulente	170
5.11	Energie_multiphase	171
5.12	Energie_multiphase_h	172
5.13	Energie_cinetique_turbulente	173
5.14	Energie_cinetique_turbulente_wit	174
5.15	Masse_multiphase	174
5.16	Qdm_multiphase	175
5.17	Taux_dissipation_turbulent	176
5.18	Convection_diffusion_chaleur_qc	177
5.19	Convection_diffusion_chaleur_wc	178
5.20	Convection_diffusion_chaleur_turbulent_qc	179
5.21	Convection_diffusion_concentration	180
5.22	Convection_diffusion_concentration_turbulent	181
5.23	Convection_diffusion_espece_binaire_qc	182
5.24	Convection_diffusion_espece_binaire_wc	183
5.25	Convection_diffusion_espece_multi_qc	184
5.26	Convection_diffusion_espece_multi_wc	185
5.27	Convection_diffusion_espece_multi_turbulent_qc	185
5.28	Convection_diffusion_temperature	186
5.29	Convection_diffusion_temperature_ibm	187
5.30	Convection_diffusion_temperature_ibm_turbulent	188
5.31	Convection_diffusion_temperature_turbulent	189
5.32	Eqn_base	190
5.33	Navier_stokes_qc	191
5.34	Deuxmots	193
5.35	Traitement_particulier	193
5.35.1	Traitement_particulier_base	193
5.35.2	Profils_thermo	193
5.35.3	Temperature	193
5.35.4	Canal	194
5.35.5	Chmoy_faceperio	194
5.35.6	Ec	195
5.35.7	Thi	195
5.36	Floatfloat	196
5.37	Navier_stokes_wc	196
5.38	Navier_stokes_ibm	198
5.39	Navier_stokes_ibm_turbulent	200
5.40	Modele_turbulence_hyd_deriv	202
5.40.1	Dt_impr_ustar_mean_only	202
5.40.2	Mod_turb_hyd_ss_maille	203
5.40.3	Form_a_nb_points	204
5.40.4	Sous_maille_smago	204
5.40.5	Sous_maille_wale	205
5.40.6	Longueur_melange	206
5.40.7	Null	208
5.41	Navier_stokes_standard	208
5.42	Navier_stokes_turbulent	210
5.43	Navier_stokes_turbulent_qc	212
6	domaine_base	213
6.1	Domaine_ijk	214
6.2	Troismots	214

7	interface_base	214
7.1	Interface_sigma_constant	215
7.2	Saturation_base	215
7.3	Saturation_constant	215
7.4	Saturation_sodium	216
8	/*	216
8.1	/*	216
9	champ_generique_base	217
9.1	Champ_post_de_champs_post	217
9.2	Listchamp_generique	217
9.3	List_nom_virgule	218
9.4	Champ_post_operateur_base	218
9.5	Champ_post_operateur_eqn	218
9.6	Champ_post_statistiques_base	219
9.7	Correlation	220
9.8	Champ_post_operateur_divergence	220
9.9	Ecart_type	221
9.10	Champ_post_extraction	221
9.11	Champ_post_operateur_gradient	222
9.12	Interpolation	222
9.13	Champ_post_morceau_equation	223
9.14	Moyenne	224
9.15	Predefini	225
9.16	Champ_post_reduction_0d	225
9.17	Champ_post_refchamp	226
9.18	Champ_post_tparoi_vef	226
9.19	Champ_post_transformation	227
10	chimie	228
10.1	Reactions	228
10.1.1	Reaction	228
11	class_generic	229
11.1	Amg	229
11.2	Amgx	230
11.3	Cholesky	230
11.4	Cudss	230
11.5	Dt_calc	230
11.6	Dt_fixe	231
11.7	Dt_min	231
11.8	Dt_start	231
11.9	Gcp_ns	231
11.10	Gen	232
11.11	Gmres	233
11.12	Optimal	233
11.13	Petsc	234
11.14	Petsc_gpu	234
11.15	Gcp	234
11.16	Solveur_sys_base	235
12	#	236
12.1	#	236

13 condlim_base	236
13.1 Echange_couplage_thermique	236
13.2 Paroi_echange_interne_global_impose	236
13.3 Paroi_echange_interne_global_parfait	237
13.4 Paroi_echange_interne_impose	237
13.5 Paroi_echange_interne_parfait	237
13.6 Neumann_homogene	237
13.7 Neumann_paro	238
13.8 Neumann_paro_adiabatique	238
13.9 Paroi	238
13.10Dirichlet	238
13.11Paroi_echange_externeradiatif	238
13.12Entree_temperature_imposee_h	239
13.13Frontiere_ouverte	239
13.14Frontiere_ouverte_alpha_impose	239
13.15Frontiere_ouverte_concentration_imposee	240
13.16Frontiere_ouverte_fraction_massique_imposee	240
13.17Frontiere_ouverte_gradient_pression_impose	240
13.18Frontiere_ouverte_gradient_pression_impose_vefprep1b	240
13.19Frontiere_ouverte_gradient_pression_libre_vef	241
13.20Frontiere_ouverte_gradient_pression_libre_vefprep1b	241
13.21Frontiere_ouverte_pression_imposee	241
13.22Frontiere_ouverte_pression_imposee_orlansky	241
13.23Frontiere_ouverte_pression_moyenne_imposee	241
13.24Frontiere_ouverte_rho_u_impose	242
13.25Frontiere_ouverte_enthalpie_imposee	242
13.26Frontiere_ouverte_vitesse_imposee	242
13.27Frontiere_ouverte_vitesse_imposee_sortie	243
13.28Neumann	243
13.29Paroi_adiabatique	243
13.30Paroi_contact	243
13.31Paroi_contact_fictif	244
13.32Paroi_decalee_robin	244
13.33Paroi_defilante	245
13.34Paroi_echange_contact_correlation_vdf	245
13.35Paroi_echange_contact_correlation_vef	246
13.36Paroi_echange_contact_vdf	247
13.37Paroi_echange_externer_impose	247
13.38Paroi_echange_externer_impose_h	248
13.39Paroi_echange_global_impose	248
13.40Paroi_fixe	248
13.41Paroi_fixe_iso_genepi2_sans_contribution_aux_vitesses_sommets	248
13.42Paroi_flux_impose	249
13.43Paroi_knudsen_non_negligeable	249
13.44Paroi_temperature_imposee	249
13.45Periodique	250
13.46Robin_vef	250
13.47Scalaire_impose_paro	250
13.48Sortie_libre_temperature_imposee_h	250
13.49Symetrie	251
13.50Enthalpie_imposee_paro	251

14 discretisation_base	251
14.1 Dg	251
14.2 Ef_axi	251
14.3 Ef	252
14.4 Ijk	252
14.5 Polymac	252
14.6 Polymac_p0p1nc	252
14.7 Polymac_p0	252
14.8 Vdf	252
14.9 Vef	252
15 domaine	253
15.1 Domaineaxild	253
15.2 Ijk_grid_geometry	254
16 champ_base	254
16.1 Champ_base	254
16.2 Champ_fonc_interp	254
16.3 Champ_fonc_med_table_temps	255
16.4 Champ_fonc_med_tabule	256
16.5 Champ_tabule_morceaux	257
16.6 Champ_fonc_tabule_morceaux_interp	257
16.7 Champ_parametrique	257
16.8 Champ_composite	258
16.9 Champ_don_base	258
16.10 Champ_don_lu	258
16.11 Champ_fonc_fonction	259
16.12 Champ_fonc_fonction_txyz	259
16.13 Champ_fonc_fonction_txyz_morceaux	259
16.14 Champ_fonc_med	260
16.15 Champ_fonc_reprise	260
16.16 Fonction_champ_reprise	261
16.17 Champ_fonc_t	261
16.18 Champ_fonc_tabule	261
16.19 Champ_init_canal_sinal	262
16.20 Bloc_lec_champ_init_canal_sinal	262
16.21 Champ_input_base	263
16.22 Champ_input_p0	263
16.23 Champ_input_p0_composite	264
16.24 Champ_musig	264
16.25 Champ_ostwald	265
16.26 Champ_som_lu_vdf	265
16.27 Champ_som_lu_vef	265
16.28 Champ_tabule_lu	266
16.29 Champ_tabule_temps	266
16.30 Champ_uniforme_morceaux	266
16.31 Champ_uniforme_morceaux_tabule_temps	266
16.32 Champ_fonc_txyz	267
16.33 Champ_fonc_xyz	267
16.34 Init_par_partie	267
16.35 Tayl_green	268
16.36 Uniform_field	268
16.37 Valeur_totale_sur_volume	268

17	champ_front_base	268
17.1	Champ_front_base	268
17.2	Champ_front_xyz_tabule	269
17.3	Champ_front_parametrique	269
17.4	Champ_front_debit_qc_vdf	269
17.5	Champ_front_debit_qc_vdf_fonc_t	270
17.6	Boundary_field_inward	270
17.7	Ch_front_input	270
17.8	Ch_front_input_uniforme	271
17.9	Champ_front_med	271
17.10	Champ_front_bruite	272
17.11	Champ_front_calc	272
17.12	Champ_front_composite	272
17.13	Champ_front_contact_vef	273
17.14	Champ_front_debit	273
17.15	Champ_front_debit_massique	273
17.16	Champ_front_fonc_pois_ipsn	273
17.17	Champ_front_fonc_pois_tube	274
17.18	Champ_front_fonc_t	274
17.19	Champ_front_fonc_txyz	274
17.20	Champ_front_fonc_xyz	274
17.21	Champ_front_fonction	275
17.22	Champ_front_lu	275
17.23	Champ_front_musig	275
17.24	Champ_front_normal_vef	275
17.25	Champ_front_pression_from_u	276
17.26	Champ_front_recyclage	276
17.27	Champ_front_tabule	277
17.28	Champ_front_tabule_lu	277
17.29	Champ_front_tangentiel_vef	277
17.30	Champ_front_uniforme	278
17.31	Champ_front_xyz_debit	278
18	interpolation_ibm_base	278
18.1	Interpolation_ibm_power_law_tbl_u_star	278
18.2	Ibm_aucune	279
18.3	Ibm_element_fluide	279
18.4	Ibm_hybride	280
18.5	Ibm_gradient_moyen	281
18.6	Ibm_power_law_tbl	281
19	loi_etat_base	282
19.1	Eos_qc	282
19.2	Eos_wc	282
19.3	Binaire_gaz_parfait_qc	283
19.4	Binaire_gaz_parfait_wc	283
19.5	Coolprop_qc	284
19.6	Coolprop_wc	284
19.7	Loi_etat_gaz_parfait_base	285
19.8	Loi_etat_gaz_reel_base	285
19.9	Loi_etat_tppi_base	285
19.10	Multi_gaz_parfait_qc	285
19.11	Multi_gaz_parfait_wc	286
19.12	Gaz_parfait_qc	286

19.13	Gaz_parfait_wc	287
19.14	Rhot_gaz_parfait_qc	287
19.15	Rhot_gaz_reel_qc	288
20	loi_fermeture_base	288
20.1	Loi_fermeture_test	288
21	loi_horaire	288
22	milieu_base	289
22.1	Constituant	289
22.2	Fluide_base	290
22.3	Fluide_dilatable_base	291
22.4	Fluide_incompressible	291
22.5	Fluide_ostwald	292
22.6	Fluide_quasi_compressible	293
22.7	Bloc_sutherland	294
22.8	Fluide_reel_base	295
22.9	Fluide_sodium_gaz	296
22.10	Fluide_sodium_liquide	296
22.11	Fluide_stiffened_gas	297
22.12	Fluide_weakly_compressible	298
22.13	Solide	299
23	modele_turbulence_scal_base	300
23.1	Dt_impr_nusselt_mean_only	301
23.2	Null	301
23.3	Prandtl	302
23.4	Schmidt	302
24	moyenne_imposee_deriv	303
24.1	Connexion_approchee	303
24.2	Connexion_exacte	304
24.3	Interpolation	304
24.4	Logarithmique	305
24.5	Profil	305
25	nom	305
25.1	Nom_anonyme	306
26	partitionneur_deriv	306
26.1	Fichier_med	306
26.2	Fichier_decoupage	307
26.3	Metis	307
26.4	Partition	308
26.5	Sous_dom	308
26.6	Sous_zones	309
26.7	Tranche	309
26.8	Union	310
27	pb_champ_evaluateur	310
28	porosites	310
28.1	Bloc_lecture_poro	311

29	precond_base	311
29.1	Ilu	311
29.2	Precondsolv	312
29.3	Ssor	312
29.4	Ssor_bloc	312
30	preconditionneur_petsc_deriv	313
30.1	Block_jacobi_icc	313
30.2	Eisentat	313
30.3	Block_jacobi_ilu	314
30.4	Boomeramg	314
30.5	C-amg	314
30.6	Diag	314
30.7	Jacobi	314
30.8	Lu	314
30.9	Null	315
30.10	Pilut	315
30.11	Sa-amg	315
30.12	Spai	315
30.13	Ssor	316
31	schema_temps_base	316
31.1	Sch_cn_ex_iteratif	318
31.2	Sch_cn_iteratif	320
31.3	Scheme_euler_explicit	323
31.4	Leap_frog	325
31.5	Runge_kutta_ordre_2	327
31.6	Runge_kutta_ordre_2_classique	329
31.7	Runge_kutta_ordre_3	331
31.8	Runge_kutta_ordre_3_classique	333
31.9	Runge_kutta_ordre_4_d3p	335
31.10	Runge_kutta_ordre_4_classique	337
31.11	Runge_kutta_ordre_4_classique_3_8	339
31.12	Runge_kutta_rationnel_ordre_2	341
31.13	Schema_adams_bashforth_order_2	343
31.14	Schema_adams_bashforth_order_3	345
31.15	Schema_adams_moulton_order_2	347
31.16	Schema_adams_moulton_order_3	349
31.17	Schema_backward_differentiation_order_2	352
31.18	Schema_backward_differentiation_order_3	354
31.19	Scheme_euler_implicit	357
31.20	Schema_implicite_base	360
31.21	Schema_predictor_corrector	362
32	solveur_implicite_base	364
32.1	Ice	364
32.2	Implicite	365
32.3	Piso	366
32.4	Sets	367
32.5	Simple	368
32.6	Simpler	369
32.7	Solveur_lineaire_std	370
32.8	Solveur_u_p	370

33	solveur_petsc_deriv	371
33.1	Bicgstab	372
33.2	Cholesky_out_of_core	372
33.3	Cholesky_pastix	373
33.4	Cholesky_superlu	373
33.5	Cholesky_umfpack	374
33.6	Ibicgstab	374
33.7	Pipecg	375
33.8	Cholesky	376
33.9	Cholesky_mumps_blr	377
33.10	Cli	378
33.11	Cli_quiet	378
33.12	Gcp	379
33.13	Gmres	380
33.14	Lu	380
34	source_base	381
34.1	Correction_antal	382
34.2	Correction_tomiyama	382
34.3	Dp_impose	382
34.4	Type_perte_charge_deriv	382
34.4.1	Dp	382
34.4.2	Dp_regul	383
34.5	Dispersion_bulles	383
34.6	Portance_interfaciale	383
34.7	Source_dep_inco_bases	384
34.8	Acceleration	384
34.9	Boussinesq_concentration	385
34.10	Boussinesq_temperature	385
34.11	Canal_perio	385
34.12	Coriolis	386
34.13	Darcy	386
34.14	Dirac	387
34.15	Flux_interfacial	387
34.16	Forchheimer	387
34.17	Frottement_interfacial	387
34.18	Perte_charge_anisotrope	388
34.19	Perte_charge_circulaire	388
34.20	Perte_charge_directionnelle	389
34.21	Perte_charge_isotrope	389
34.22	Perte_charge_reguliere	390
34.23	Spec_pdc_base	390
34.23.1	Longitudinale	390
34.23.2	Transversale	390
34.24	Perte_charge_singuliere	391
34.25	Puissance_thermique	391
34.26	Radioactive_decay	392
34.27	Source_constituant	392
34.28	Source_generique	392
34.29	Source_pdf	392
34.30	Bloc_pdf_model	393
34.31	Source_pdf_base	393
34.32	Source_qdm	394
34.33	Source_qdm_lambdaup	394

34.34	Source_th_tdivu	395
34.35	Terme_puissance_thermique_echange_impose	395
34.36	Travail_pression	395
34.37	Vitesse_derive_base	396
34.38	Vitesse_relative_base	396
35	sous_zone	396
35.1	Bloc_origine_cotes	397
35.2	Deuxentiers	397
35.3	Bloc_couronne	397
35.4	Bloc_tube	398
36	turbulence_paro_base	398
36.1	Negligeable	398
37	turbulence_paro_scalaire_base	398
37.1	Negligeable_scalaire	399
38	listobj_impl	399
38.1	Milieu_musig	399
38.2	Milieu_composite	399
38.3	List_un_pb	399
38.4	Un_pb	399
38.5	Listobj	400
39	objet_lecture	400
39.1	Quatremots	400
39.2	Entierfloat	401
40	index	401

1 Syntax to define a mathematical function

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

ABS : absolute value function
 COS : cosine function
 SIN : sine function
 TAN : tangent function
 ATAN : arctangent function
 EXP : exponential function
 LN : natural logarithm function
 SQRT : square root function
 INT : integer function
 ERF : error function
 RND(x) : random function (values between 0 and x)
 COSH : hyperbolic cosine function
 SINH : hyperbolic sine function
 TANH : hyperbolic tangent function
 ACOS : inverse cosine function
 ASIN : inverse sine function
 ATANH : inverse hyperbolic tangent function
 NOT(x) : NOT x (returns 1 if x is false, 0 otherwise)
 SGN(x) : SGN x (returns 1 if x is positive, -1 if negative, 0 if zero)

`x_AND_y` : boolean logical operation AND (returns 1 if both x and y are true, else 0)
`x_OR_y` : boolean logical operation OR (returns 1 if x or y is true, else 0)
`x_GT_y` : greater than (returns 1 if $x > y$, else 0)
`x_GE_y` : greater than or equal to (returns 1 if $x \geq y$, else 0)
`x_LT_y` : less than (returns 1 if $x < y$, else 0)
`x_LE_y` : less than or equal to (returns 1 if $x \leq y$, else 0)
`x_MIN_y` : returns the smallest of x and y
`x_MAX_y` : returns the largest of x and y
`x_MOD_y` : modular division of x per y
`x_EQ_y` : equal to (returns 1 if $x == y$, else 0)
`x_NEQ_y` : not equal to (returns 1 if $x != y$, else 0)

You can also use the following operations:

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

You can also use the following constants:

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

The variables which can be used are:

`x,y,z` : coordinates
`t` : time

Examples:

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

Possible errors:

Error 1:

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

Error 2:

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

2 Existing & predefined fields names

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

Physical values	Keyword for field_name	Unit
Velocity	Vitesse or Velocity	$m.s^{-1}$
Velocity residual	Vitesse_residu	$m.s^{-2}$
Kinetic energy per elements ($0.5\rho u_i ^2$)	Energie_cinetique_elem	$kg.m^{-1}.s^{-2}$
Total kinetic energy $\left(\frac{\sum_{i=1}^{nb_elem} 0.5\rho u_i ^2 vol_i}{\sum_{i=1}^{nb_elem} vol_i}\right)$	Energie_cinetique_totale	$kg.m^{-1}.s^{-2}$
Vorticity	Vorticite	s^{-1}
Pressure in incompressible flow ($P/\rho + gz$) For Front Tracking probleme ($P + \rho gz$)	Pression ¹	$Pa.m^3.kg^{-1}$ or Pa
Pressure in incompressible flow ($P+\rho gz$)	Pression_pa or Pressure	Pa
Pressure in compressible flow	Pression	Pa
Hydrostatic pressure (ρgz)	Pression_hydrostatique	Pa
Totale pressure (when quasi compressible model is used)=Pth+P	Pression_tot	Pa
Pressure gradient ($\nabla(P/\rho + gz)$)	Gradient_pression	$m.s^{-2}$
Velocity gradient	gradient_vitesse	s^{-1}
Temperature	Temperature	$^{\circ}C$ or K
Temperature residual	Temperature_residu	$^{\circ}C.s^{-1}$ or $K.s^{-1}$
Phase temperature of a two phases flow	Temperature_EquationName	$^{\circ}C$ or K
Mass transfer rate between two phases	Temperature_mpoint	$kg.m^{-2}.s^{-1}$
Temperature variance	Variance_Temperature	K^2
Temperature dissipation rate	Taux_Dissipation_Temperature	$K^2.s^{-1}$
Temperature gradient	Gradient_temperature	$K.m^{-1}$
Heat exchange coefficient	H_echange_Tref ²	$W.m^{-2}.K^{-1}$
Turbulent heat flux	Flux_Chaleur_Turbulente	$m.K.s^{-1}$
Turbulent viscosity	Viscosite_turbulente	$m^2.s^{-1}$
Turbulent dynamic viscosity (when quasi compressible model is used)	Viscosite_dynamique_turbulente	$kg.m.s^{-1}$
Turbulent kinetic energy	K	$m^2.s^{-2}$
Turbulent dissipation rate	Eps	$m^3.s^{-1}$
Turbulent quantities K and Epsilon	K_Eps	$(m^2.s^{-2}, m^3.s^{-1})$
Residuals of turbulent quantities K and Epsilon residuals	K_Eps_residu	$(m^2.s^{-3}, m^3.s^{-2})$
Constituent concentration	Concentration	
Constituent concentration residual	Concentration_residu	
Component velocity along X	VitesseX	$m.s^{-1}$
... continued on next page ...		

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

²Tref indicates the value of a reference temperature and must be specified by the user. For example, H_echange_293 is the keyword to use for Tref=293K.

Physical values	Keyword for field_name	Unit
Component velocity along Y	VitesseY	$m.s^{-1}$
Component velocity along Z	VitesseZ	$m.s^{-1}$
Mass balance on each cell	Divergence_U	$m^3.s^{-1}$
Irradiancy	Irradiance	$W.m^{-2}$
Q-criteria	Critere_Q	s^{-1}
Distance to the wall $Y^+ = yU/\nu$ (only computed on boundaries of wall type)	Y_plus	dimensionless
Friction velocity	U_star	$m.s^{-1}$
Void fraction	alpha	dimensionless
Cell volumes	Volume_maille	m^3
Chemical potential	Potentiel_Chimique_Generalise	
Source term in non Galilean referential	Acceleration_terme_source	$m.s^{-2}$
Stability time steps	Pas_de_temps	S
Listing of boundary fluxes	Flux_bords	cf each *.out file
Volumetric porosity	Porosite_volumique	dimensionless
Distance to the wall	Distance_Paroi ³	m
Volumic thermal power	Puissance_volumique	$W.m^{-3}$
Local shear strain rate defined as $\sqrt{(2S_{ij}S_{ij})}$	Taux_cisaillement	s^{-1}
Cell Courant number (VDF only)	Courant_maille	dimensionless
Cell Reynolds number (VDF only)	Reynolds_maille	dimensionless
Viscous force	viscous_force	$kg.m^2.s^{-1}$
Pressure force	pressure_force	$kg.m^2.s^{-1}$
Total force	total_force	$kg.m^2.s^{-1}$
Viscous force along X	viscous_force_x	$kg.m^2.s^{-1}$
Viscous force along Y	viscous_force_y	$kg.m^2.s^{-1}$
Viscous force along Z	viscous_force_z	$kg.m^2.s^{-1}$
Pressure force along X	pressure_force_x	$kg.m^2.s^{-1}$
Pressure force along Y	pressure_force_y	$kg.m^2.s^{-1}$
Pressure force along Z	pressure_force_z	$kg.m^2.s^{-1}$
Total force along X	total_force_x	$kg.m^2.s^{-1}$
Total force along Y	total_force_y	$kg.m^2.s^{-1}$
Total force along Z	total_force_z	$kg.m^2.s^{-1}$

3 interprete

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

See also: objet_u (40) { (3.29) } (3.58) export (3.44) ecrire_fichier_xyz_valeur (3.40) option_vdf (3.82) criteres_convergence (3.27) residuals (3.108) espece (3.42) mass_source (3.74) Option_PolyMAC (3.15) Op_Conv_EF_Stab_PolyMAC_Face (3.7) Op_Conv_EF_Stab_PolyMAC_P0P1NC_Elem (3.8) Op_Conv_EF_Stab_PolyMAC_P0P1NC_Face (3.9) Op_Conv_EF_Stab_PolyMAC_P0_Face (3.10) Option_DG (3.12) verifiercoin (3.132) scatter (3.110) read_med (3.18) integrer_champ_med (3.62) ecriturelecturespecial (3.41) facsec_expert (3.56) trianguler (3.126) nettoiepasnoeuds (3.81) extraire_surface (3.49) precisiongeom (3.90) tetraedriser (3.120) redresser_hexaedres_vdf (3.99) Raffiner_isotrope_parallele (3.17) transformer (3.125)

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

[modifydomaineAxi1d \(3.77\)](#) [modif_bord_to_raccord \(3.76\)](#) [remove_invalid_internal_boundaries \(3.104\)](#)
[extrudebord \(3.50\)](#) [analyse_angle \(3.20\)](#) [lire_ideas \(3.69\)](#) [extruder \(3.52\)](#) [reorienter_triangles \(3.106\)](#) [corriger-
_frontiere_periodique \(3.26\)](#) [reorienter_tetraedres \(3.105\)](#) [refine_mesh \(3.100\)](#) [bidim_axi \(3.23\)](#) [extraire-
_plan \(3.48\)](#) [dimension \(3.33\)](#) [polyedriser \(3.88\)](#) [orientefacesbord \(3.83\)](#) [orienter_simplexes \(3.98\)](#) [verifier-
_qualite_raffinements \(3.129\)](#) [interprete_geometrique_base \(3.63\)](#) [distance_parois \(3.37\)](#) [extrudeparois \(3.51\)](#)
[reordonner \(3.107\)](#) [calculer_moments \(3.24\)](#) [regroupebord \(3.101\)](#) [extract_2d_from_3d \(3.45\)](#) [raffiner-
_anisotrope \(3.91\)](#) [mailler \(3.71\)](#) [discretiser_domaine \(3.35\)](#) [maillerparallel \(3.73\)](#) [axi \(3.22\)](#) [extruder-
_en20 \(3.54\)](#) [rotation \(3.109\)](#) [imprimer_flux \(3.59\)](#) [lire_tgrid \(3.96\)](#) [dilate \(3.32\)](#) [supprime_bord \(3.114\)](#)
[decouper_bord_coincident \(3.31\)](#) [decoupebord_pour_rayonnement \(3.30\)](#) [remove_elem \(3.102\)](#) [raffiner-
_isotrope \(3.92\)](#) [extraire_domaine \(3.47\)](#) [verifier_simplexes \(3.131\)](#) [partition_multi \(3.86\)](#) [partition \(3.84\)](#)
[associate \(3.21\)](#) [debog \(3.28\)](#) [discretize \(3.36\)](#) [solve \(3.112\)](#) [testeur \(3.118\)](#) [end \(3.57\)](#) [read \(3.93\)](#) [My-
_Comm_Group \(3.6\)](#) [mkdir \(3.75\)](#) [ecrire_fichier_bin \(3.135\)](#) [system \(3.116\)](#) [stat_per_proc_perf_log \(3.113\)](#)
[disable_TU \(3.34\)](#) [MultipleFiles \(3.5\)](#) [Option_Interpolation \(3.14\)](#) [ecrire \(3.134\)](#) [read_file \(3.94\)](#) [execute-
_parallel \(3.43\)](#) [testeur_medcoupling \(3.119\)](#) [pilote_icoco \(3.87\)](#) [test_solveur \(3.117\)](#) [postraiter_domaine
\(3.89\)](#) [lml_2_lata \(3.70\)](#) [lata_2_other \(3.68\)](#) [ecrire_champ_med \(3.38\)](#) [Write_MED \(3.2\)](#) [Merge_MED
\(3.4\)](#) [lata_2_med \(3.66\)](#) [lata_to_CGNS \(3.64\)](#) [Link_CGNS_Files \(3.3\)](#) [Option_CGNS \(3.11\)](#) [moyenne-
_volumique \(3.78\)](#) [Parallel_io_parameters \(3.16\)](#) [Option_IJK \(3.13\)](#) [Test_SSE_Kernels \(3.19\)](#) [multigrid-
_solver \(3.79\)](#)

Usage:

interprete

3.1 Create_domain_from_sub_domain

Description: This keyword fills the domain `domaine_final` with the subdomaine `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 subdomaine into Gmsh. A MED mesh file will be saved from Gmsh and read with `Lire_Med` keyword by the TRUST data file. And with this keyword, a domain will be created for each medium in the TRUST data file.

See also: [interprete_geometrique_base \(3.63\)](#)

Usage:

Create_domain_from_sub_domain {

```

    [ domaine_final str]
    [ par_sous_dom|par_sous_zone str]
    domaine_init str

```

}

where

- **domaine_final** *str*: new domain in which faces are stored
- **par_sous_dom**|**par_sous_zone** *str*: a sub-area (a group in a MED file) allowing to choose the elements
- **domaine_init** *str*: initial domain

3.2 Write_med

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

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

Usage:

Write_MED nom_dom file

where

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

3.3 Link_cgns_files

Description: Creates a single CGNS xxxx.cgns file that links to a xxxx.grid.cgns and xxxx.solution.*.cgns files

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

Usage:

Link_CGNS_Files base_name output_name

where

- **base_name** *str*: Base name of the gid/solution cgns files.
- **output_name** *str*: Name of the output cgns file.

3.4 Merge_med

Description: This keyword allows to merge multiple MED files produced during a parallel computation into a single MED file.

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

Usage:

Merge_MED med_files_base_name time_iterations

where

- **med_files_base_name** *str*: Base name of multiple med files that should appear as base_name-
_xxxxx.med, where xxxxx denotes the MPI rank number. If you specify NOM_DU_CAS, it will automatically take the basename from your datafile's name.
- **time_iterations** *str into ['all_times', 'last_time']*: Identifies whether to merge all time iterations present in the MED files or only the last one.

3.5 Multiplefiles

Description: Change MPI rank limit for multiple files during I/O

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

Usage:

MultipleFiles type

where

- **type** *int*: New MPI rank limit

3.6 My_comm_group

Description: This keyword allows to create a user MPI Comm Group of size N using the processors allocated to TRUST. The set of processors is split in N subsets.

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

Usage:

```
My_Comm_Group {  
    group_nb int  
}
```

where

- **group_nb** *int*: Number of groups to define in your Comm Group.

3.7 Op_conv_ef_stab_polymac_face

Description: Class Op_Conv_EF_Stab_PolyMAC_Face_PolyMAC

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

Usage:

```
Op_Conv_EF_Stab_PolyMAC_Face {  
    [ alpha float ]  
}
```

where

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

3.8 Op_conv_ef_stab_polymac_p0p1nc_elem

Description: Class Op_Conv_EF_Stab_PolyMAC_P0P1NC_Elem

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

Usage:

```
Op_Conv_EF_Stab_PolyMAC_P0P1NC_Elem {  
    [ alpha float ]  
}
```

where

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

3.9 Op_conv_ef_stab_polymac_p0p1nc_face

Description: Class Op_Conv_EF_Stab_PolyMAC_P0P1NC_Face

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

Usage:

3.10 Op_conv_ef_stab_polymac_p0_face

Description: Class Op_Conv_EF_Stab_PolyMAC_P0_Face

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

Usage:

3.11 Option_cgns

Description: Class for CGNS options.

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

Usage:

```
Option_CGNS {  
    [ single_precision ]  
    [ parallel_over_zone ]  
    [ use_links ]  
    [ file_per_comm_group ]  
    [ single_safe_file ]  
    [ close_every_n int]  
    [ flush_every_n int]  
}
```

where

- **single_precision** : If used, data will be written with a single_precision format inside the CGNS file (it concerns both mesh coordinates and field values).
- **parallel_over_zone** : If used, data will be written in separate zones (ie: one zone per processor). This is not so performant but easier to read later ...
- **use_links** : If used, data will be written in separate files; one file for mesh, and then one file for solution time. Links will be used.
- **file_per_comm_group** : If used, data will be written (at each comm group) in separate files; one file for mesh, and then one file for solution time. Links will be used.
- **single_safe_file** : If used, data will be written in a single file that will be opened and closed at each dt post so that file can be visualized in live. Safer if simulation stops, the file can be used.
- **close_every_n** *int*: Used to fix the opening/closing frequency when the SINGLE_SAFE_FILE option is used.
- **flush_every_n** *int*: Used to fix the flush-to-disc frequency when the SINGLE_SAFE_FILE option is used.

3.12 Option_dg

Description: Class for DG options.

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

Usage:

```
Option_DG {  
    [ order int]  
    [ velocity_order int]  
    [ pressure_order int]
```

```

    [ temperature_order int ]
    [ gram_schmidt int ]

```

```

}

```

where

- **order** *int*: global order for the DG unknowns (1 by default)
- **velocity_order** *int*: optional order for DG velocity unknown
- **pressure_order** *int*: optional order for DG pressure unknown
- **temperature_order** *int*: optional order for DG temperature unknown
- **gram_schmidt** *int*: Gram Schmidt orthogonalization (1 by default)

3.13 Option_ijk

Description: Class of IJK options.

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

Usage:

```

Option_IJK {
    [ check_divergence ]
    [ disable_diphasique ]

```

```

}

```

where

- **check_divergence** : Flag to compute and print the value of $\text{div}(\mathbf{u})$ after each pressure-correction
- **disable_diphasique** : Disable all calculations related to interfaces (phase properties, interfacial force, ...)

3.14 Option_interpolation

Description: Class for interpolation fields using MEDCoupling.

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

Usage:

```

Option_Interpolation {
    [ without_declsans_dec ]
    [ sharing_algo int ]

```

```

}

```

where

- **without_declsans_dec** : Use remapper even for a parallel calculation
- **sharing_algo** *int*: Setting the DEC sharing algo : 0,1,2

3.15 Option_polymac

Description: Class of PolyMAC options.

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

Usage:

```
Option_PolyMAC {  
    [ use_osqp ]  
    [ vdf_meshmaillage_vdf ]  
    [ interp_ve1 ]  
    [ traitement_axi ]  
}
```

}

where

- **use_osqp** : Flag to use the old formulation of the M2 matrix provided by the OSQP library. Only useful for PolyMAC version.
- **vdf_meshmaillage_vdf** : Flag used to force the calculation of the equiv tab.
- **interp_ve1** : Flag to enable a first-order face-to-element velocity interpolation. By default, it is not activated which means a second order interpolation. Only useful for PolyMAC_P0 version.
- **traitement_axi** : Flag used to relax the time-step stability criterion in case of a thin slice geometry while modelling an axi-symmetrical case. Only useful for PolyMAC_P0 version.

3.16 Parallel_io_parameters

Description: Object to handle parallel files in IJK discretization

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

Usage:

```
Parallel_io_parameters {  
    [ block_size_bytes int]  
    [ block_size_megabytes int]  
    [ writing_processes int]  
    [ bench_ijk_splitting_write str]  
    [ bench_ijk_splitting_read str]  
}
```

}

where

- **block_size_bytes** *int*: File writes will be performed by chunks of this size (in bytes). This parameter will not be taken into account if **block_size_megabytes** has been defined
- **block_size_megabytes** *int*: File writes will be performed by chunks of this size (in megabytes). The size should be a multiple of the GPFS block size or lustre stripping size (typically several megabytes)
- **writing_processes** *int*: This is the number of processes that will write concurrently to the file system (this must be set according to the capacity of the filesystem, set to 1 on small computers, can be up to 64 or 128 on very large systems).
- **bench_ijk_splitting_write** *str*: Name of the splitting object we want to use to run a parallel write bench (optional parameter)
- **bench_ijk_splitting_read** *str*: Name of the splitting object we want to use to run a parallel read bench (optional parameter)

3.17 Raffiner_isotrope_parallele

Description: Refine parallel mesh in parallel

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

Usage:

```
Raffiner_isotrope_parallele {  
    name_of_initial_zones|name_of_initial_domaines str  
    name_of_new_zones|name_of_new_domaines str  
    [ ascii ]  
    [ single_hdf ]  
}
```

where

- **name_of_initial_zones|name_of_initial_domaines** *str*: name of initial Domaines
- **name_of_new_zones|name_of_new_domaines** *str*: name of new Domaines
- **ascii** : writing Domaines in ascii format
- **single_hdf** : writing Domaines in hdf format

3.18 Read_med

Synonymous: **lire_med**

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

Note about naming boundaries: When reading 'file', TRUST will detect boundaries between domains (Raccord) when the name of the boundary begins by 'type_raccord -_'. For example, a boundary named type_raccord_wall in 'file' will be considered by TRUST as a boundary named 'wall' between two domains.

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

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

Read_Med

Read_file domain_name_ssz.geo ;

During the parallel calculation, you will include something:

Scatter { ... }

Read_file domain_name_ssz_par.geo ;

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

Usage:

```
read_med {  
    [ convertalltopoly ]  
    domaine|domain str  
    fichier|file str  
    [ maillage|mesh str ]  
    [ exclure_groupes|exclude_groups n word1 word2 ... wordn ]  
    [ inclure_groupes_faces_additionnels|include_additional_face_groups n word1 word2 ... wordn ]  
}
```

}
where

- **convertalltopoly** : Option to convert mesh with mixed cells into polyhedral/polygonal cells
- **domain|domain** *str*: Corresponds to the domain name.
- **fichier|file** *str*: File (written in the MED format, with extension '.med') containing the mesh
- **maillage|mesh** *str*: Name of the mesh in med file. If not specified, the first mesh will be read.
- **exclure_groupe|exclude_groups** *n word1 word2 ... wordn*: List of face groups to skip in the MED file.
- **inclure_groupe|faces_additionnels|include_additional_face_groups** *n word1 word2 ... wordn*: List of face groups to read and register in the MED file.

3.19 Test_sse_kernels

Description: Object to test the different kernel methods used in the multigrid solver in IJK discretization

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

Usage:

```
Test_SSE_Kernels {  
    [ nmax int ]
```

```
}
```

where

- **nmax** *int*: Number of tests we want to perform

3.20 Analyse_angle

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

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

Usage:

```
analyse_angle domain_name nb_histo  
where
```

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

3.21 Associate

Synonymous: **associer**

Description: This interpreter allows one object to be associated with another. The order of the two objects in this instruction is not important. The object objet_2 is associated to objet_1 if this makes sense; if not either objet_1 is associated to objet_2 or the program exits with error because it cannot execute the Associate (Associer) instruction. For example, to calculate water flow in a pipe, a Pb_Hydraulique type object needs to be defined. But also a Domaine type object to represent the pipe, a Scheme_euler_explicit

type object for time discretization, a discretization type object (VDF or VEF) and a `Fluide_Incompressible` type object which will contain the water properties. These objects must then all be associated with the problem.

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

Usage:

associate objet_1 objet_2

where

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

3.22 Axi

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

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

Usage:

axi

3.23 Bidim_axi

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

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

Usage:

bidim_axi

3.24 Calculer_moments

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

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

Usage:

calculer_moments nom_dom mot

where

- **nom_dom** *str*: Name of domain.
- **mot** *lecture_bloc_moment_base* ([3.25](#)): Keyword.

3.25 Lecture_bloc_moment_base

Description: Auxiliary class to compute and print the moments.

See also: [objet_lecture \(39\)](#) [calcul \(3.25.1\)](#) [centre_de_gravite \(3.25.2\)](#)

Usage:

3.25.1 Calcul

Description: The centre of gravity will be calculated.

See also: (3.25)

Usage:

calcul

3.25.2 Centre_de_gravite

Description: To specify the centre of gravity.

See also: (3.25)

Usage:

centre_de_gravite point

where

- **point** *un_point* (3.25.3): A centre of gravity.

3.25.3 Un_point

Description: A point.

See also: objet_lecture (39)

Usage:

pos

where

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

3.26 Corriger_frontiere_periodique

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

See also: interprete (3)

Usage:

corriger_frontiere_periodique {

domaine *str*

bord *str*

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

[**fichier_post** *str*]

}

where

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

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

3.27 Criteres_convergence

Description: convergence criteria

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

Usage:

aco [**inco**] [**val**] **acof**

where

- **aco** *str* into [' ']: Opening curly bracket.
- **inco** *str*: Unknown (i.e: *alpha*, *temperature*, *velocity* and *pressure*)
- **val** *float*: Convergence threshold
- **acof** *str* into [' ']: Closing curly bracket.

3.28 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 *Kernel/Framework/Resoudre.cpp* file the instruction: *Debug::verifier(msg,value)*; Where *msg* is a string and *value* may be a double, an integer or an array.

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

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

Usage:

debug pb fichier1 fichier2 seuil mode

where

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

3.29 {

Description: Block's beginning.

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

Usage:
{

3.30 Decoupebord_pour_rayonnement

Synonymous: **decoupebord**

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

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

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

Usage:

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

where

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

3.31 Decouper_bord_coincident

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

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

Usage:

decouper_bord_coincident domain_name bord

where

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

3.32 Dilate

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

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

Usage:

dilate domain_name alpha

where

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

3.33 Dimension

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

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

Usage:

dimension dim

where

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

3.34 Disable_tu

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

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

Usage:

disable_TU

3.35 Discretiser_domaine

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

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

Usage:

discretiser_domaine domain_name

where

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

3.36 Discretize

Synonymous: **discretiser**

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

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

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

Usage:

discretize problem_name dis

where

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

3.37 Distance_pari

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

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

Usage:

distance_pari dom bords format

where

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

3.38 Ecrire_champ_med

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

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

Usage:

ecrire_champ_med nom_dom nom_chp file

where

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

3.39 Ecrire_fichier_formatte

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

See also: `ecrire_fichier_bin` ([3.135](#))

Usage:

ecrire_fichier_formatte `name_obj` `filename`

where

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

3.40 Ecrire_fichier_xyz_valeur

Description: This keyword is used to write the values of a field only for some boundaries in a text file with the following format: `n_valeur`

`x_1 y_1 [z_1] val_1`

...

`x_n y_n [z_n] val_n`

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

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

Usage:

ecrire_fichier_xyz_valeur {

 [**binary_file**]

 [**dt** *float*]

 [**fields** *n word1 word2 ... wordn*]

 [**boundaries** *n word1 word2 ... wordn*]

}

where

- **binary_file** : To write file in binary format
- **dt** *float*: File writing frequency
- **fields** *n word1 word2 ... wordn*: Names of the fields we want to write
- **boundaries** *n word1 word2 ... wordn*: Names of the boundaries on which to write fields

3.41 Ecriturelecturespecial

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

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

Usage:

ecriturelecturespecial `type`

where

- **type** *str*: If set to 0, no xyz file is created. If set to 1 (the default) the .xyz file is written at the end of the computation.

3.42 Espece

Description: not_set

See also: interpret (3)

Usage:

```
espece {  
    mu champ_base  
    cp champ_base  
    masse_molaire float  
}  
where
```

- **mu** *champ_base* (16.1): Species dynamic viscosity value (kg.m-1.s-1).
- **cp** *champ_base* (16.1): Species specific heat value (J.kg-1.K-1).
- **masse_molaire** *float*: Species molar mass.

3.43 Execute_parallel

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

See also: interpret (3)

Usage:

```
execute_parallel {  
    liste_cas n word1 word2 ... wordn  
    [ nb_procs n n1 n2 ... nn ]  
}  
where
```

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

3.44 Export

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

See also: interpret (3)

Usage:

```
export
```

3.45 Extract_2d_from_3d

Description: Keyword to extract a 2D mesh by selecting a boundary of the 3D mesh. To generate a 2D axisymmetric mesh prefer Extract_2Daxi_from_3D keyword.

See also: [interpret \(3\)](#) [extract_2daxi_from_3d \(3.46\)](#)

Usage:

extract_2d_from_3d **dom3D** **bord** **dom2D**

where

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

3.46 Extract_2daxi_from_3d

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

See also: [extract_2d_from_3d \(3.45\)](#)

Usage:

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

where

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

3.47 Extraire_domaine

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

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

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

Usage:

extraire_domaine {

domaine *str*

probleme *str*

 [**condition_elements** *str*]

 [**sous_zone**|**sous_domaine** *str*]

}

where

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

3.48 Extraire_plan

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

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

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

Usage:

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

where

- **domaine** *str*: domain name
- **probleme** *str*: `pb_name`
- **origine** *n x1 x2 ... xn*
- **point1** *n x1 x2 ... xn*
- **point2** *n x1 x2 ... xn*
- **point3** *n x1 x2 ... xn*
- **triangle**
- **epaisseur** *float*: thickness
- **via_extraire_surface**
- **inverse_condition_element**
- **avec_certains_bords_pour_extraire_surface** *n word1 word2 ... wordn*: name of boundaries to include when extracting plan

3.49 Extraire_surface

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

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

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

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

See also: `interpret` (3)

Usage:

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

where

- **domaine** *str*: Domain in which faces are saved
- **probleme** *str*: Problem from which faces should be extracted
- **condition_elements** *str*: condition on center of elements
- **condition_faces** *str*
- **avec_les_bords**
- **avec_certaines_bords** *n word1 word2 ... wordn*

3.50 Extrudebord

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

Warning: If the initial domain is a tetrahedral mesh, the boundary will be moved in the XY plane then extrusion will be applied (you should maybe use the `Transformer` keyword on the final domain to have the domain you really want). You can use the keyword `Postraiter_domaine` to generate a `latalmedl...` 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 hexahedral mesh. This periodic box may be used then to engender turbulent inlet flow condition for the main domain.

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

See also: `interpret` (3)

Usage:

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

where

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

3.51 Extrudeparoi

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

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

Usage:

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

where

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

3.52 Extruder

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

See also: [interprete \(3\)](#) [extruder_en3 \(3.55\)](#)

Usage:

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

where

- **domaine** *str*: Name of the domain.
- **nb_tranches** *int*: Number of elements in the extrusion direction.
- **direction** *troisf* [\(3.53\)](#): Direction of the extrude operation.

3.53 Troisf

Description: Auxiliary class to extrude.

See also: `objet_lecture` ([39](#))

Usage:

lx ly lz

where

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

3.54 Extruder_en20

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

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

Usage:

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

where

- **domaine** *str*: Name of the domain.
- **nb_tranches** *int*: Number of elements in the extrusion direction.
- **direction** *troisf* ([3.53](#)): 0 Direction of the extrude operation.

3.55 Extruder_en3

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

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

See also: `extruder` ([3.52](#))

Usage:

```
extruder_en3 {  
    domaine n word1 word2 ... wordn  
    [ nom_cl_devant str]  
    [ nom_cl_derriere str]  
    nb_tranches int
```



```

    direction troisf
}
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.
- **nb_tranches** *int* for inheritance: Number of elements in the extrusion direction.
- **direction** *troisf* (3.53) for inheritance: Direction of the extrude operation.

3.56 Facsec_expert

Description: To parameter the safety factor for the time step during the simulation.

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

Usage:

```

facsec_expert {
    [ facsec_ini float ]
    [ facsec_max float ]
    [ rappport_residus float ]
    [ nb_ite_sans_accel_max int ]
}
where

```

- **facsec_ini** *float*: Initial facsec taken into account at the beginning of the simulation.
- **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.
- **rappport_residus** *float*: Ratio between the residual at time n and the residual at time n+1 above which the facsec is increased by multiplying by sqrt(rappport_residus) (1.2 by default).
- **nb_ite_sans_accel_max** *int*: Maximum number of iterations without facsec increases (20000 by default): if facsec does not increase with the previous condition (ration between 2 consecutive residuals too high), we increase it by force after nb_ite_sans_accel_max iterations.

3.57 End

Synonymous: **fin**

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

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

Usage:

end

3.58 }

Description: Block's end.

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

Usage:

}

3.59 Imprimer_flux

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

See also: [interpret \(3\)](#) [imprimer_flux_sum \(3.61\)](#)

Usage:

imprimer_flux domain_name noms_bord

where

- **domain_name** *str*: Name of the domain.
- **noms_bord** *bloc_lecture (3.60)*: List of boundaries, for ex: { Bord1 Bord2 }

3.60 Bloc_lecture

Description: to read between two braces

See also: [objet_lecture \(39\)](#) [bloc_criteres_convergence \(3.60.1\)](#) [solveur_petsc_option_cli \(3.60.2\)](#)

Usage:

bloc_lecture

where

- **bloc_lecture** *str*

3.60.1 Bloc_criteres_convergence

Description: Not set

See also: [\(3.60\)](#)

Usage:

bloc_lecture

where

- **bloc_lecture** *str*

3.60.2 Solveur_petsc_option_cli

Description: solver

See also: (3.60)

Usage:

bloc_lecture

where

- **bloc_lecture** *str*

3.61 Imprimer_flux_sum

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

See also: imprimer_flux (3.59)

Usage:

imprimer_flux_sum domain_name noms_bord

where

- **domain_name** *str*: Name of the domain.
- **noms_bord** *bloc_lecture* (3.60): List of boundaries, for ex: { Bord1 Bord2 }

3.62 Integrer_champ_med

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

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

See also: interprete (3)

Usage:

integrer_champ_med {

champ_med *str*

methode *str* into ['integrale_en_z', 'debit_total']

[**zmin** *float*]

[**zmax** *float*]

[**nb_tranche** *int*]

[**fichier_sortie** *str*]

}
where

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

3.63 Interpret_geometrique_base

Description: Class for interpreting a data file

See also: [interpret](#) (3) [Create_domain_from_sub_domain](#) (3.1)

Usage:

interpret_geometrique_base

3.64 Lata_to_cgns

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

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

Usage:

lata_to_CGNS [**format**] **file** **file_CGNS**

where

- **format** *format_lata_to_cgns* (3.65): generated file post_CGNS.data use format (CGNS or LATA or LML keyword).
- **file** *str*: LATA file to convert to the new format.
- **file_CGNS** *str*: Name of the CGNS file.

3.65 Format_lata_to_cgns

Description: not_set

See also: [objet_lecture](#) (39)

Usage:

mot [**format**]

where

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

3.66 Lata_2_med

Synonymous: **lata_to_med**

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

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

Usage:

lata_2_med [**format**] **file** **file_med**

where

- **format** *format_lata_to_med* (3.67): generated file post_med.data use format (MED or LATA or LML keyword).
- **file** *str*: LATA file to convert to the new format.
- **file_med** *str*: Name of the MED file.

3.67 Format_lata_to_med

Description: not_set

See also: [objet_lecture](#) (39)

Usage:

mot [**format**]

where

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

3.68 Lata_2_other

Synonymous: **lata_to_other**

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

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

Usage:

lata_2_other [**format**] **file** **file_post**

where

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

3.69 Lire_ideas

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

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

Usage:

lire_ideas nom_dom file

where

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

3.70 Lml_2_lata

Synonymous: **lml_to_lata**

Description: To convert results file written with LML format to a single LATA file.

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

Usage:

lml_2_lata file_lml file_lata

where

- **file_lml** *str*: LML file to convert to the new format.
- **file_lata** *str*: Name of the single LATA file.

3.71 Mailler

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

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

Usage:

mailler domaine bloc

where

- **domaine** *str*: Name of domain.
- **bloc** *list_bloc_mailler* ([3.72](#)): Instructions to mesh.

3.72 List_bloc_mailler

Description: List of block mesh.

See also: [listobj \(38.5\)](#)

Usage:

{ *object1* , *object2* }

list of *mailler_base* ([3.72.1](#)) separated with ,

3.72.1 Mailler_base

Description: Basic class to mesh.

See also: objet_lecture (39) pave (3.72.2) epsilon (3.72.12) domain (3.72.13)

Usage:

3.72.2 Pave

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

See also: mailler_base (3.72.1)

Usage:

pave name bloc list_bord
where

- **name** *str*: Name of the pave (block).
- **bloc** *bloc_pave* (3.72.3): Definition of the pave (block).
- **list_bord** *list_bord* (3.72.4): Domain boundaries definition.

3.72.3 Bloc_pave

Description: Class to create a pave.

See also: objet_lecture (39)

Usage:

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

where

- **Origine** *x1 x2 (x3)*: Keyword to define the pave (block) origin, that is to say one of the 8 block points (or 4 in a 2D coordinate system).
- **longueurs** *x1 x2 (x3)*: Keyword to define the block dimensions, that is to say knowing the origin, length along the axes.

- **nombre_de_noeuds** *n1 n2 (n3)*: Keyword to define the discretization (nodenum) in each direction.
- **facteurs** *x1 x2 (x3)*: Keyword to define stretching factors for mesh discretization in each direction. This is a real number which must be positive (by default 1.0). A stretching factor other than 1 allows refinement on one edge in one direction.
- **symx**: Keyword to define a block mesh that is symmetrical with respect to the YZ plane (respectively Y-axis in 2D) passing through the block centre.
- **symy**: Keyword to define a block mesh that is symmetrical with respect to the XZ plane (respectively X-axis in 2D) passing through the block centre.
- **symz**: Keyword defining a block mesh that is symmetrical with respect to the XY plane passing through the block centre.
- **xtanh** *float*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the X-direction.
- **xtanh_dilatation** *int into [-1, 0, 1]*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the X-direction. *xtanh_dilatation*: The value may be -1,0,1 (0 by default): 0: coarse mesh at the middle of the channel and smaller near the walls -1: coarse mesh at the left side of the channel and smaller at the right side 1: coarse mesh at the right side of the channel and smaller near the left side of the channel.
- **xtanh_taille_premiere_maille** *float*: Size of the first cell of the mesh with tanh (hyperbolic tangent) variation in the X-direction.
- **ytnh** *float*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the Y-direction.
- **ytnh_dilatation** *int into [-1, 0, 1]*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the Y-direction. *ytnh_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.
- **ytnh_taille_premiere_maille** *float*: Size of the first cell of the mesh with tanh (hyperbolic tangent) variation in the Y-direction.
- **ztnh** *float*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the Z-direction.
- **ztnh_dilatation** *int into [-1, 0, 1]*: Keyword to generate mesh with tanh (hyperbolic tangent) variation in the Z-direction. *ztnh_dilatation*: The value may be -1,0,1 (0 by default): 0: coarse mesh at the middle of the channel and smaller near the walls -1: coarse mesh at the back of the channel and smaller near the front 1: coarse mesh at the front of the channel and smaller near the back.
- **ztnh_taille_premiere_maille** *float*: Size of the first cell of the mesh with tanh (hyperbolic tangent) variation in the Z-direction.

3.72.4 List_bord

Description: The block sides.

See also: [listobj \(38.5\)](#)

Usage:

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

list of *bord_base* ([3.72.5](#))

3.72.5 Bord_base

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

See also: [objet_lecture \(39\)](#) [raccord \(3.72.6\)](#) [internes \(3.72.10\)](#) [bord \(3.72.11\)](#)

Usage:

3.72.6 Raccord

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

See also: `bord_base` ([3.72.5](#))

Usage:

raccord type1 type2 nom defbord

where

- **type1** *str* into ['local', 'distant']: Contact type.
- **type2** *str* into ['homogene']: Contact type.
- **nom** *str*: Name of block side.
- **defbord** *defbord* ([3.72.7](#)): Definition of block side.

3.72.7 Defbord

Description: Class to define an edge.

See also: `objet_lecture` ([39](#)) `defbord_2` ([3.72.8](#)) `defbord_3` ([3.72.9](#))

Usage:

3.72.8 Defbord_2

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

See also: ([3.72.7](#))

Usage:

dir eq pos pos2_min inf1 dir2 inf2 pos2_max

where

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

3.72.9 Defbord_3

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

See also: ([3.72.7](#))

Usage:

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

where

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

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

3.72.10 Internes

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

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

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

See also: [bord_base \(3.72.5\)](#)

Usage:

internes nom defbord

where

- **nom** *str*: Name of block side.
- **defbord** *defbord (3.72.7)*: Definition of block side.

3.72.11 Bord

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

See also: [bord_base \(3.72.5\)](#)

Usage:

bord nom defbord

where

- **nom** *str*: Name of block side.
- **defbord** *defbord (3.72.7)*: Definition of block side.

3.72.12 Epsilon

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

See also: [mailler_base \(3.72.1\)](#)

Usage:

epsilon eps

where

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

3.72.13 Domain

Description: Class to reuse a domain.

See also: `mailler_base` (3.72.1)

Usage:

domain *domain_name*

where

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

3.73 Maillerparallel

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

See also: `interpret` (3)

Usage:

maillerparallel {

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

}

where

- **domain** *str*: the name of the domain to mesh (it must be an empty domain object).
- **nb_nodes** *n n1 n2 ... nn*: dimension defines the spatial dimension (currently only dimension=3 is supported), and nX, nY and nZ defines the total number of nodes in the mesh in each direction.
- **splitting** *n n1 n2 ... nn*: dimension is the spatial dimension and npartsX, npartsY and npartsZ are the number of parts created. The product of the number of parts must be equal to the number of processors used for the computation.
- **ghost_thickness** *int*: the number of ghost cells (equivalent to the `epaisseur_joint` parameter of `Decouper`).
- **perio_x** : change the splitting method to provide a valid mesh for periodic boundary conditions.

- **perio_y** : change the splitting method to provide a valid mesh for periodic boundary conditions.
- **perio_z** : change the splitting method to provide a valid mesh for periodic boundary conditions.
- **function_coord_x** *str*: By default, the meshing algorithm creates nX nY nZ coordinates ranging between 0 and 1 (eg a unity size box). If **function_coord_x** is specified, it is used to transform the [0,1] segment to the coordinates of the nodes. **funcX** must be a function of the x variable only.
- **function_coord_y** *str*: like **function_coord_x** for y
- **function_coord_z** *str*: like **function_coord_x** for z
- **file_coord_x** *str*: Keyword to read the Nx floating point values used as nodes coordinates in the file.
- **file_coord_y** *str*: idem **file_coord_x** for y
- **file_coord_z** *str*: idem **file_coord_x** for z
- **boundary_xmin** *str*: the name of the boundary at the minimum X direction. If it not provided, the default boundary names are xmin, xmax, ymin, ymax, zmin and zmax. If the mesh is periodic in a given direction, only the MIN boundary name is used, for both sides of the box.
- **boundary_xmax** *str*
- **boundary_ymin** *str*
- **boundary_ymax** *str*
- **boundary_zmin** *str*
- **boundary_zmax** *str*

3.74 Mass_source

Description: Mass source used in a dilatable simulation to add/reduce a mass at the boundary (volumetric source in the first cell of a given boundary).

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

Usage:

```
mass_source {
    bord str
    surfacic_flux champ_front_base
}
```

where

- **bord** *str*: Name of the boundary where the source term is applied
- **surfacic_flux** *champ_front_base* ([17.1](#)): The boundary field that the user likes to apply: for example, *champ_front_uniforme*, *ch_front_input_uniform* or *champ_front_fonc_t*

3.75 Mkdir

Description: equivalent to system mkdir

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

Usage:

```
mkdir directory
where
```

- **directory** *str*: directory to create

3.76 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: [interpret \(3\)](#)

Usage:

modif_bord_to_raccord **domaine** **nom_bord**

where

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

3.77 Modifydomaineaxi1d

Description: Convert a 1D mesh to 1D axisymmetric mesh

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

Usage:

modifydomaineAxi1d **dom** **bloc**

where

- **dom** *str*
- **bloc** *bloc_lecture* ([3.60](#))

3.78 Moyenne_volumique

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

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

Usage:

moyenne_volumique {

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

}

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
- **format_post** *str*: gives the fileformat for the result (by default : lata)
- **nom_fichier_post** *str*: indicates the filename where the result is written
- **fonction_filtre** *bloc_lecture* (3.60): 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.

- **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.

3.79 Multigrid_solver

Description: Object defining a multigrid solver in IJK discretization

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

Usage:

```
multigrid_solver {
  [ coarsen_operators coarsen_operators]
  [ ghost_size int]
  [ relax_jacobi n x1 x2 ... xn]
  [ pre_smooth_steps n n1 n2 ... nn]
  [ smooth_steps n n1 n2 ... nn]
  [ nb_full_mg_steps n n1 n2 ... nn]
  [ solveur_grossier solveur_sys_base]
  [ seuil float]
  [ impr ]
  [ solver_precision str into ['mixed', 'double']]
  [ iterations_mixed_solver int]
```

```
}
```

where

- **coarsen_operators** *coarsen_operators* (3.80): Definition of the number of grids that will be used, in addition to the finest (original) grid, followed by the list of the coarsen operators that will be applied to get those grids
- **ghost_size** *int*: Number of ghost cells known by each processor in each of the three directions
- **relax_jacobi** *n x1 x2 ... xn*: Parameter between 0 and 1 that will be used in the Jacobi method to solve equation on each grid. Should be around 0.7
- **pre_smooth_steps** *n n1 n2 ... nn*: First integer of the list indicates the numbers of integers that has to be read next. Following integers define the numbers of iterations done before solving the equation on each grid. For example, 2 7 8 means that we have a list of 2 integers, the first one tells us to perform 7 pre-smooth steps on the first grid, the second one tells us to perform 8 pre-smooth steps on the second grid. If there are more than 2 grids in the solver, then the remaining ones will have as many pre-smooth steps as the last mentioned number (here, 8)
- **smooth_steps** *n n1 n2 ... nn*: First integer of the list indicates the numbers of integers that has to be read next. Following integers define the numbers of iterations done after solving the equation on each grid. Same behavior as `pre_smooth_steps`
- **nb_full_mg_steps** *n n1 n2 ... nn*: Number of multigrid iterations at each level
- **solveur_grossier** *solveur_sys_base* (11.16): Name of the iterative solver that will be used to solve the system on the coarsest grid. This resolution must be more precise than the ones occurring on the fine grids. The threshold of this solver must therefore be lower than `seuil` defined above.
- **seuil** *float*: Define an upper bound on the norm of the final residue (i.e. the one obtained after applying the multigrid solver). With hybrid precision, as long as we have not obtained a residue whose norm is lower than the imposed threshold, we keep applying the solver
- **impr** : Flag to display some info on the resolution on each grid
- **solver_precision** *str into ['mixed', 'double']*: Precision with which the variables at stake during the resolution of the system will be stored. We can have a simple or floatant precision or both. In the case of a hybrid precision, the multigrid solver is launched in simple precision, but the residual is calculated in floatant precision.
- **iterations_mixed_solver** *int*: Define the maximum number of iterations in mixed precision solver

3.80 Coarsen_operators

Description: `not_set`

See also: `listobj` (38.5)

Usage:

`n object1 object2`

list of *coarsen_operator_uniform* (3.80.1)

3.80.1 Coarsen_operator_uniform

Description: Object defining the uniform coarsening process of the given grid in IJK discretization

See also: `objet_lecture` (39)

Usage:

`[Coarsen_Operator_Uniform] aco [coarsen_i] [coarsen_i_val] [coarsen_j] [coarsen_j_val] [coarsen_k] [coarsen_k_val] acof`

where

- **Coarsen_Operator_Uniform** *str*
- **aco** *str into ['{']*: opening curly brace
- **coarsen_i** *str into ['coarsen_i']*

- **coarsen_i_val** *int*: Integer indicating the number by which we will divide the number of elements in the I direction (in order to obtain a coarser grid)
- **coarsen_j** *str* into ['coarsen_j']
- **coarsen_j_val** *int*: Integer indicating the number by which we will divide the number of elements in the J direction (in order to obtain a coarser grid)
- **coarsen_k** *str* into ['coarsen_k']
- **coarsen_k_val** *int*: Integer indicating the number by which we will divide the number of elements in the K direction (in order to obtain a coarser grid)
- **acof** *str* into [''] : closing curly brace

3.81 Nettoiepasnoeuds

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

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

Usage:

nettoiepasnoeuds **domain_name**

where

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

3.82 Option_vdf

Description: Class of VDF options.

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

Usage:

```
option_vdf {
    [ traitement_coins str into ['oui', 'non']]
    [ traitement_gradients str into ['oui', 'non']]
    [ p_imposee_aux_faces str into ['oui', 'non']]
    [ deactivate_arete_mixte ]
    [ toutes_les_optionslall_options ]
```

}

where

- **traitement_coins** *str* into ['oui', 'non']: Treatment of corners (yes or no). This option modifies slightly the calculations at the outlet of the plane channel. It supposes that the boundary continues after channel outlet (i.e. velocity vector remains parallel to the boundary).
- **traitement_gradients** *str* into ['oui', 'non']: Treatment of gradient calculations (yes or no). This option modifies slightly the gradient calculation at the corners and activates also the corner treatment option.
- **p_imposee_aux_faces** *str* into ['oui', 'non']: Pressure imposed at the faces (yes or no).
- **deactivate_arete_mixte** : Deactivate the arete_mixte contribution in the conv op of the momentum equation.
- **toutes_les_optionslall_options** : Activates all Option_VDF options. If used, must be used alone without specifying the other options, nor combinations.

3.83 Orientefacesbord

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

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

Usage:

orientefacesbord **domain_name**

where

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

3.84 Partition

Synonymous: **decouper**

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

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

Usage:

partition **domaine** **bloc_decouper**

where

- **domaine** *str*: Name of the domain to be cut.
- **bloc_decouper** *bloc_decouper* ([3.85](#)): Description how to cut a domain.

3.85 Bloc_decouper

Description: Auxiliary class to cut a domain.

See also: [objet_lecture \(39\)](#)

Usage:

```
{  
    [ Partition_toolpartitionneur partitionneur_deriv]  
    [ larg_joint int]  
    [ nom_zones str]  
    [ ecrire_decoupage str]  
    [ ecrire_lata str]  
    [ ecrire_med str]  
    [ nb_parts_tot int]  
    [ periodique n word1 word2 ... wordn]  
    [ reorder int]  
    [ single_hdf ]  
    [ print_more_infos int]  
}
```

where

- **Partition_tool**partitionneur *partitionneur_deriv* ([26](#)): Defines the partitionning algorithm (the effective C++ object used is 'Partitionneur_ALGORITHM_NAME').

- **larg_joint** *int*: This keyword specifies the thickness of the virtual ghost domaine (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.
- **nom_zones** *str*: Name of the files containing the different partition of the domain. The files will be :
 name_0001.Zones
 name_0002.Zones
 ...
 name_000n.Zones. If this keyword is not specified, the geometry is not written on disk (you might just want to generate a 'ecrire_decoupage' or 'ecrire_lata').
- **ecrire_decoupage** *str*: After having called the partitionning algorithm, the resulting partition is written on disk in the specified filename. See also partitionneur Fichier_Decoupage. This keyword is useful to change the partition numbers: first, you write the partition into a file with the option `ecrire_decoupage`. This file contains the domaine number for each element's mesh. Then you can easily permute domaine numbers in this file. Then read the new partition to create the .Zones files with the Fichier_Decoupage keyword.
- **ecrire_lata** *str*: Save the partition field in a LATA format file for visualization
- **ecrire_med** *str*: Save the partition field in a MED format file for visualization
- **nb_parts_tot** *int*: Keyword to generates N .Domaine 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 domaines 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.
- **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.
- **single_hdf** : Optional keyword to enable you to write the partitioned domaines in a single file in hdf5 format.
- **print_more_infos** *int*: If this option is set to 1 (0 by default), print infos about number of remote elements (ghosts) and additional infos about the quality of partitionning. Warning, it slows down the cutting operations.

3.86 Partition_multi

Synonymous: **decouper_multi**

Description: allows to partition multiple domains in contact with each other in parallel: necessary for resolution monolithique in implicit schemes and for all coupled problems using PolyMAC_POPIINC. By default, this keyword is commented in the reference test cases.

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

Usage:

partition_multi *aco domaine1 dom blocdecoupdom1 domaine2 dom2 blocdecoupdom2 acof*
 where

- **aco** *str* into ['{']: Opening curly bracket.
- **domaine1** *str* into ['domaine']: not set.
- **dom** *str*: Name of the first domain to be cut.
- **blocdecoupd1** *bloc_decouper* (3.85): Partition bloc for the first domain.
- **domaine2** *str* into ['domaine']: not set.
- **dom2** *str*: Name of the second domain to be cut.
- **blocdecoupd2** *bloc_decouper* (3.85): Partition bloc for the second domain.
- **acof** *str* into ['}']: Closing curly bracket.

3.87 Pilote_icoco

Description: not_set

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

Usage:

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

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

3.88 Polyedriser

Description: cast hexahedra into polyhedra so that the indexing of the mesh vertices is compatible with PolyMAC_POPINC discretization. Must be used in PolyMAC_POPINC discretization if a hexahedral mesh has been produced with TRUST's internal mesh generator.

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

Usage:

```
polyedriser domain_name
where
```

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

3.89 Postraiter_domaine

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

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

Usage:

```
postraiter_domaine {
    format str into ['lml', 'lata', 'single_lata', 'lata_v2', 'med', 'cgns']
    [ binaire int into [0, 1]]
```

```

[ ecrire_frontiere int into [0, 1]]
[ dual int into [0, 1]]
[ file|fichier str]
[ joints_non_postraites int into [0, 1]]
[ domain|domaine str]
[ domaines bloc_lecture]
}
where

```

- **format** *str into ['lml', 'lata', 'single_lata', 'lata_v2', 'med', 'cgns']*: File format.
- **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)
- **dual** *int into [0, 1]*: This option indicates whether the original mesh (default) or the dual one (the one used for postprocessing of field faces) is to be written.
- **file|fichier** *str*: The file name can be changed with the fichier option.
- **joints_non_postraites** *int into [0, 1]*: The joints_non_postraites (1 by default) will not write the boundaries between the partitioned mesh.
- **domain|domaine** *str*: Name of domain
- **domaines** *bloc_lecture* (3.60): Names of domains : { name1 name2 }

3.90 Precisiongeom

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

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

Usage:

```

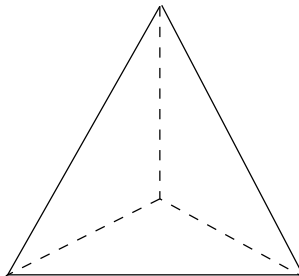
precisiongeom precision
where

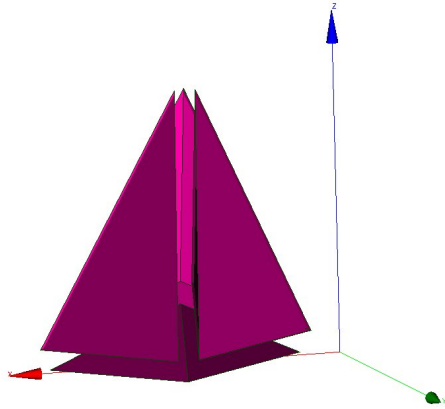
```

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

3.91 Raffiner_anisotrope

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





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

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

Usage:

raffiner_anisotrope **domain_name**

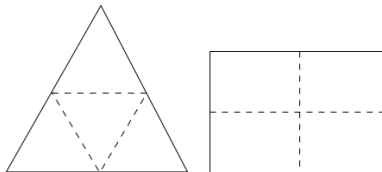
where

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

3.92 Raffiner_isotrope

Synonymous: **raffiner_simplexes**

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



For 3D elements:

.

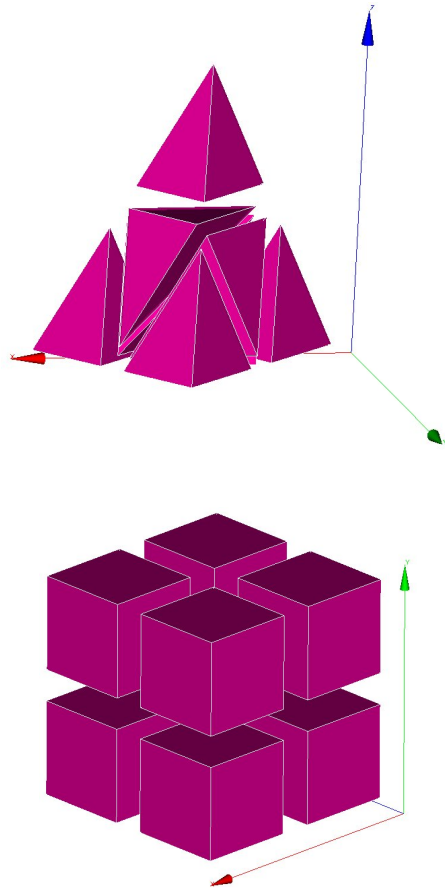
See also: [interpret \(3\)](#)

Usage:

raffiner_isotrope **domain_name**

where

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



3.93 Read

Synonymous: **lire**

Description: Interpreter to read the `a_object` object defined between the braces.

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

Usage:

read a_object bloc
where

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

3.94 Read_file

Synonymous: **lire_fichier**

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

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

If the filename is `;`, is to execute a data set given in the file of name `name_obj` (a space must be entered

between the semi-colon and the file name).

See also: [interpret \(3\)](#) [read_unsupported_ascii_file_from_icem \(3.97\)](#) [read_file_binary \(3.95\)](#)

Usage:

read_file name_obj filename

where

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

3.95 Read_file_binary

Synonymous: **lire_fichier_bin**

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

See also: [read_file \(3.94\)](#)

Usage:

read_file_binary name_obj filename

where

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

3.96 Lire_tgrid

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

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

Usage:

lire_tgrid dom filename

where

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

3.97 Read_unsupported_ascii_file_from_icem

Description: not_set

See also: [read_file \(3.94\)](#)

Usage:

read_unsupported_ascii_file_from_icem name_obj filename

where

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

3.98 Orienter_simplexes

Synonymous: **rectify_mesh**

Description: Keyword to raffine a mesh

See also: interpreté (3)

Usage:

orienter_simplexes domain_name

where

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

3.99 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: interpreté (3)

Usage:

redresser_hexaedres_vdf domain_name

where

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

3.100 Refine_mesh

Description: not_set

See also: interpreté (3)

Usage:

refine_mesh domaine

where

- **domaine** *str*

3.101 Regroupebord

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

See also: interpreté (3)

Usage:

regroupebord domaine new_bord bords

where

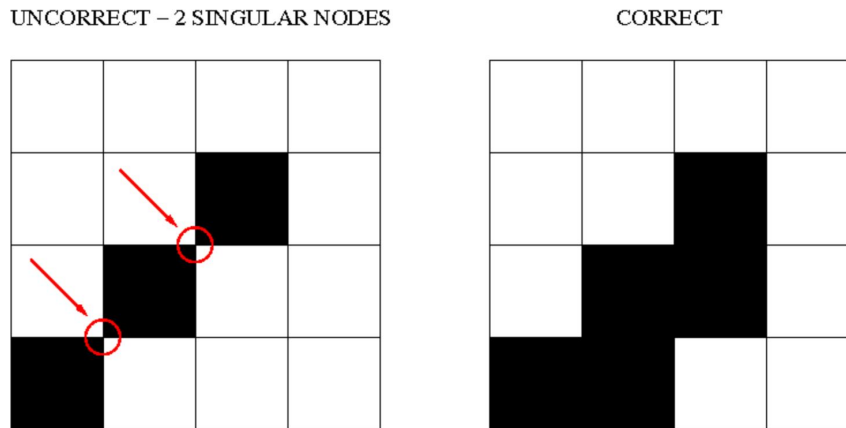
- **domaine** *str*: Name of domain
- **new_bord** *str*: Name of the new boundary
- **bords** *bloc_lecture* (3.60): { Bound1 Bound2 }

3.102 Remove_elem

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

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

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



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

Usage:

remove_elem **domaine** **bloc**

where

- **domaine** *str*: Name of domain
- **bloc** *remove_elem_bloc* ([3.103](#))

3.103 Remove_elem_bloc

Description: `not_set`

See also: [objet_lecture \(39\)](#)

Usage:

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

}

where

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

3.104 Remove_invalid_internal_boundaries

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

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

Usage:

remove_invalid_internal_boundaries domain_name

where

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

3.105 Reorienter_tetraedres

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

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

Usage:

reorienter_tetraedres domain_name

where

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

3.106 Reorienter_triangles

Description: not_set

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

Usage:

reorienter_triangles domain_name

where

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

3.107 Reordonner

Description: The Reordonner_32_64 interpreter is required sometimes for a VDF mesh which is not produced by the internal mesher. Example where this is used:

Read_file dom fichier.geom

Reordonner_32_64 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_32_64 (Resequencing) keyword is required but not included in the data set...

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

Usage:

reordonner domain_name

where

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

3.108 Residuals

Description: To specify how the residuals will be computed.

See also: interpret (3)

Usage:

residuals {

[**norm** *str* into ['L2', 'max']]
[**relative** *str* into ['0', '1', '2']]

}

where

- **norm** *str* into ['L2', 'max']: allows to choose the norm we want to use (max norm by default). Possible to specify L2-norm.
- **relative** *str* into ['0', '1', '2']: This is the old keyword `seuil_statio_relatif_deconseille`. If it is set to 1, it will normalize the residuals with the residuals of the first 5 timesteps (default is 0). if set to 2, residual will be computed as $R/(\max - \min)$.

3.109 Rotation

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

See also: interpret (3)

Usage:

rotation domain_name dir coord1 coord2 angle

where

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

3.110 Scatter

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

See also: interpret (3) scattermed (3.111)

Usage:

scatter file domaine

where

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

3.111 Scattermed

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

See also: scatter ([3.110](#))

Usage:

scattermed file domaine
where

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

3.112 Solve

Synonymous: **resoudre**

Description: Interpreter to start calculation with TRUST.

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

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

Usage:

solve pb
where

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

3.113 Stat_per_proc_perf_log

Description: Keyword allowing to activate the detailed statistics per processor (by default this is false, and only the master proc will produce stats).

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

Usage:

stat_per_proc_perf_log flg
where

- **flg** *int*: A rien that can be either 0 or 1 to turn off (default) or on the detailed stats.

3.114 Supprime_bord

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

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

Usage:

supprime_bord **domaine** **bords**

where

- **domaine** *str*: Name of domain
- **bords** *list_nom* (3.115): { Boundary_name1 Boundaray_name2 }

3.115 List_nom

Description: List of name.

See also: listobj (38.5)

Usage:

{ object1 object2 }

list of *nom_anonyme* (25.1)

3.116 System

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

See also: interprete (3)

Usage:

system **cmd**

where

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

3.117 Test_solveur

Description: To test several solvers

See also: interprete (3)

Usage:

test_solveur {

[**fichier_secmem** *str*]
[**fichier_matrice** *str*]
[**fichier_solution** *str*]
[**nb_test** *int*]
[**impr**]
[**solveur** *solveur_sys_base*]
[**fichier_solveur** *str*]
[**genre_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* (11.16): To specify a solver
- **fichier_solveur** *str*: To specify a file containing a list of solvers
- **genere_fichier_solveur** *float*: To create a file of the solver with a threshold convergence
- **seuil_verification** *float*: Check if the solution satisfy $\|Ax-B\| < \text{precision}$
- **pas_de_solution_initiale** : Resolution isn't initialized with the solution x
- **ascii** : Ascii files

3.118 Testeur

Description: not_set

See also: interpret (3)

Usage:

testeur data

where

- **data** *bloc_lecture* (3.60)

3.119 Testeur_medcoupling

Description: not_set

See also: interpret (3)

Usage:

testeur_medcoupling pb_name field_name

where

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

3.120 Tetraedriser

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

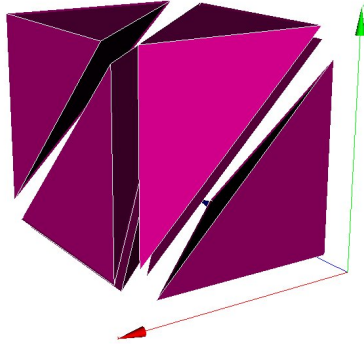
See also: interpret (3) tetraedriser_homogene_fin (3.123) tetraedriser_homogene_compact (3.122) tetraedriser_homogene (3.121) tetraedriser_par_prisme (3.124)

Usage:

tetraedriser domain_name

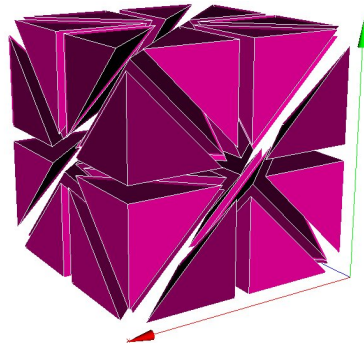
where

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



3.121 Tetraedriser_homogene

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



See also: tetraedriser ([3.120](#))

Usage:

tetraedriser_homogene domain_name

where

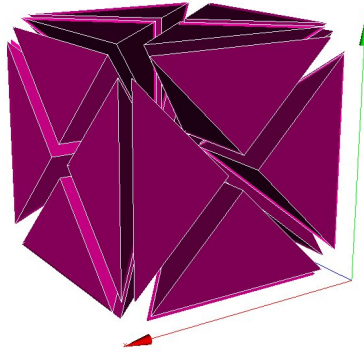
- **domain_name** *str*: Name of domain.

3.122 Tetraedriser_homogene_compact

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

See also: tetraedriser ([3.120](#))

Usage:



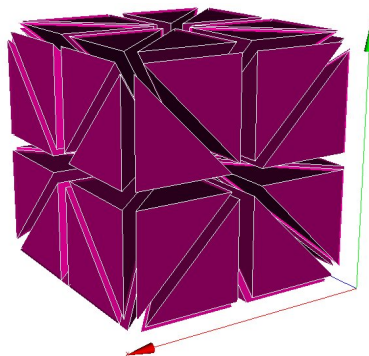
tetraedriser_homogene_compact **domain_name**
where

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

3.123 Tetraedriser_homogene_fin

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

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



See also: tetraedriser ([3.120](#))

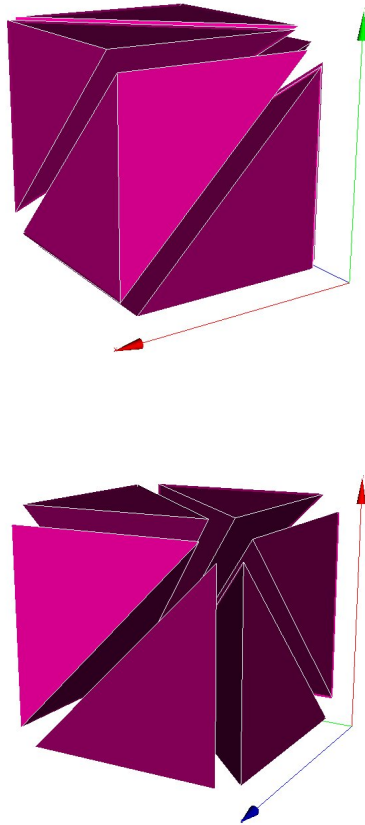
Usage:

tetraedriser_homogene_fin **domain_name**
where

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

3.124 Tetraedriser_par_prisme

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



Initial block is divided in 6 prisms.

See also: tetraedriser ([3.120](#))

Usage:

tetraedriser_par_prisme **domain_name**
where

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

3.125 Transformer

Description: Keyword to transform the coordinates of the geometry.

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

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

Usage:

transformer domain_name formule

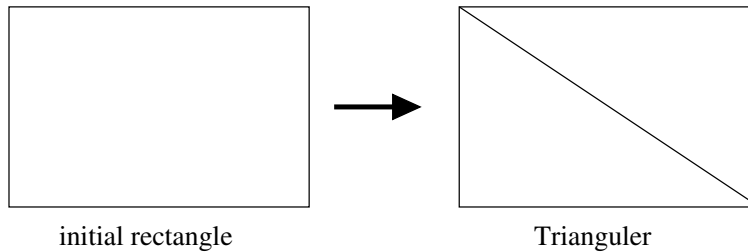
where

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

Function_for_z

3.126 Trianguler

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



See also: interpret (3) [triangler_h \(3.128\)](#) [triangler_fin \(3.127\)](#)

Usage:

triangler domain_name

where

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

3.127 Triangler_fin

Description: Triangler_fin is the recommended option to triangulate rectangles.

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

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

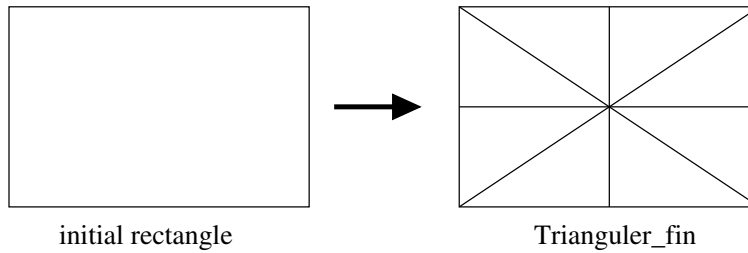
See also: [triangler \(3.126\)](#)

Usage:

triangler_fin domain_name

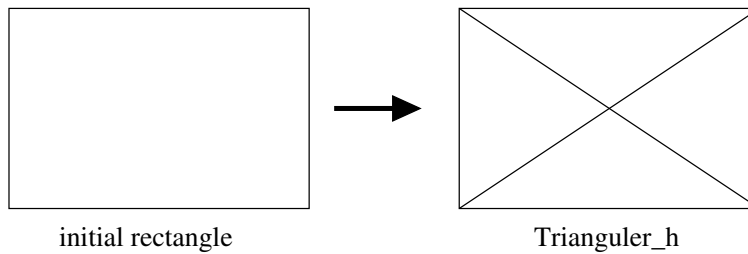
where

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



3.128 Trianguler_h

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



See also: [trianguler \(3.126\)](#)

Usage:

trianguler_h **domain_name**

where

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

3.129 Verifier_qualite_raffinements

Description: not_set

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

Usage:

verifier_qualite_raffinements **domain_names**

where

- **domain_names** *vect_nom* ([3.130](#))

3.130 Vect_nom

Description: Vect of name.

See also: [listobj \(38.5\)](#)

Usage:

n object1 object2
list of *nom_anonyme* (25.1)

3.131 Verifier_simplexes

Description: Keyword to raffine a simplexes

See also: interpret (3)

Usage:

verifier_simplexes **domain_name**
where

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

3.132 Verifiercoin

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

See also: interpret (3)

Usage:

verifiercoin **domain_name** **bloc**
where

- **domain_name** *str*: Name of the domaine
- **bloc** *verifiercoin_bloc* (3.133)

3.133 Verifiercoin_bloc

Description: not_set

See also: objet_lecture (39)

Usage:

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

where

- **Lire_fichier|Read_file** *str*: name of the *.decoupage_som file

3.134 Ecrire

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

See also: interpret (3)

Usage:

ecrire **name_obj**
where

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

3.135 Ecrire_fichier_bin

Synonymous: **ecrire_fichier**

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

See also: [interpret](#) (3) [ecrire_fichier_formatte](#) (3.39)

Usage:

ecrire_fichier_bin **name_obj** **filename**

where

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

4 pb_gen_base

Description: Basic class for problems.

See also: [objet_u](#) (40) [Pb_base](#) (4.22) [pbc_med](#) (4.53) [probleme_couple](#) (4.23)

Usage:

4.1 Pb_conduction

Description: Resolution of the heat equation.

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

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

Usage:

Pb_Conduction *str*

Read *str* {

```
[ solide solide]  
[ Conduction conduction]  
[ milieu milieu_base]  
[ constituant constituant]  
[ Post_processing|postraitement corps_postraitement]  
[ Post_processings|postraitements post_processings]  
[ liste_de_postraitements liste_post_ok]  
[ liste_postraitements liste_post]  
[ sauvegarde format_file_base]  
[ sauvegarde_simple format_file_base]  
[ reprise format_file_base]  
[ resume_last_time format_file_base]
```

}

where

- **solide** *solide* (22.13): The medium associated with the problem.

- **Conduction** *conduction* (5.1): Heat equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.2 Corps_postraitement

Description: not_set

See also: post_processing (4.4.3)

Usage:

```
{
  [ t_debut_statistiques float]
  [ nb_pas_dt_post_stats_plans float]
  [ nb_pas_dt_post_stats_bulles float]
  [ expression_vx_ana str]
  [ expression_vy_ana str]
  [ expression_vz_ana str]
  [ expression_p_ana str]
  [ interfaces interface_posts]
  [ fichier str]
  [ format str into ['lml', 'lata', 'single_lata', 'lata_v2', 'med', 'cgns']]
  [ dt_post str]
  [ nb_pas_dt_post int]
  [ domaine str]
  [ sous_zone|sous_domaine str]
  [ parallele str into ['simple', 'multiple', 'mpi-io']]
}
```

```

[ definition_champs definition_champs]
[ definition_champs_file|definition_champs_fichier definition_champs_fichier]
[ probes|sondes sondes]
[ probes_file|sondes_fichier sondes_fichier]
[ mobile_probes|sondes mobiles sondes]
[ mobile_probes_file|sondes mobiles_fichier sondes_fichier]
[ deprecatedkeepduplicatedprobes int]
[ fields|champs champs_posts]
[ fields_file|champs_fichier champs_posts_fichier]
[ statistics|statistiques stats_posts]
[ statistics_file|statistiques_fichier stats_posts_fichier]
[ serial_statistics|statistiques_en_serie stats_serie_posts]
[ serial_statistics_file|statistiques_en_serie_fichier stats_serie_posts_fichier]
[ suffix_for_reset str]
}
where

```

- **t_debut_statistiques** *float* for inheritance: not_set (for IJK)
- **nb_pas_dt_post_stats_plans** *float* for inheritance: not_set (for IJK)
- **nb_pas_dt_post_stats_bulles** *float* for inheritance: not_set (for IJK)
- **expression_vx_ana** *str* for inheritance: not_set (for IJK)
- **expression_vy_ana** *str* for inheritance: not_set (for IJK)
- **expression_vz_ana** *str* for inheritance: not_set (for IJK)
- **expression_p_ana** *str* for inheritance: not_set (for IJK)
- **interfaces** *interface_posts* (4.2.1) for inheritance: Keyword to read all the characteristics of the interfaces. Different kind of interfaces exist as well as different interface initialisations.
- **fichier** *str* for inheritance: Name of file.
- **format** *str into* ['lml', 'lata', 'single_lata', 'lata_v2', 'med', 'cgns'] for inheritance: This optional parameter specifies the format of the output file. The basename used for the output file is the basename of the data file. For the fmt parameter, choices are lml or lata. A short description of each format can be found below. The default value is lml.
- **dt_post** *str* for inheritance: Field's write frequency (as a time period) - can also be specified after the 'field' keyword.
- **nb_pas_dt_post** *int* for inheritance: Field's write frequency (as a number of time steps) - can also be specified after the 'field' keyword.
- **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).
- **sous_zonelsous_domaine** *str* for inheritance: This optional parameter specifies the sub_domaine on which the data should be interpolated before it is written in the output file. It is only available for sequential computation.
- **parallele** *str into* ['simple', 'multiple', 'mpi-io'] for inheritance: Select simple (single file, sequential write), multiple (several files, parallel write), or mpi-io (single file, parallel write) for LATA format
- **definition_champs** *definition_champs* (4.2.4) for inheritance: Keyword to create new or more complex field for advanced postprocessing.
- **definition_champs_file|definition_champs_fichier** *definition_champs_fichier* (4.2.6) for inheritance: Definition_champs read from file.
- **probes|sondes** *sondes* (4.2.7) for inheritance: Probe.
- **probes_file|sondes_fichier** *sondes_fichier* (4.2.25) for inheritance: Probe read from a file.
- **mobile_probes|sondes mobiles** *sondes* (4.2.7) for inheritance: Mobile probes useful for ALE, their positions will be updated in the mesh.
- **mobile_probes_file|sondes mobiles_fichier** *sondes_fichier* (4.2.25) for inheritance: Mobile probes read in a file

- **deprecatedkeepduplicatedprobes** *int* for inheritance: Flag to not remove duplicated probes in .son files (1: keep duplicate probes, 0: remove duplicate probes)
- **fields|champs** *champs_posts* (4.2.26) for inheritance: Field's write mode.
- **fields_file|champs_fichier** *champs_posts_fichier* (4.2.27) for inheritance: Fields read from file.
- **statistics|statistiques** *stats_posts* (4.2.29) for inheritance: Statistics between two points fixed : start of integration time and end of integration time.
- **statistics_file|statistiques_fichier** *stats_posts_fichier* (4.2.37) for inheritance: Statistics read from file.
- **serial_statistics|statistiques_en_serie** *stats_serie_posts* (4.2.38) for inheritance: Statistics between two points not fixed : on period of integration.
- **serial_statistics_file|statistiques_en_serie_fichier** *stats_serie_posts_fichier* (4.2.39) for inheritance: Serial_statistics read from a file
- **suffix_for_reset** *str* for inheritance: Suffix used to modify the postprocessing file name if the ICoCo resetTime() method is invoked.

4.2.1 Interface_posts

Description: not set

See also: objet_lecture (39)

Usage:

[**nom_interf**] **blocs**

where

- **nom_interf** *str*: name of the interface to post process
- **blocs** *champs_a_post* (4.2.2): Post-processed fields.

4.2.2 Champs_a_post

Description: Fields to be post-processed.

See also: listobj (38.5)

Usage:

{ object1 object2 }

list of *champ_a_post* (4.2.3)

4.2.3 Champ_a_post

Description: Field to be post-processed.

See also: objet_lecture (39)

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

4.2.4 Definition_champs

Description: List of definition champ

See also: listobj (38.5)

Usage:

{ object1 object2 }

list of *definition_champ* (4.2.5)

4.2.5 Definition_champ

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

See also: objet_lecture (39)

Usage:

name champ_generique

where

- **name** *str*: The name of the new created field.
- **champ_generique** *champ_generique_base* (9)

4.2.6 Definition_champs_fichier

Description: Keyword to read definition_champs from a file

See also: objet_lecture (39)

Usage:

{

filefichier *str*

}

where

- **filefichier** *str*: name of file

4.2.7 Sondes

Description: List of probes.

See also: listobj (38.5)

Usage:

{ object1 object2 }

list of *sonde* (4.2.8)

4.2.8 Sonde

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

See also: objet_lecture (39)

Usage:

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

where

- **nom_sonde** *str*: Name of the file in which the values taken over time will be saved. The complete file name is nom_sonde.son.
- **special** *str into* ['grav', 'som', 'nodes', 'chsom', 'gravcl']: Option to change the positions of the probes. Several options are available:
 - grav : each probe is moved to the nearest cell center of the mesh;
 - som : each probe is moved to the nearest vertex of the mesh
 - nodes : each probe is moved to the nearest face center of the mesh;
 - chsom : only available for P1NC sampled field. The values of the probes are calculated according to P1-Conform corresponding field.
 - gravcl : Extend to the domain face boundary a cell-located segment probe in order to have the boundary condition for the field. For this type the extreme probe point has to be on the face center of gravity.
- **nom_inco** *str*: Name of the sampled field.
- **mperiode** *str into* ['periode']: Keyword to set the sampled field measurement frequency.
- **prd** *float*: Period value. Every prd seconds, the field value calculated at the previous time step is written to the nom_sonde.son file.
- **type** *sonde_base* (4.2.9): Type of probe.

4.2.9 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 (39) points (4.2.10) segment (4.2.14) segmentfacesx (4.2.15) segmentfacesy (4.2.16) segmentfacesz (4.2.17) radius (4.2.18) numero_elem_sur_maitre (4.2.19) position_like (4.2.20) plan (4.2.21) volume (4.2.22) circle (4.2.23) circle_3 (4.2.24)

Usage:

sonde_base

4.2.10 Points

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

See also: sonde_base (4.2.9) point (4.2.12) segmentpoints (4.2.13)

Usage:

points points

where

- **points** *listpoints* (4.2.11): Probe points.

4.2.11 Listpoints

Description: Points.

See also: `listobj` ([38.5](#))

Usage:

`n object1 object2`

list of `un_point` ([3.25.3](#))

4.2.12 Point

Description: Point as class-daughter of Points.

See also: `points` ([4.2.10](#))

Usage:

point points

where

- **points** *listpoints* ([4.2.11](#)): Probe points.

4.2.13 Segmentpoints

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

See also: `points` ([4.2.10](#))

Usage:

segmentpoints points

where

- **points** *listpoints* ([4.2.11](#)): Probe points.

4.2.14 Segment

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

See also: `sonde_base` ([4.2.9](#))

Usage:

segment nbr point_deb point_fin

where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* ([3.25.3](#)): First outer probe segment point.
- **point_fin** *un_point* ([3.25.3](#)): Second outer probe segment point.

4.2.15 Segmentfacesx

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

See also: `sonde_base` ([4.2.9](#))

Usage:

segmentfacesx nbr point_deb point_fin

where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* (3.25.3): First outer probe segment point.
- **point_fin** *un_point* (3.25.3): Second outer probe segment point.

4.2.16 Segmentfacesy

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

See also: sonde_base (4.2.9)

Usage:

segmentfacesy nbr point_deb point_fin
where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* (3.25.3): First outer probe segment point.
- **point_fin** *un_point* (3.25.3): Second outer probe segment point.

4.2.17 Segmentfacesz

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

See also: sonde_base (4.2.9)

Usage:

segmentfacesz nbr point_deb point_fin
where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* (3.25.3): First outer probe segment point.
- **point_fin** *un_point* (3.25.3): Second outer probe segment point.

4.2.18 Radius

Description: not_set

See also: sonde_base (4.2.9)

Usage:

radius nbr point_deb radius teta1 teta2
where

- **nbr** *int*: Number of probe points of the segment, evenly distributed.
- **point_deb** *un_point* (3.25.3): First outer probe segment point.
- **radius** *float*
- **teta1** *float*
- **teta2** *float*

4.2.19 Numero_elem_sur_maitre

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

See also: sonde_base ([4.2.9](#))

Usage:

numero_elem_sur_maitre **numero**
where

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

4.2.20 Position_like

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

See also: sonde_base ([4.2.9](#))

Usage:

position_like **autre_sonde**
where

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

4.2.21 Plan

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

See also: sonde_base ([4.2.9](#))

Usage:

plan **nbr** **nbr2** **point_deb** **point_fin** **point_fin_2**
where

- **nbr** *int*: Number of probes in the first direction.
- **nbr2** *int*: Number of probes in the second direction.
- **point_deb** *un_point* ([3.25.3](#)): First point defining the angle. This angle should be positive.
- **point_fin** *un_point* ([3.25.3](#)): Second point defining the angle. This angle should be positive.
- **point_fin_2** *un_point* ([3.25.3](#)): Third point defining the angle. This angle should be positive.

4.2.22 Volume

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

See also: sonde_base ([4.2.9](#))

Usage:

volume **nbr** **nbr2** **nbr3** **point_deb** **point_fin** **point_fin_2** **point_fin_3**
where

- **nbr** *int*: Number of probes in the first direction.
- **nbr2** *int*: Number of probes in the second direction.

- **nbr3** *int*: Number of probes in the third direction.
- **point_deb** *un_point* (3.25.3): Point of origin.
- **point_fin** *un_point* (3.25.3): Point defining the first direction (from point of origin).
- **point_fin_2** *un_point* (3.25.3): Point defining the second direction (from point of origin).
- **point_fin_3** *un_point* (3.25.3): Point defining the third direction (from point of origin).

4.2.23 Circle

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

See also: sonde_base (4.2.9)

Usage:

circle **nbr** **point_deb** [**direction**] **radius** **theta1** **theta2**

where

- **nbr** *int*: Number of probes between teta1 and teta2 (angles given in degrees).
- **point_deb** *un_point* (3.25.3): Center of the circle.
- **direction** *int into* [0, 1, 2]: Axis normal to the circle plane (0:x axis, 1:y axis, 2:z axis).
- **radius** *float*: Radius of the circle.
- **theta1** *float*: First angle.
- **theta2** *float*: Second angle.

4.2.24 Circle_3

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

See also: sonde_base (4.2.9)

Usage:

circle_3 **nbr** **point_deb** **direction** **radius** **theta1** **theta2**

where

- **nbr** *int*: Number of probes between teta1 and teta2 (angles given in degrees).
- **point_deb** *un_point* (3.25.3): Center of the circle.
- **direction** *int into* [0, 1, 2]: Axis normal to the circle plane (0:x axis, 1:y axis, 2:z axis).
- **radius** *float*: Radius of the circle.
- **theta1** *float*: First angle.
- **theta2** *float*: Second angle.

4.2.25 Sondes_fichier

Description: Keyword to read probes from a file

See also: objet_lecture (39)

Usage:

```
{
    filefichier str
}
```

where

- **filefichier** *str*: name of file

4.2.26 Champs_posts

Description: Field's write mode.

See also: [objet_lecture \(39\)](#)

Usage:

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

where

- **format** *str* into ['binaire', 'formatte']: Type of file.
- **mot** *str* into ['dt_post', 'nb_pas_dt_post']: Keyword to set the kind of the field's write frequency. Either a time period or a time step period. it can be specified either here, or at the begining of the postprocessing bloc.
- **period** *str*: Value of the period which can be like (2.*t).
- **fields|champs** *champs_a_post* ([4.2.2](#)): Post-processed fields.

4.2.27 Champs_posts_fichier

Description: Fields read from file.

See also: [objet_lecture \(39\)](#)

Usage:

[**format**] [**mot**] [**period**] **fichier**

where

- **format** *str* into ['binaire', 'formatte']: Type of file.
- **mot** *str* into ['dt_post', 'nb_pas_dt_post']: Keyword to set the kind of the field's write frequency. Either a time period or a time step period.
- **period** *str*: Value of the period which can be like (2.*t).
- **fichier** *bloc_fichier* ([4.2.28](#)): name of file

4.2.28 Bloc_fichier

Description: Block containing the name of the file

See also: [objet_lecture \(39\)](#)

Usage:

{

fichier *str*

}

where

- **fichier** *str*: File name

4.2.29 Stats_posts

Description: Post-processing for statistics.

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:

$t \leq t_{\text{deb}}$ or $t \geq t_{\text{fin}}$:

average: $\overline{P(t)} = 0$

std_deviation: $\langle P(t) \rangle = 0$

correlation: $\langle U(t).V(t) \rangle = 0$

$t > t_{\text{deb}}$ and $t < t_{\text{fin}}$:

average: $\overline{P(t)} = \frac{1}{t-t_{\text{deb}}} \int_{t_{\text{deb}}}^t P(s) ds$

std_deviation: $\langle P(t) \rangle = \sqrt{\frac{1}{t-t_{\text{deb}}} \int_{t_{\text{deb}}}^t [P(s) - \overline{P(t)}]^2 ds}$

correlation: $\langle U(t).V(t) \rangle = \frac{1}{t-t_{\text{deb}}} \int_{t_{\text{deb}}}^t [U(s) - \overline{U(t)}] \cdot [V(s) - \overline{V(t)}] ds$

See also: [objet_lecture \(39\)](#)

Usage:

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

where

- **mot** *str* into ['dt_post', 'nb_pas_dt_post']: Keyword to set the kind of the field's write frequency. Either a time period or a time step period.
- **period** *str*: Value of the period which can be like (2.*t).
- **fields|champs** *list_stat_post* ([4.2.30](#)): Post-processed fields.

4.2.30 List_stat_post

Description: Post-processing for statistics

See also: [listobj \(38.5\)](#)

Usage:

{ object1 object2 }

list of *stat_post_deriv* ([4.2.31](#))

4.2.31 Stat_post_deriv

Description: not_set

See also: [objet_lecture \(39\)](#) [t_deb \(4.2.32\)](#) [t_fin \(4.2.33\)](#) [moyenne \(4.2.34\)](#) [ecart_type \(4.2.35\)](#) [correlation \(4.2.36\)](#)

Usage:

stat_post_deriv

4.2.32 T_deb

Description: Start of integration time

See also: `stat_post_deriv` ([4.2.31](#))

Usage:

t_deb **val**

where

- **val** *float*

4.2.33 T_fin

Description: End of integration time

See also: `stat_post_deriv` ([4.2.31](#))

Usage:

t_fin **val**

where

- **val** *float*

4.2.34 Moyenne

Synonymous: **champ_post_statistiques_moyenne**

Description: to calculate the average of the field over time

See also: `stat_post_deriv` ([4.2.31](#))

Usage:

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

where

- **field** *str*: name of the field on which statistical analysis will be performed. Possible keywords are Vitesse (velocity), Pression (pressure), Temperature, Concentration, ...
- **localisation** *str* into [*'elem'*, *'som'*, *'faces'*]: Localisation of post-processed field value

4.2.35 Ecart_type

Synonymous: **champ_post_statistiques_ecart_type**

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

See also: `stat_post_deriv` ([4.2.31](#))

Usage:

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

where

- **field** *str*: name of the field on which statistical analysis will be performed. Possible keywords are Vitesse (velocity), Pression (pressure), Temperature, Concentration, ...
- **localisation** *str* into [*'elem'*, *'som'*, *'faces'*]: Localisation of post-processed field value

4.2.36 Correlation

Synonymous: **champ_post_statistiques_correlation**

Description: correlation between the two fields

See also: stat_post_deriv (4.2.31)

Usage:

correlation first_field second_field [localisation]

where

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

4.2.37 Stats_posts_fichier

Description: Statistics read from file..

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:

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

See also: objet_lecture (39)

Usage:

mot period fichier

where

- **mot** *str* into [*'dt_post'*, *'nb_pas_dt_post'*]: Keyword to set the kind of the field's write frequency. Either a time period or a time step period.
- **period** *str*: Value of the period which can be like (2.*t).
- **fichier** *bloc_fichier* (4.2.28): name of file

4.2.38 Stats_serie_posts

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

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 (39)

Usage:

mot dt_integr stat
where

- **mot** *str into* [*'dt_integr'*]: Keyword is used to set the statistics period of integration and write period.
- **dt_integr** *float*: Average on dt_integr time interval is post-processed every dt_integr seconds.
- **stat** *list_stat_post* (4.2.30)

4.2.39 Stats_serie_posts_fichier

Description: This keyword is used to set the statistics read from a file. Average on dt_integr time interval is post-processed every dt_integr seconds.

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 (39)

Usage:

mot dt_integr fichier
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.
- **fichier** *bloc_fichier* (4.2.28): name of file

4.3 Post_processings

Synonymous: **postraitements**

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

See also: listobj ([38.5](#))

Usage:

{ object1 object2 }

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

4.3.1 Un_postraitement

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

See also: objet_lecture ([39](#))

Usage:

nom post

where

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

4.4 Liste_post_ok

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

See also: listobj ([38.5](#))

Usage:

{ object1 object2 }

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

4.4.1 Nom_postraitement

Description: not_set

See also: objet_lecture ([39](#))

Usage:

nom post

where

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

4.4.2 Postraitement_base

Description: not_set

See also: objet_lecture ([39](#)) post_processing ([4.4.3](#))

Usage:

4.4.3 Post_processing

Synonymous: **postraitement**

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

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

Usage:

```
post_processing {  
    [ t_debut_statistiques float]  
    [ nb_pas_dt_post_stats_plans float]  
    [ nb_pas_dt_post_stats_bulles float]  
    [ expression_vx_ana str]  
    [ expression_vy_ana str]  
    [ expression_vz_ana str]  
    [ expression_p_ana str]  
    [ interfaces interface_posts]  
    [ fichier str]  
    [ format str into ['lml', 'lata', 'single_lata', 'lata_v2', 'med', 'cgns']]  
    [ dt_post str]  
    [ nb_pas_dt_post int]  
    [ domaine str]  
    [ sous_zone|sous_domaine str]  
    [ parallele str into ['simple', 'multiple', 'mpi-io']]  
    [ definition_champs definition_champs]  
    [ definition_champs_file|definition_champs_fichier definition_champs_fichier]  
    [ probes|sondes sondes]  
    [ probes_file|sondes_fichier sondes_fichier]  
    [ mobile_probes|sondes mobiles sondes]  
    [ mobile_probes_file|sondes mobiles_fichier sondes_fichier]  
    [ deprecatedkeepduplicatedprobes int]  
    [ fields|champs champs_posts]  
    [ fields_file|champs_fichier champs_posts_fichier]  
    [ statistics|statistiques stats_posts]  
    [ statistics_file|statistiques_fichier stats_posts_fichier]  
    [ serial_statistics|statistiques_en_serie stats_serie_posts]  
    [ serial_statistics_file|statistiques_en_serie_fichier stats_serie_posts_fichier]  
    [ suffix_for_reset str]  
}
```

where

- **t_debut_statistiques** *float*: not_set (for IJK)
- **nb_pas_dt_post_stats_plans** *float*: not_set (for IJK)
- **nb_pas_dt_post_stats_bulles** *float*: not_set (for IJK)
- **expression_vx_ana** *str*: not_set (for IJK)
- **expression_vy_ana** *str*: not_set (for IJK)
- **expression_vz_ana** *str*: not_set (for IJK)
- **expression_p_ana** *str*: not_set (for IJK)
- **interfaces** *interface_posts* (4.2.1): Keyword to read all the characteristics of the interfaces. Different kind of interfaces exist as well as different interface initialisations.
- **fichier** *str*: Name of file.
- **format** *str* into ['lml', 'lata', 'single_lata', 'lata_v2', 'med', 'cgns']: This optional parameter specifies the format of the output file. The basename used for the output file is the basename of the data

file. For the `fmt` parameter, choices are `lml` or `lata`. A short description of each format can be found below. The default value is `lml`.

- **dt_post** *str*: Field's write frequency (as a time period) - can also be specified after the 'field' keyword.
- **nb_pas_dt_post** *int*: Field's write frequency (as a number of time steps) - can also be specified after the 'field' keyword.
- **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).
- **sous_zonelsous_domaine** *str*: This optional parameter specifies the sub_domain on which the data should be interpolated before it is written in the output file. It is only available for sequential computation.
- **parallele** *str* into [`'simple'`, `'multiple'`, `'mpi-io'`]: Select simple (single file, sequential write), multiple (several files, parallel write), or mpi-io (single file, parallel write) for LATA format
- **definition_champs** *definition_champs* (4.2.4): Keyword to create new or more complex field for advanced postprocessing.
- **definition_champs_filedefinition_champs_fichier** *definition_champs_fichier* (4.2.6): Definition_champs read from file.
- **probesondes** *sondes* (4.2.7): Probe.
- **probes_filesondes_fichier** *sondes_fichier* (4.2.25): Probe read from a file.
- **mobile_probesondes_mobiles** *sondes* (4.2.7): Mobile probes useful for ALE, their positions will be updated in the mesh.
- **mobile_probes_filesondes_mobiles_fichier** *sondes_fichier* (4.2.25): Mobile probes read in a file
- **deprecatedkeepduplicatedprobes** *int*: Flag to not remove duplicated probes in .son files (1: keep duplicate probes, 0: remove duplicate probes)
- **fieldslchamps** *champs_posts* (4.2.26): Field's write mode.
- **fields_filelchamps_fichier** *champs_posts_fichier* (4.2.27): Fields read from file.
- **statisticslstatistiques** *stats_posts* (4.2.29): Statistics between two points fixed : start of integration time and end of integration time.
- **statistics_filelstatistiques_fichier** *stats_posts_fichier* (4.2.37): Statistics read from file.
- **serial_statisticslstatistiques_en_serie** *stats_serie_posts* (4.2.38): Statistics between two points not fixed : on period of integration.
- **serial_statistics_filelstatistiques_en_serie_fichier** *stats_serie_posts_fichier* (4.2.39): Serial_statistics read from a file
- **suffix_for_reset** *str*: Suffix used to modify the postprocessing file name if the ICoCo `resetTime()` method is invoked.

4.5 Liste_post

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

See also: `listobj` (38.5)

Usage:

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

list of `un_postraitement_spec` (4.5.1)

4.5.1 Un_postraitement_spec

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

See also: `objet_lecture` (39)

Usage:

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

where

- **type_un_post** *type_un_post* (4.5.2)
- **type_postraitement_ft_lata** *type_postraitement_ft_lata* (4.5.3)

4.5.2 Type_un_post

Description: not_set

See also: objet_lecture (39)

Usage:

type post

where

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

4.5.3 Type_postraitement_ft_lata

Description: not_set

See also: objet_lecture (39)

Usage:

type nom bloc

where

- **type** *str* into ['postraitement_ft_lata']
- **nom** *str*: Name of the post-processing.
- **bloc** *bloc_lecture* (3.60)

4.6 Format_file_base

Description: Format of the file

See also: objet_lecture (39) binaire (4.6.1) formatte (4.6.2) xyz (4.6.3) single_hdf (4.6.4) pdi (4.6.5) pdi-expert (4.6.6)

Usage:

checkpoint_fname

where

- **checkpoint_fname** *str*: Name of file.

4.6.1 Binaire

Description: Format of the file - binary version

See also: (4.6)

Usage:

binaire checkpoint_fname

where

- **checkpoint_fname** *str*: Name of file.

4.6.2 Formatte

Description: Format of the file - formatte version

See also: (4.6)

Usage:

formatte checkpoint_fname

where

- **checkpoint_fname** *str*: Name of file.

4.6.3 Xyz

Description: Format of the file - xyz version

See also: (4.6)

Usage:

xyz checkpoint_fname

where

- **checkpoint_fname** *str*: Name of file.

4.6.4 Single_hdf

Description: Format of the file - single_hdf version

See also: (4.6)

Usage:

single_hdf checkpoint_fname

where

- **checkpoint_fname** *str*: Name of file.

4.6.5 Pdi

Description: Format of the file - pdi version

See also: (4.6)

Usage:

pdi checkpoint_fname

where

- **checkpoint_fname** *str*: Name of file.

4.6.6 Pdi_expert

Description: Format of the file - PDI expert version

See also: (4.6)

Usage:

```
pdi_expert {  
    yaml_fname str  
    checkpoint_fname str  
}  
where
```

- **yaml_fname** *str*: YAML file name
- **checkpoint_fname** *str* for inheritance: Name of file.

4.7 Pb_conduction_ibm

Description: Resolution of the IBM heat equation.

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

See also: Pb_base (4.22)

Usage:

```
Pb_Conduction_ibm str  
Read str {  
    [ solide solide]  
    [ Conduction_ibm conduction_ibm]  
    [ milieu milieu_base]  
    [ constituant constituant]  
    [ Post_processing|postraitement corps_postraitement]  
    [ Post_processings|postraitements post_processings]  
    [ liste_de_postraitements liste_post_ok]  
    [ liste_postraitements liste_post]  
    [ sauvegarde format_file_base]  
    [ sauvegarde_simple format_file_base]  
    [ reprise format_file_base]  
    [ resume_last_time format_file_base]  
}  
where
```

- **solide** *solide* (22.13): The medium associated with the problem.
- **Conduction_ibm** *conduction_ibm* (5.8): IBM Heat equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This

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

4.8 Pb_hydraulique_cloned_concentration

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

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

See also: Pb_base (4.22)

Usage:

Pb_Hydraulique_Cloned_Concentration *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_standard navier_stokes_standard ]
    [ convection_diffusion_concentration convection_diffusion_concentration ]
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]

```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_standard** *navier_stokes_standard* (5.41): Navier-Stokes equations.

- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.21): Constituent transport vectorial equation (concentration diffusion convection).
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.9 Pb_hydraulique_cloned_concentration_turbulent

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

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

See also: Pb_base (4.22)

Usage:

Pb_Hydraulique_Cloned_Concentration_Turbulent *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    [ constituant constituant]
    [ navier_stokes_turbulent navier_stokes_turbulent]
    [ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent]
    [ milieu milieu_base]
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processing|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file_base]
    [ sauvegarde_simple format_file_base]

```

```
[ reprise format_file_base]
[ resume_last_time format_file_base]
}
```

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.42): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (5.22): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitements** *corps_postraitements* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitements objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.10 Pb_hydraulique_ibm_turbulent

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

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

See also: Pb_base (4.22)

Usage:

Pb_Hydraulique_IBM_Turbulent *str*

Read *str* {

fluide_incompressible *fluide_incompressible*

```

navier_stokes_ibm_turbulent navier_stokes_ibm_turbulent
[ milieu milieu_base ]
[ constituant constituant ]
[ Post_processing|postraitement corps_postraitement ]
[ Post_processings|postraitements post_processings ]
[ liste_de_postraitements liste_post_ok ]
[ liste_postraitements liste_post ]
[ sauvegarde format_file_base ]
[ sauvegarde_simple format_file_base ]
[ reprise format_file_base ]
[ resume_last_time format_file_base ]
}
where

```

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **navier_stokes_ibm_turbulent** *navier_stokes_ibm_turbulent* (5.39): IBM Navier-Stokes equations as well as the associated turbulence model equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.11 Pb_hydraulique_list_concentration

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

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

See also: `pb_avec_liste_conc` (4.25)

Usage:

Pb_Hydraulique_List_Concentration *str*

Read *str* {

```
    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_standard navier_stokes_standard ]
    list_equations listeqn
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]
```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_standard** *navier_stokes_standard* (5.41): Navier-Stokes equations.
- **list_equations** *listeqn* (4.12) for inheritance: convection_diffusion_concentration equations. The unknown of the concentration equation number N is named concentrationN. 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.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \geq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the

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

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

4.12 Listeqn

Description: List of equations.

See also: listobj (38.5)

Usage:

{ object1 object2 }

list of *eqn_base* (5.32)

4.13 Pb_hydraulique_list_concentration_turbulent

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

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

See also: pb_avec_liste_conc (4.25)

Usage:

Pb_Hydraulique_List_Concentration_Turbulent *str*

Read *str* {

```

fluide_incompressible fluide_incompressible
[ constituant constituant ]
[ navier_stokes_turbulent navier_stokes_turbulent ]
list_equations listeqn
[ milieu milieu_base ]
[ Post_processing|postraitement corps_postraitement ]
[ Post_processings|postraitements post_processings ]
[ liste_de_postraitements liste_post_ok ]
[ liste_postraitements liste_post ]
[ sauvegarde format_file_base ]
[ sauvegarde_simple format_file_base ]
[ reprise format_file_base ]
[ resume_last_time format_file_base ]

```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.42): Navier-Stokes equations as well as the associated turbulence model equations.
- **list_equations** *listeqn* (4.12) for inheritance: convection_diffusion_concentration equations. The unknown of the concentration equation number N is named concentrationN. 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.

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

4.14 Pb_multiphase

Description: A problem that allows the resolution of N-phases with $3*N$ equations

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

See also: Pb_base (4.22) Pb_Multiphase_h (4.15) Pb_HEM (4.16)

Usage:

Pb_Multiphase *str*

Read *str* {

```
[ milieu_composite bloc_lecture]
[ Milieu_MUSIG bloc_lecture]
[ correlations bloc_lecture]
[ models bloc_lecture]
QDM_Multiphase qdm_multiphase
Masse_Multiphase masse_multiphase
Energie_Multiphase energie_multiphase
[ Echelle_temporelle_turbulente echelle_temporelle_turbulente]
[ Energie_cinetique_turbulente energie_cinetique_turbulente]
[ Energie_cinetique_turbulente_WIT energie_cinetique_turbulente_wit]
[ Taux_dissipation_turbulent taux_dissipation_turbulent]
[ milieu milieu_base]
[ constituant constituant]
[ Post_processing|postraitement corps_postraitement]
```



```

[ Post_processings|postraitements post_processings]
[ liste_de_postraitements liste_post_ok]
[ liste_postraitements liste_post]
[ sauvegarde format_file_base]
[ sauvegarde_simple format_file_base]
[ reprise format_file_base]
[ resume_last_time format_file_base]
}
where

```

- **milieu_composite** *bloc_lecture* (3.60): The composite medium associated with the problem.
- **Milieu_MUSIG** *bloc_lecture* (3.60): The composite medium associated with the problem.
- **correlations** *bloc_lecture* (3.60): List of correlations used in specific source terms (i.e. interfacial flux, interfacial friction, ...)
- **models** *bloc_lecture* (3.60): List of models used in specific source terms (i.e. interfacial flux, interfacial friction, ...)
- **QDM_Multiphase** *qdm_multiphase* (5.16): Momentum conservation equation for a multi-phase problem where the unknown is the velocity
- **Masse_Multiphase** *masse_multiphase* (5.15): Mass conservation equation for a multi-phase problem where the unknown is the alpha (void fraction)
- **Energie_Multiphase** *energie_multiphase* (5.11): Internal energy conservation equation for a multi-phase problem where the unknown is the temperature
- **Echelle_temporelle_turbulente** *echelle_temporelle_turbulente* (5.10): Turbulent Dissipation time scale equation for a turbulent mono/multi-phase problem (available in TrioCFD)
- **Energie_cinetique_turbulente** *energie_cinetique_turbulente* (5.13): Turbulent kinetic Energy conservation equation for a turbulent mono/multi-phase problem (available in TrioCFD)
- **Energie_cinetique_turbulente_WIT** *energie_cinetique_turbulente_wit* (5.14): Bubble Induced Turbulent kinetic Energy equation for a turbulent multi-phase problem (available in TrioCFD)
- **Taux_dissipation_turbulent** *taux_dissipation_turbulent* (5.17): Turbulent Dissipation frequency equation for a turbulent mono/multi-phase problem (available in TrioCFD)
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the

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

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

4.15 Pb_multiphase_h

Description: A problem that allows the resolution of N-phases with 3*N equations

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

See also: Pb_Multiphase (4.14)

Usage:

Pb_Multiphase_h *str*

Read *str* {

```
[ milieu_composite bloc_lecture]
[ correlations bloc_lecture]
QDM_Multiphase qdm_multiphase
Masse_Multiphase masse_multiphase
Energie_Multiphase_h energie_multiphase_h
[ Milieu_MUSIG bloc_lecture]
[ models bloc_lecture]
[ Echelle_temporelle_turbulente echelle_temporelle_turbulente]
[ Energie_cinetique_turbulente energie_cinetique_turbulente]
[ Energie_cinetique_turbulente_WIT energie_cinetique_turbulente_wit]
[ Taux_dissipation_turbulent taux_dissipation_turbulent]
[ milieu milieu_base]
[ constituant constituant]
[ Post_processing|postraitements corps_postraitements]
[ Post_processings|postraitements post_processings]
[ liste_de_postraitements liste_post_ok]
[ liste_postraitements liste_post]
[ sauvegarde format_file_base]
[ sauvegarde_simple format_file_base]
[ reprise format_file_base]
[ resume_last_time format_file_base]
```

}

where

- **milieu_composite** *bloc_lecture* (3.60): The composite medium associated with the problem.
- **correlations** *bloc_lecture* (3.60): List of correlations used in specific source terms (i.e. interfacial flux, interfacial friction, ...)
- **QDM_Multiphase** *qdm_multiphase* (5.16): Momentum conservation equation for a multi-phase problem where the unknown is the velocity
- **Masse_Multiphase** *masse_multiphase* (5.15): Mass conservation equation for a multi-phase problem where the unknown is the alpha (void fraction)
- **Energie_Multiphase_h** *energie_multiphase_h* (5.12): Internal energy conservation equation for a multi-phase problem where the unknown is the enthalpy
- **Milieu_MUSIG** *bloc_lecture* (3.60) for inheritance: The composite medium associated with the problem.

- **models** *bloc_lecture* (3.60) for inheritance: List of models used in specific source terms (i.e. interfacial flux, interfacial friction, ...)
- **Echelle_temporelle_turbulente** *echelle_temporelle_turbulente* (5.10) for inheritance: Turbulent Dissipation time scale equation for a turbulent mono/multi-phase problem (available in TrioCFD)
- **Energie_cinetique_turbulente** *energie_cinetique_turbulente* (5.13) for inheritance: Turbulent kinetic Energy conservation equation for a turbulent mono/multi-phase problem (available in TrioCFD)
- **Energie_cinetique_turbulente_WIT** *energie_cinetique_turbulente_wit* (5.14) for inheritance: Bubble Induced Turbulent kinetic Energy equation for a turbulent multi-phase problem (available in TrioCFD)
- **Taux_dissipation_turbulent** *taux_dissipation_turbulent* (5.17) for inheritance: Turbulent Dissipation frequency equation for a turbulent mono/multi-phase problem (available in TrioCFD)
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitemnt** *corps_postraitemnt* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.16 Pb_hem

Description: A problem that allows the resolution of 2-phases mechanically and thermally coupled with 3 equations

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

See also: Pb_Multiphase (4.14)

Usage:

Pb_HEM *str*

Read *str* {

[**milieu_composite** *bloc_lecture*]

```

[ Milieu_MUSIG bloc_lecture]
[ correlations bloc_lecture]
[ models bloc_lecture]
QDM_Multiphase qdm_multiphase
Masse_Multiphase masse_multiphase
Energie_Multiphase energie_multiphase
[ Echelle_temporelle_turbulente echelle_temporelle_turbulente]
[ Energie_cinetique_turbulente energie_cinetique_turbulente]
[ Energie_cinetique_turbulente_WIT energie_cinetique_turbulente_wit]
[ Taux_dissipation_turbulent taux_dissipation_turbulent]
[ milieu milieu_base]
[ constituant constituant]
[ Post_processing|postraitement corps_postraitement]
[ Post_processings|postraitements post_processings]
[ liste_de_postraitements liste_post_ok]
[ liste_postraitements liste_post]
[ sauvegarde format_file_base]
[ sauvegarde_simple format_file_base]
[ reprise format_file_base]
[ resume_last_time format_file_base]
}

```

where

- **milieu_composite** *bloc_lecture* (3.60) for inheritance: The composite medium associated with the problem.
- **Milieu_MUSIG** *bloc_lecture* (3.60) for inheritance: The composite medium associated with the problem.
- **correlations** *bloc_lecture* (3.60) for inheritance: List of correlations used in specific source terms (i.e. interfacial flux, interfacial friction, ...)
- **models** *bloc_lecture* (3.60) for inheritance: List of models used in specific source terms (i.e. interfacial flux, interfacial friction, ...)
- **QDM_Multiphase** *qdm_multiphase* (5.16) for inheritance: Momentum conservation equation for a multi-phase problem where the unknown is the velocity
- **Masse_Multiphase** *masse_multiphase* (5.15) for inheritance: Mass conservation equation for a multi-phase problem where the unknown is the alpha (void fraction)
- **Energie_Multiphase** *energie_multiphase* (5.11) for inheritance: Internal energy conservation equation for a multi-phase problem where the unknown is the temperature
- **Echelle_temporelle_turbulente** *echelle_temporelle_turbulente* (5.10) for inheritance: Turbulent Dissipation time scale equation for a turbulent mono/multi-phase problem (available in TrioCFD)
- **Energie_cinetique_turbulente** *energie_cinetique_turbulente* (5.13) for inheritance: Turbulent kinetic Energy conservation equation for a turbulent mono/multi-phase problem (available in TrioCFD)
- **Energie_cinetique_turbulente_WIT** *energie_cinetique_turbulente_wit* (5.14) for inheritance: Bubble Induced Turbulent kinetic Energy equation for a turbulent multi-phase problem (available in TrioCFD)
- **Taux_dissipation_turbulent** *taux_dissipation_turbulent* (5.17) for inheritance: Turbulent Dissipation frequency equation for a turbulent mono/multi-phase problem (available in TrioCFD)
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This

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

4.17 Pb_thermohydraulique_cloned_concentration

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

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

See also: Pb_base (4.22)

Usage:

Pb_Thermohydraulique_Cloned_Concentration *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_standard navier_stokes_standard ]
    [ convection_diffusion_concentration convection_diffusion_concentration ]
    [ convection_diffusion_temperature convection_diffusion_temperature ]
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]

```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.

- **navier_stokes_standard** *navier_stokes_standard* (5.41): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.21): Constituent transport equations (concentration diffusion convection).
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.28): Energy equation (temperature diffusion convection).
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.18 Pb_thermohydraulique_cloned_concentration_turbulent

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

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

See also: Pb_base (4.22)

Usage:

Pb_Thermohydraulique_Cloned_Concentration_Turbulent *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_turbulent navier_stokes_turbulent ]
    [ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent ]
    [ convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent ]
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]

```



```

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

```

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.42): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (5.22): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (5.31): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.19 Pb_thermohydraulique_ibm_turbulent

Description: Resolution of thermohydraulic problem, with turbulence modelling.

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

See also: `Pb_base` (4.22)

Usage:

Pb_Thermohydraulique_IBM_Turbulent *str*

Read *str* {

```
    fluide_incompressible fluide_incompressible
    navier_stokes_ibm_turbulent navier_stokes_ibm_turbulent
    convection_diffusion_temperature_ibm_turbulent convection_diffusion_temperature_ibm_turbulent
    [ milieu milieu_base ]
    [ constituant constituant ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]
```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **navier_stokes_ibm_turbulent** *navier_stokes_ibm_turbulent* (5.39): IBM Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_temperature_ibm_turbulent** *convection_diffusion_temperature_ibm_turbulent* (5.30): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.

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

4.20 Pb_thermohydraulique_list_concentration

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

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

See also: *pb_avec_liste_conc* (4.25)

Usage:

Pb_Thermohydraulique_List_Concentration *str*

```
Read str {
    fluide_incompressible fluide_incompressible
    [ constituant constituant]
    [ navier_stokes_standard navier_stokes_standard]
    [ convection_diffusion_temperature convection_diffusion_temperature]
    list_equations listeqn
    [ milieu milieu_base]
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file_base]
    [ sauvegarde_simple format_file_base]
    [ reprise format_file_base]
    [ resume_last_time format_file_base]
}
```

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_standard** *navier_stokes_standard* (5.41): Navier-Stokes equations.
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.28): Energy equation (temperature diffusion convection).
- **list_equations** *listeqn* (4.12) for inheritance: *convection_diffusion_concentration* equations. The unknown of the concentration equation number *N* is named *concentrationN*. 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.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is *lata* in order to use *OpenDX* to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory *lata* used in this example should be created before running the computation or the *lata* files will be lost.

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

4.21 Pb_thermohydraulique_list_concentration_turbulent

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

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

See also: pb_avec_liste_conc (4.25)

Usage:

Pb_Thermohydraulique_List_Concentration_Turbulent *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_turbulent navier_stokes_turbulent ]
    [ convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent ]
    list_equations listeqn
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]

```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.42): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (5.31): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.

- **list_equations** *listeqn* (4.12) for inheritance: convection_diffusion_concentration equations. The unknown of the concentration equation number N is named concentrationN. 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.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.22 Pb_base

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

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

See also: pb_gen_base (4) Pb_Conduction (4.1) Pb_Conduction_ibm (4.7) Pb_Thermohydraulique_IBM_Turbulent (4.19) pb_thermohydraulique_ibm (4.48) Pb_Hydraulique_IBM_Turbulent (4.10) pb_hydraulique_ibm (4.32) Pb_Multiphase (4.14) pb_thermohydraulique_concentration_turbulent (4.43) pb_thermohydraulique_turbulent (4.50) pb_avec_liste_conc (4.25) pb_thermohydraulique_turbulent_qc (4.51) pb_hydraulique_turbulent (4.36) Pb_Thermohydraulique_Cloned_Concentration_Turbulent (4.18) Pb_Hydraulique_Cloned_Concentration_Turbulent (4.9) pb_hydraulique_concentration_turbulent (4.30) pb_hydraulique_melange_binaire_turbulent_qc (4.35) pb_avec_passif (4.26) pb_thermohydraulique_QC (4.39) pb_hydraulique_melange_binaire_QC (4.33) pb_thermohydraulique_WC (4.40) pb_hydraulique_melange_binaire_WC (4.34) Pb_Thermohydraulique_Cloned_Concentration (4.17) Pb_Hydraulique_Cloned_Concentration (4.8) pb_thermohydraulique (4.38) pb_hydraulique_concentration (4.28) pb_thermohydraulique_concentration (4.41) pb_hydraulique (4.27) pb_post (4.37) problem_read_generic (4.55)

Usage:

```

Pb_base str
Read str {
    [ milieu milieu_base]
    [ constituant constituant]
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file_base]
    [ sauvegarde_simple format_file_base]
    [ reprise format_file_base]
    [ resume_last_time format_file_base]
}

```

where

- **milieu** *milieu_base* (22): The medium associated with the problem.
- **constituant** *constituant* (22.1): Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2): One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3): List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4): This
- **liste_postraitements** *liste_post* (4.5): This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6): Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6): The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6): Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6): Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.23 Probleme_couple

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

Probleme_Couple pbc

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

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

Each problem is resolved in a domain.

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

See also: pb_gen_base (4)

Usage:

probleme_couple *str*

Read *str* {

 [**groupes** *list_list_nom*]

}

where

- **groupes** *list_list_nom* (4.24): { groupes { { pb1 , pb2 } , { pb3 , pb4 } } }

4.24 List_list_nom

Description: pour les groupes

See also: listobj (38.5)

Usage:

{ object1 , object2 }

list of *list_un_pb* (38.3) separated with ,

4.25 Pb_avec_liste_conc

Description: Class to create a classical problem with a list of scalar concentration equations.

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

See also: Pb_base (4.22) Pb_Thermohydraulique_List_Concentration_Turbulent (4.21) Pb_Hydraulique_List_Concentration_Turbulent (4.13) Pb_Thermohydraulique_List_Concentration (4.20) Pb_Hydraulique_List_Concentration (4.11)

Usage:

pb_avec_liste_conc *str*

Read *str* {

list_equations *listeqn*

 [**milieu** *milieu_base*]

 [**constituant** *constituant*]

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

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

 [**liste_de_postraitements** *liste_post_ok*]

 [**liste_postraitements** *liste_post*]

 [**sauvegarde** *format_file_base*]

 [**sauvegarde_simple** *format_file_base*]

 [**reprise** *format_file_base*]

 [**resume_last_time** *format_file_base*]

}
where

- **list_equations** *listeqn* (4.12): convection_diffusion_concentration equations. The unknown of the concentration equation number N is named concentrationN. 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.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitemnt** *corps_postraitemnt* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.26 Pb_avec_passif

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

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

See also: Pb_base (4.22) pb_thermohydraulique_turbulent_scalaires_passifs (4.52) pb_thermohydraulique_especes_turbulent_qc (4.47) pb_hydraulique_concentration_turbulent_scalaires_passifs (4.31) pb_thermohydraulique_concentration_turbulent_scalaires_passifs (4.44) pb_thermohydraulique_especes_QC (4.45) pb_thermohydraulique_especes_WC (4.46) pb_thermohydraulique_concentration_scalaires_passifs (4.42) pb_thermohydraulique_scalaires_passifs (4.49) pb_hydraulique_concentration_scalaires_passifs (4.29)

Usage:

pb_avec_passif *str*

Read *str* {

equations_scalaires_passifs *listeqn*

```

[ milieu milieu_base]
[ constituant constituant]
[ Post_processing|postraitement corps_postraitement]
[ Post_processings|postraitements post_processings]
[ liste_de_postraitements liste_post_ok]
[ liste_postraitements liste_post]
[ sauvegarde format_file_base]
[ sauvegarde_simple format_file_base]
[ reprise format_file_base]
[ resume_last_time format_file_base]
}
where

```

- **equations_scalaires_passifs** *listeqn* (4.12): Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.27 Pb_hydraulique

Description: Resolution of the Navier-Stokes equations.

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

See also: `Pb_base` (4.22)

Usage:

pb_hydraulique *str*

Read *str* {

```
    fluide_incompressible fluide_incompressible
    navier_stokes_standard navier_stokes_standard
    [ milieu milieu_base ]
    [ constituant constituant ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]
```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **navier_stokes_standard** *navier_stokes_standard* (5.41): Navier-Stokes equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \geq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.28 Pb_hydraulique_concentration

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

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

See also: Pb_base (4.22)

Usage:

pb_hydraulique_concentration *str*

Read *str* {

```
    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_standard navier_stokes_standard ]
    [ convection_diffusion_concentration convection_diffusion_concentration ]
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]
```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_standard** *navier_stokes_standard* (5.41): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.21): Constituent transport vectorial equation (concentration diffusion convection).
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the

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

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

4.29 Pb_hydraulique_concentration_scalaires_passifs

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

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

See also: pb_avec_passif (4.26)

Usage:

pb_hydraulique_concentration_scalaires_passifs *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_standard navier_stokes_standard ]
    [ convection_diffusion_concentration convection_diffusion_concentration ]
    equations_scalaires_passifs listeqn
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]

```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_standard** *navier_stokes_standard* (5.41): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.21): Constituent transport equations (concentration diffusion convection).
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_masseN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This

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

4.30 Pb_hydraulique_concentration_turbulent

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

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

See also: Pb_base (4.22)

Usage:

pb_hydraulique_concentration_turbulent *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_turbulent navier_stokes_turbulent ]
    [ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent ]
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]

```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.

- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.42): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (5.22): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.31 Pb_hydraulique_concentration_turbulent_scalaires_passifs

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

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

See also: pb_avec_passif (4.26)

Usage:

pb_hydraulique_concentration_turbulent_scalaires_passifs *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    [ constituant constituant]
    [ navier_stokes_turbulent navier_stokes_turbulent]
    [ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent]
    equations_scalaires_passifs listeqn
    [ milieu milieu_base]
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]

```

```

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

```

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.42): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (5.22): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_masseN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.32 Pb_hydraulique_ibm

Description: Resolution of the IBM Navier-Stokes equations.

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

See also: Pb_base (4.22)

Usage:

pb_hydraulique_ibm *str*

Read *str* {

```
    fluide_incompressible fluide_incompressible
    navier_stokes_ibm navier_stokes_ibm
    [ milieu milieu_base ]
    [ constituant constituant ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]
```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **navier_stokes_ibm** *navier_stokes_ibm* (5.38): IBM Navier-Stokes equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \geq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.

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

4.33 Pb_hydraulique_melange_binaire_qc

Description: Resolution of a binary mixture problem for a quasi-compressible fluid with an iso-thermal condition.

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

masse_volumique : density

pression : reduced pressure

pression_tot : total pressure.

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

See also: Pb_base (4.22)

Usage:

pb_hydraulique_melange_binaire_QC *str*

Read *str* {

```

    fluide_quasi_compressible fluide_quasi_compressible
    [ constituant constituant]
    navier_stokes_QC navier_stokes_qc
    convection_diffusion_espece_binaire_QC convection_diffusion_espece_binaire_qc
    [ milieu milieu_base]
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file_base]
    [ sauvegarde_simple format_file_base]
    [ reprise format_file_base]
    [ resume_last_time format_file_base]

```

}

where

- **fluide_quasi_compressible** *fluide_quasi_compressible* (22.6): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): The various constituents associated to the problem.
- **navier_stokes_QC** *navier_stokes_qc* (5.33): Navier-Stokes equation for a quasi-compressible fluid.
- **convection_diffusion_espece_binaire_QC** *convection_diffusion_espece_binaire_qc* (5.23): Species conservation equation for a binary quasi-compressible fluid.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.

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

4.34 Pb_hydraulique_melange_binaire_wc

Description: Resolution of a binary mixture problem for a weakly-compressible fluid with an iso-thermal condition.

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

masse_volumique : density

pression : reduced pressure

pression_tot : total pressure

pression_hydro : hydro-static pressure

pression_eos : pressure used in state equation.

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

See also: Pb_base (4.22)

Usage:

pb_hydraulique_melange_binaire_WC *str*

Read *str* {

```

    fluide_weakly_compressible fluide_weakly_compressible
    navier_stokes_WC navier_stokes_wc
    convection_diffusion_espece_binaire_WC convection_diffusion_espece_binaire_wc
    [ milieu milieu_base ]
    [ constituant constituant ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]

```

}

where

- **fluide_weakly_compressible** *fluide_weakly_compressible* (22.12): The fluid medium associated with the problem.

- **navier_stokes_WC** *navier_stokes_wc* (5.37): Navier-Stokes equation for a weakly-compressible fluid.
- **convection_diffusion_espece_binaire_WC** *convection_diffusion_espece_binaire_wc* (5.24): Species conservation equation for a binary weakly-compressible fluid.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.35 Pb_hydraulique_melange_binaire_turbulent_qc

Description: Resolution of a turbulent binary mixture problem for a quasi-compressible fluid with an isothermal condition.

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

See also: Pb_base (4.22)

Usage:

pb_hydraulique_melange_binaire_turbulent_qc *str*

Read *str* {

```

fluide_quasi_compressible fluide_quasi_compressible
navier_stokes_turbulent_qc navier_stokes_turbulent_qc
Convection_Diffusion_Espece_Binaire_Turbulent_QC convection_diffusion_espece_binaire_turbulent-
_qc
[ milieu milieu_base]
[ constituant constituant]
[ Post_processing|postraitement corps_postraitement]
[ Post_processings|postraitements post_processings]

```

```

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

```

- **fluide_quasi_compressible** *fluide_quasi_compressible* (22.6): The fluid medium associated with the problem.
- **navier_stokes_turbulent_qc** *navier_stokes_turbulent_qc* (5.43): Navier-Stokes equation for a quasi-compressible fluid as well as the associated turbulence model equations.
- **Convection_Diffusion_Espece_Binaire_Turbulent_QC** *convection_diffusion_espece_binaire_turbulent_qc* (5.9): Species conservation equation for a quasi-compressible fluid as well as the associated turbulence model equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.36 Pb_hydraulique_turbulent

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

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

See also: Pb_base (4.22)

Usage:

```

pb_hydraulique_turbulent str
Read str {
    fluide_incompressible fluide_incompressible
    navier_stokes_turbulent navier_stokes_turbulent
    [ milieu milieu_base ]
    [ constituant constituant ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]
}

```

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.42): Navier-Stokes equations as well as the associated turbulence model equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \geq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.37 Pb_post

Description: not_set

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

See also: Pb_base (4.22)

Usage:

pb_post *str*

Read *str* {

```
[ milieu milieu_base]  
[ constituant constituant]  
[ Post_processing|postraitement corps_postraitement]  
[ Post_processings|postraitements post_processings]  
[ liste_de_postraitements liste_post_ok]  
[ liste_postraitements liste_post]  
[ sauvegarde format_file_base]  
[ sauvegarde_simple format_file_base]  
[ reprise format_file_base]  
[ resume_last_time format_file_base]
```

}

where

- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.38 Pb_thermohydraulique

Description: Resolution of thermohydraulic problem.

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

See also: `Pb_base` (4.22)

Usage:

pb_thermohydraulique *str*

Read *str* {

```
[ fluide_incompressible fluide_incompressible]  
[ fluide_ostwald fluide_ostwald]  
[ fluide_sodium_liquide fluide_sodium_liquide]  
[ fluide_sodium_gaz fluide_sodium_gaz]  
[ correlations bloc_lecture]  
[ navier_stokes_standard navier_stokes_standard]  
[ convection_diffusion_temperature convection_diffusion_temperature]  
[ milieu milieu_base]  
[ constituant constituant]  
[ Post_processing|postraitement corps_postraitement]  
[ Post_processings|postraitements post_processings]  
[ liste_de_postraitements liste_post_ok]  
[ liste_postraitements liste_post]  
[ sauvegarde format_file_base]  
[ sauvegarde_simple format_file_base]  
[ reprise format_file_base]  
[ resume_last_time format_file_base]
```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem (only one possibility).
- **fluide_ostwald** *fluide_ostwald* (22.5): The fluid medium associated with the problem (only one possibility).
- **fluide_sodium_liquide** *fluide_sodium_liquide* (22.10): The fluid medium associated with the problem (only one possibility).
- **fluide_sodium_gaz** *fluide_sodium_gaz* (22.9): The fluid medium associated with the problem (only one possibility).
- **correlations** *bloc_lecture* (3.60): List of correlations used in specific source terms (i.e. interfacial flux, interfacial friction, ...)
- **navier_stokes_standard** *navier_stokes_standard* (5.41): Navier-Stokes equations.
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.28): Energy equation (temperature diffusion convection).
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified

for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.

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

4.39 Pb_thermohydraulique_qc

Description: Resolution of thermo-hydraulic problem for a quasi-compressible fluid.

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

masse_volumique : density

enthalpie : enthalpy

pression : reduced pressure

pression_tot : total pressure.

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

See also: Pb_base (4.22)

Usage:

pb_thermohydraulique_QC *str*

Read *str* {

```

    fluide_quasi_compressible fluide_quasi_compressible
    navier_stokes_QC navier_stokes_qc
    convection_diffusion_chaleur_QC convection_diffusion_chaleur_qc
    [ milieu milieu_base ]
    [ constituant constituant ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]

```

}

where

- **fluide_quasi_compressible** *fluide_quasi_compressible* (22.6): The fluid medium associated with the problem.
- **navier_stokes_QC** *navier_stokes_qc* (5.33): Navier-Stokes equation for a quasi-compressible fluid.
- **convection_diffusion_chaleur_QC** *convection_diffusion_chaleur_qc* (5.18): Temperature equation for a quasi-compressible fluid.

- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.40 Pb_thermohydraulique_wc

Description: Resolution of thermo-hydraulic problem for a weakly-compressible fluid.

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

masse_volumique : density

pression : reduced pressure

pression_tot : total pressure

pression_hydro : hydro-static pressure

pression_eos : pressure used in state equation.

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

See also: Pb_base (4.22)

Usage:

pb_thermohydraulique_WC *str*

Read *str* {

fluide_weakly_compressible *fluide_weakly_compressible*

navier_stokes_WC *navier_stokes_wc*

convection_diffusion_chaleur_WC *convection_diffusion_chaleur_wc*

[**milieu** *milieu_base*]

[**constituant** *constituant*]

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

[**Post_processing|postraitements** *post_processings*]


```

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

```

- **fluide_weakly_compressible** *fluide_weakly_compressible* (22.12): The fluid medium associated with the problem.
- **navier_stokes_WC** *navier_stokes_wc* (5.37): Navier-Stokes equation for a weakly-compressible fluid.
- **convection_diffusion_chaleur_WC** *convection_diffusion_chaleur_wc* (5.19): Temperature equation for a weakly-compressible fluid.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.41 Pb_thermohydraulique_concentration

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

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

See also: Pb_base (4.22)

Usage:

pb_thermohydraulique_concentration *str*

Read *str* {

```
    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_standard navier_stokes_standard ]
    [ convection_diffusion_concentration convection_diffusion_concentration ]
    [ convection_diffusion_temperature convection_diffusion_temperature ]
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]
```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_standard** *navier_stokes_standard* (5.41): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.21): Constituent transport equations (concentration diffusion convection).
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.28): Energy equation (temperature diffusion convection).
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time

of saved files).

4.42 Pb_thermohydraulique_concentration_scalaires_passifs

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

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

See also: pb_avec_passif (4.26)

Usage:

pb_thermohydraulique_concentration_scalaires_passifs *str*

Read *str* {

```
    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_standard navier_stokes_standard ]
    [ convection_diffusion_concentration convection_diffusion_concentration ]
    [ convection_diffusion_temperature convection_diffusion_temperature ]
    equations_scalaires_passifs listeqn
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]
```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_standard** *navier_stokes_standard* (5.41): Navier-Stokes equations.
- **convection_diffusion_concentration** *convection_diffusion_concentration* (5.21): Constituent transport equations (concentration diffusion convection).
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.28): Energy equations (temperature diffusion convection).
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_masseN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This

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

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

4.43 Pb_thermohydraulique_concentration_turbulent

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

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

See also: Pb_base (4.22)

Usage:

pb_thermohydraulique_concentration_turbulent *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_turbulent navier_stokes_turbulent ]
    [ convection_diffusion_concentration_turbulent convection_diffusion_concentration_turbulent ]
    [ convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent ]
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]

```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.

- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.42): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (5.22): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (5.31): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.44 Pb_thermohydraulique_concentration_turbulent_scalaires_passifs

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

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

See also: pb_avec_passif (4.26)

Usage:

pb_thermohydraulique_concentration_turbulent_scalaires_passifs *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ 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
[ milieu milieu_base ]
[ Post_processing|postraitement corps_postraitement ]
[ Post_processings|postraitements post_processings ]
[ liste_de_postraitements liste_post_ok ]
[ liste_postraitements liste_post ]
[ sauvegarde format_file_base ]
[ sauvegarde_simple format_file_base ]
[ reprise format_file_base ]
[ resume_last_time format_file_base ]
}
where

```

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.42): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_concentration_turbulent** *convection_diffusion_concentration_turbulent* (5.22): Constituent transport equations (concentration diffusion convection) as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (5.31): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_masseN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.

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

4.45 Pb_thermohydraulique_especes_qc

Description: Resolution of thermo-hydraulic problem for a multi-species quasi-compressible fluid.

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

See also: **pb_avec_passif** (4.26)

Usage:

pb_thermohydraulique_especes_QC *str*

Read *str* {

```

    fluide_quasi_compressible fluide_quasi_compressible
    navier_stokes_QC navier_stokes_qc
    convection_diffusion_chaleur_QC convection_diffusion_chaleur_qc
    equations_scalaires_passifs listeqn
    [ milieu milieu_base ]
    [ constituant constituant ]
    [ Post_processing|postraitemment corps_postraitemment ]
    [ Post_processings|postraitemments post_processings ]
    [ liste_de_postraitemments liste_post_ok ]
    [ liste_postraitemments liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]

```

}

where

- **fluide_quasi_compressible** *fluide_quasi_compressible* (22.6): The fluid medium associated with the problem.
- **navier_stokes_QC** *navier_stokes_qc* (5.33): Navier-Stokes equation for a quasi-compressible fluid.
- **convection_diffusion_chaleur_QC** *convection_diffusion_chaleur_qc* (5.18): Temperature equation for a quasi-compressible fluid.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named *temperatureN* or *concentrationN* or *fraction_masseN*. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitemment** *corps_postraitemment* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitemments** *post_processings* (4.3) for inheritance: List of Postraitemment objects (with name).
- **liste_de_postraitemments** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitemments** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is *lata* in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and

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

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

4.46 Pb_thermohydraulique_especes_wc

Description: Resolution of thermo-hydraulic problem for a multi-species weakly-compressible fluid.

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

See also: `pb_avec_passif` (4.26)

Usage:

pb_thermohydraulique_especes_WC *str*

Read *str* {

```

    fluide_weakly_compressible fluide_weakly_compressible
    navier_stokes_WC navier_stokes_wc
    convection_diffusion_chaleur_WC convection_diffusion_chaleur_wc
    equations_scalaires_passifs listeqn
    [ milieu milieu_base]
    [ constituant constituant]
    [ Post_processing|postraitements corps_postraitements]
    [ Post_processing|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file_base]
    [ sauvegarde_simple format_file_base]
    [ reprise format_file_base]
    [ resume_last_time format_file_base]

```

}

where

- **fluide_weakly_compressible** *fluide_weakly_compressible* (22.12): The fluid medium associated with the problem.
- **navier_stokes_WC** *navier_stokes_wc* (5.37): Navier-Stokes equation for a weakly-compressible fluid.
- **convection_diffusion_chaleur_WC** *convection_diffusion_chaleur_wc* (5.19): Temperature equation for a weakly-compressible fluid.

- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction-massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitemnt** *corps_postraitemnt* (4.2) for inheritance: One post-processing (without name).
- **Post_processing|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.47 Pb_thermohydraulique_especes_turbulent_qc

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

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

See also: pb_avec_passif (4.26)

Usage:

pb_thermohydraulique_especes_turbulent_qc *str*

Read *str* {

```

    fluide_quasi_compressible fluide_quasi_compressible
    navier_stokes_turbulent_qc navier_stokes_turbulent_qc
    convection_diffusion_chaleur_turbulent_qc convection_diffusion_chaleur_turbulent_qc
    equations_scalaires_passifs listeqn
    [ milieu milieu_base]
    [ constituant constituant]
    [ Post_processing|postraitemnt corps_postraitemnt]

```



```

[ Post_processings|postraitements post_processings]
[ liste_de_postraitements liste_post_ok]
[ liste_postraitements liste_post]
[ sauvegarde format_file_base]
[ sauvegarde_simple format_file_base]
[ reprise format_file_base]
[ resume_last_time format_file_base]
}
where

```

- **fluide_quasi_compressible** *fluide_quasi_compressible* (22.6): The fluid medium associated with the problem.
- **navier_stokes_turbulent_qc** *navier_stokes_turbulent_qc* (5.43): Navier-Stokes equations under low Mach number as well as the associated turbulence model equations.
- **convection_diffusion_chaleur_turbulent_qc** *convection_diffusion_chaleur_turbulent_qc* (5.20): Energy equation under low Mach number as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N (N<>P) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.48 Pb_thermohydraulique_ibm

Description: Resolution of IBM thermohydraulic problem.

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

See also: Pb_base (4.22)

Usage:

pb_thermohydraulique_ibm *str*

Read *str* {

```
[ fluide_incompressible fluide_incompressible]  
[ fluide_ostwald fluide_ostwald]  
[ navier_stokes_ibm navier_stokes_ibm]  
[ convection_diffusion_temperature_ibm convection_diffusion_temperature_ibm]  
[ milieu milieu_base]  
[ constituant constituant]  
[ Post_processing|postraitement corps_postraitement]  
[ Post_processings|postraitements post_processings]  
[ liste_de_postraitements liste_post_ok]  
[ liste_postraitements liste_post]  
[ sauvegarde format_file_base]  
[ sauvegarde_simple format_file_base]  
[ reprise format_file_base]  
[ resume_last_time format_file_base]
```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem (only one possibility).
- **fluide_ostwald** *fluide_ostwald* (22.5): The fluid medium associated with the problem (only one possibility).
- **navier_stokes_ibm** *navier_stokes_ibm* (5.38): IBM Navier-Stokes equations.
- **convection_diffusion_temperature_ibm** *convection_diffusion_temperature_ibm* (5.29): IBM Energy equation (temperature diffusion convection).
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz

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

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

4.49 Pb_thermohydraulique_scalaires_passifs

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

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

See also: pb_avec_passif (4.26)

Usage:

pb_thermohydraulique_scalaires_passifs *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_standard navier_stokes_standard ]
    [ convection_diffusion_temperature convection_diffusion_temperature ]
    equations_scalaires_passifs listeqn
    [ milieu milieu_base ]
    [ Post_processing|postraitemment corps_postraitemment ]
    [ Post_processings|postraitemments post_processings ]
    [ liste_de_postraitemments liste_post_ok ]
    [ liste_postraitemments liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]

```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_standard** *navier_stokes_standard* (5.41): Navier-Stokes equations.
- **convection_diffusion_temperature** *convection_diffusion_temperature* (5.28): Energy equations (temperature diffusion convection).
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_masseN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitemment** *corps_postraitemment* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitemments** *post_processings* (4.3) for inheritance: List of Postraitemment objects (with name).

- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.50 Pb_thermohydraulique_turbulent

Description: Resolution of thermohydraulic problem, with turbulence modelling.

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

See also: Pb_base (4.22)

Usage:

pb_thermohydraulique_turbulent *str*

Read *str* {

```

    fluide_incompressible fluide_incompressible
    navier_stokes_turbulent navier_stokes_turbulent
    convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent
    [ milieu milieu_base]
    [ constituant constituant]
    [ Post_processing|postraitement corps_postraitement]
    [ Post_processings|postraitements post_processings]
    [ liste_de_postraitements liste_post_ok]
    [ liste_postraitements liste_post]
    [ sauvegarde format_file_base]
    [ sauvegarde_simple format_file_base]
    [ reprise format_file_base]
    [ resume_last_time format_file_base]

```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.

- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.42): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (5.31): Energy equation (temperature diffusion convection) as well as the associated turbulence model equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N < P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.51 Pb_thermohydraulique_turbulent_qc

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

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

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

See also: Pb_base (4.22)

Usage:

pb_thermohydraulique_turbulent_qc *str*

Read *str* {

```

fluide_quasi_compressible fluide_quasi_compressible
navier_stokes_turbulent_qc navier_stokes_turbulent_qc
convection_diffusion_chaleur_turbulent_qc convection_diffusion_chaleur_turbulent_qc
[ milieu milieu_base]
[ constituant constituant]
[ Post_processing|postraitement corps_postraitement]
[ Post_processings|postraitements post_processings]

```

```

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

```

- **fluide_quasi_compressible** *fluide_quasi_compressible* (22.6): The fluid medium associated with the problem.
- **navier_stokes_turbulent_qc** *navier_stokes_turbulent_qc* (5.43): Navier-Stokes equations under low Mach number as well as the associated turbulence model equations.
- **convection_diffusion_chaleur_turbulent_qc** *convection_diffusion_chaleur_turbulent_qc* (5.20): Energy equation under low Mach number as well as the associated turbulence model equations.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

4.52 Pb_thermohydraulique_turbulent_scalaires_passifs

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

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

See also: `pb_avec_passif` (4.26)

Usage:

pb_thermohydraulique_turbulent_scalaires_passifs *str*

Read *str* {

```
    fluide_incompressible fluide_incompressible
    [ constituant constituant ]
    [ navier_stokes_turbulent navier_stokes_turbulent ]
    [ convection_diffusion_temperature_turbulent convection_diffusion_temperature_turbulent ]
    equations_scalaires_passifs listeqn
    [ milieu milieu_base ]
    [ Post_processing|postraitement corps_postraitement ]
    [ Post_processings|postraitements post_processings ]
    [ liste_de_postraitements liste_post_ok ]
    [ liste_postraitements liste_post ]
    [ sauvegarde format_file_base ]
    [ sauvegarde_simple format_file_base ]
    [ reprise format_file_base ]
    [ resume_last_time format_file_base ]
```

}

where

- **fluide_incompressible** *fluide_incompressible* (22.4): The fluid medium associated with the problem.
- **constituant** *constituant* (22.1): Constituents.
- **navier_stokes_turbulent** *navier_stokes_turbulent* (5.42): Navier-Stokes equations as well as the associated turbulence model equations.
- **convection_diffusion_temperature_turbulent** *convection_diffusion_temperature_turbulent* (5.31): Energy equations (temperature diffusion convection) as well as the associated turbulence model equations.
- **equations_scalaires_passifs** *listeqn* (4.12) for inheritance: Passive scalar equations. The unknowns of the passive scalar equation number N are named temperatureN or concentrationN or fraction_massiqueN. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.
- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.

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

4.53 Pbc_med

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

See also: `pb_gen_base` (4)

Usage:

pbc_med list_info_med

where

- **list_info_med** *list_info_med* (4.54)

4.54 List_info_med

Description: `not_set`

See also: `listobj` (38.5)

Usage:

{ `object1` , `object2` }

list of *info_med* (4.54.1) separated with ,

4.54.1 Info_med

Description: `not_set`

See also: `objet_lecture` (39)

Usage:

file_med domaine pb_post

where

- **file_med** *str*: Name of the MED file.
- **domaine** *str*: Name of domain.
- **pb_post** *pb_post* (4.37)

4.55 Problem_read_generic

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

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

See also: Pb_base (4.22)

Usage:

problem_read_generic *str*

Read *str* {

```
[ milieu milieu_base]  
[ constituant constituant]  
[ Post_processing|postraitement corps_postraitement]  
[ Post_processings|postraitements post_processings]  
[ liste_de_postraitements liste_post_ok]  
[ liste_postraitements liste_post]  
[ sauvegarde format_file_base]  
[ sauvegarde_simple format_file_base]  
[ reprise format_file_base]  
[ resume_last_time format_file_base]
```

}

where

- **milieu** *milieu_base* (22) for inheritance: The medium associated with the problem.
- **constituant** *constituant* (22.1) for inheritance: Constituent.
- **Post_processing|postraitement** *corps_postraitement* (4.2) for inheritance: One post-processing (without name).
- **Post_processings|postraitements** *post_processings* (4.3) for inheritance: List of Postraitement objects (with name).
- **liste_de_postraitements** *liste_post_ok* (4.4) for inheritance: This
- **liste_postraitements** *liste_post* (4.5) for inheritance: This block defines the output files to be written during the computation. The output format is lata in order to use OpenDX to draw the results. This block can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention. The directory lata used in this example should be created before running the computation or the lata files will be lost.
- **sauvegarde** *format_file_base* (4.6) for inheritance: Keyword used when calculation results are to be backed up. When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when resuming the calculation.
- **sauvegarde_simple** *format_file_base* (4.6) for inheritance: The same keyword than Sauvegarde except, the last time step only is saved.
- **reprise** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file (see the class format_file). If format_reprise is xyz, the name_file file should be the .xyz file created by the previous calculation. With this file, it is possible to resume a parallel calculation on P processors, whereas the previous calculation has been run on N ($N \leq P$) processors. Should the calculation be resumed, values for the tinit (see schema_temps_base) time fields are taken from the name_file file. If there is no backup corresponding to this time in the name_file, TRUST exits in error.
- **resume_last_time** *format_file_base* (4.6) for inheritance: Keyword to resume a calculation based on the name_file file, resume the calculation at the last time found in the file (tinit is set to last time of saved files).

5 mor_eqn

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

See also: `objet_u` (40) `eqn_base` (5.32)

Usage:

5.1 Conduction

Description: Heat equation.

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

See also: `eqn_base` (5.32) `Conduction_ibm` (5.8)

Usage:

Conduction *str*

Read *str* {

```
[ disable_equation_residual int]  
[ convection bloc_convection]  
[ diffusion bloc_diffusion]  
[ boundary_conditions|conditions_limites condlims]  
[ initial_conditions|conditions_initiales condinits]  
[ sources sources]  
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]  
[ parametre_equation parametre_equation_base]  
[ equation_non_resolue str]  
[ renommer_equation str]
```

}

where

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if `equation_non_resolue` keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ `equation_non_resolue` (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.2 Bloc_convection

Description: `not_set`

See also: `objet_lecture` (39)

Usage:

aco **opérateur** **acof**

where

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

5.2.1 Convection_deriv

Description: not_set

See also: objet_lecture (39) ale (5.2.2) muscl_old (5.2.3) muscl3 (5.2.4) ef (5.2.5) di_l2 (5.2.7) amount_old (5.2.8) generic (5.2.9) ef_stab (5.2.10) kquick (5.2.13) muscl (5.2.14) muscl_new (5.2.15) quick (5.2.16) centre_old (5.2.17) negligible (5.2.18) amount (5.2.19) centre (5.2.20) centre4 (5.2.21) btd (5.2.22) supg (5.2.23)

Usage:

convection_deriv

5.2.2 Ale

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

See also: convection_deriv (5.2.1)

Usage:

ale **opconv**

where

- **opconv** *bloc_convection* (5.2): Choice between: amount and muscl
Example: convection { ALE { amount } }

5.2.3 Muscl_old

Description: Only for VEF discretization.

See also: convection_deriv (5.2.1)

Usage:

muscl_old

5.2.4 Muscl3

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

See also: convection_deriv (5.2.1)

Usage:

muscl3 {

[**alpha** *float*]

}
where

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

5.2.5 Ef

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

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

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

Remark:

This class requires to define a filtering operator : see solveur_bar

See also: convection_deriv (5.2.1)

Usage:

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

where

- **mot1** *str* into ['defaut_bar']: equivalent to transportant_bar 0 transporte_bar 1 filtrer_resu 1 antisym 1
- **bloc_ef** *bloc_ef* (5.2.6)

5.2.6 Bloc_ef

Description: not_set

See also: objet_lecture (39)

Usage:

mot1 val1 mot2 val2 mot3 val3 mot4 val4

where

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

5.2.7 Di_l2

Description: Only for VEF discretization.

See also: convection_deriv (5.2.1)

Usage:

di_l2

5.2.8 Amont_old

Description: Only for VEF discretization, obsolete keyword, see `amont`.

See also: convection_deriv (5.2.1)

Usage:

amont_old

5.2.9 Generic

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

Examples:

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

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

See also: convection_deriv (5.2.1)

Usage:

generic type [limiteur] [ordre] [alpha]

where

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

5.2.10 Ef_stab

Description: Keyword for a VEF convective scheme.

See also: convection_deriv (5.2.1)

Usage:

```
ef_stab {  
    [ alpha float ]
```

```

[ test int]
[ tdivu ]
[ old ]
[ volumes_etendus ]
[ volumes_non_etendus ]
[ amont_sous_zone str]
[ alpha_sous_zone listsous_zone_valeur]
}
where

```

- **alpha** *float*: To weight the scheme centering with the factor floatant (between 0 (full centered) and 1 (mix between upwind and centered), by default 1). For scalar equation, it is advised to use alpha=1 and for the momentum equation, alpha=0.2 is advised.
- **test** *int*: Developer option to compare old and new version of EF_stab
- **tdivu** : To have the convective operator calculated as $\text{div}(\text{TU}) - \text{TdivU} (= \text{UgradT})$.
- **old** : To use old version of EF_stab scheme (default no).
- **volumes_etendus** : Option for the scheme to use the extended volumes (default, yes).
- **volumes_non_etendus** : Option for the scheme to not use the extended volumes (default, no).
- **amont_sous_zone** *str*: Option to degenerate EF_stab scheme into Amont (upwind) scheme in the sub zone of name sz_name. The sub zone may be located arbitrarily in the domain but the more often this option will be activated in a zone where EF_stab scheme generates instabilities as for free outlet for example.
- **alpha_sous_zone** *listsous_zone_valeur* (5.2.11): 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.

5.2.11 Listsous_zone_valeur

Description: List of groups of two words.

See also: listobj (38.5)

Usage:

n object1 object2

list of *sous_zone_valeur* (5.2.12)

5.2.12 Sous_zone_valeur

Description: Two words.

See also: objet_lecture (39)

Usage:

sous_zone valeur

where

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

5.2.13 Kquick

Description: Only for VEF discretization.

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

Usage:

kquick

5.2.14 Muscl

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

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

Usage:

muscl

5.2.15 Muscl_new

Description: Only for VEF discretization.

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

Usage:

muscl_new

5.2.16 Quick

Description: Only for VDF discretization.

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

Usage:

quick

5.2.17 Centre_old

Description: Only for VEF discretization.

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

Usage:

centre_old

5.2.18 Negligeable

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

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

Usage:

negligeable

5.2.19 Amont

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

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

Usage:

amont

5.2.20 Centre

Description: For VDF and VEF discretizations.

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

Usage:

centre

5.2.21 Centre4

Description: For VDF and VEF discretizations.

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

Usage:

centre4

5.2.22 Btd

Description: Only for EF discretization.

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

Usage:

btd {

btd *float*

facteur *float*

}

where

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

5.2.23 Supg

Description: Only for EF discretization.

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

Usage:

supg {

facteur *float*
 }
 where

- **facteur** *float*

5.3 Bloc_diffusion

Description: not_set

See also: objet_lecture (39)

Usage:

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

- **aco** *str* into ['{']: Opening curly bracket.
- **opérateur** *diffusion_deriv* (5.3.1): if none is specified, the diffusive scheme used is a 2nd-order scheme.
- **op_implicite** *op_implicite* (5.3.17): To have diffusive implicitation, it use Uzawa algorithm. Very useful when viscosity has large variations.
- **acof** *str* into ['}']: Closing curly bracket.

5.3.1 Diffusion_deriv

Description: not_set

See also: objet_lecture (39) turbulente (5.3.2) stab (5.3.10) standard (5.3.11) p1ncp1b (5.3.13) p1b (5.3.14) negligible (5.3.15) option (5.3.16)

Usage:

diffusion_deriv

5.3.2 Turbulente

Description: Turbulent diffusion operator for multiphase problem

See also: diffusion_deriv (5.3.1)

Usage:

turbulente [**type**]
 where

- **type** *type_diffusion_turbulente_multiphase_deriv* (5.3.3): Turbulence model for multiphase problem

5.3.3 Type_diffusion_turbulente_multiphase_deriv

Description: not_set

See also: objet_lecture (39) interfacial_area (5.3.4) wale (5.3.5) l_melange (5.3.6) smago (5.3.7) Prandtl (5.3.8) SGDH (5.3.9)

Usage:

5.3.4 Interfacial_area

Synonymous: **aire_interfaciale**

Description: not_set

See also: type_diffusion_turbulente_multiphase_deriv ([5.3.3](#))

Usage:

interfacial_area {

 [**cstdiff** *float*]

 [**ng2** *float*]

}

where

- **cstdiff** *float*: Kataoka diffusion model constant. By default it is set to 0.236.
- **ng2** *float*

5.3.5 Wale

Description: LES WALE type.

See also: type_diffusion_turbulente_multiphase_deriv ([5.3.3](#))

Usage:

wale {

 [**cw** *float*]

}

where

- **cw** *float*: WALE's model constant. By default it is set to 0.5.

5.3.6 L_melange

Description: not_set

See also: type_diffusion_turbulente_multiphase_deriv ([5.3.3](#))

Usage:

l_melange {

l_melange *float*

}

where

- **l_melange** *float*

5.3.7 Smago

Description: LES Smagorinsky type.

See also: `type_diffusion_turbulente_multiphase_deriv` ([5.3.3](#))

Usage:

```
smago {  
    [ cs float ]
```

```
}
```

where

- **cs** *float*: Smagorinsky's model constant. By default it is set to 0.18.

5.3.8 Prandtl

Description: Scalar Prandtl model.

See also: `type_diffusion_turbulente_multiphase_deriv` ([5.3.3](#))

Usage:

```
Prandtl {  
    [ prandtl_turbulent|pr_t float ]
```

```
}
```

where

- **prandtl_turbulent|pr_t** *float*: Prandtl's model constant. By default it is set to 0.9.

5.3.9 Sgdh

Description: `not_set`

See also: `type_diffusion_turbulente_multiphase_deriv` ([5.3.3](#))

Usage:

```
SGDH {  
    [ Pr_t float ]  
    [ sigma_turbulent|sigma float ]  
    [ no_alpha ]  
    [ gas_turb ]
```

```
}
```

where

- **Pr_t** *float*
- **sigma_turbulent|sigma** *float*
- **no_alpha**
- **gas_turb**

5.3.10 Stab

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

See also: `diffusion_deriv` (5.3.1)

Usage:

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

where

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

5.3.11 Standard

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

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

See also: `diffusion_deriv` (5.3.1)

Usage:

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

- **mot1** *str* into [*'default_bar'*]: equivalent to `grad_Ubar 1 nu 1 nut 1 nu_transp 1 nut_transp 1 filtrer_resu 1`
- **bloc_diffusion_standard** *bloc_diffusion_standard* (5.3.12)

5.3.12 Bloc_diffusion_standard

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

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

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

See also: `objet_lecture` ([39](#))

Usage:

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

where

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

5.3.13 P1ncp1b

Description: `not_set`

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

Usage:

5.3.14 P1b

Description: `not_set`

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

Usage:

p1b

5.3.15 Negligeable

Description: the diffusivity will not taken in count

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

Usage:

negligeable

5.3.16 Option

Description: not_set

See also: diffusion_deriv (5.3.1)

Usage:

option bloc_lecture

where

- **bloc_lecture** *bloc_lecture* (3.60)

5.3.17 Op_implicite

Description: not_set

See also: objet_lecture (39)

Usage:

implicite mot solveur

where

- **implicite** *str* into ['implicite']
- **mot** *str* into ['solveur']
- **solveur** *solveur_sys_base* (11.16)

5.4 Condlims

Description: Boundary conditions.

See also: listobj (38.5)

Usage:

{ object1 object2 }

list of *condlimlu* (5.4.1)

5.4.1 Condlimlu

Description: Boundary condition specified.

See also: objet_lecture (39)

Usage:

bord cl

where

- **bord** *str*: Name of the edge where the boundary condition applies.
- **cl** *condlim_base* (13): Boundary condition at the boundary called bord (edge).

5.5 Condinit

Description: Initial conditions.

See also: `listobj` ([38.5](#))

Usage:

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

list of `condinit` ([5.5.1](#))

5.5.1 Condinit

Description: Initial condition.

See also: `objet_lecture` ([39](#))

Usage:

nom ch

where

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

5.6 Sources

Description: The sources.

See also: `listobj` ([38.5](#))

Usage:

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

list of `source_base` ([34](#)) separated with ,

5.7 Parametre_equation_base

Description: Basic class for `parametre_equation`

See also: `objet_lecture` ([39](#)) `parametre_implicit` ([5.7.1](#)) `parametre_diffusion_implicit` ([5.7.2](#))

Usage:

5.7.1 Parametre_implicit

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

See also: `parametre_equation_base` ([5.7](#))

Usage:

parametre_implicit {

[**seuil_convergence_implicit** *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* (11.16): Keyword to change for this equation only the solver used in the implicit scheme
- **resolution_explicite** : To solve explicitly the equation whereas the scheme is an implicit scheme.
- **equation_non_resolue** : Keyword to specify that the equation is not solved.
- **equation_frequence_resolue** *str*: Keyword to specify that the equation is solved only every *n* time steps (*n* is an integer or given by a time-dependent function *f(t)*).

5.7.2 Parametre_diffusion_implicit

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

See also: `parametre_equation_base` (5.7)

Usage:

```

parametre_diffusion_implicit {
    [ crank int into [0, 1]]
    [ preconditionnement_diag int into [0, 1]]
    [ niter_max_diffusion_implicit int]
    [ seuil_diffusion_implicit float]
    [ solveur solveur_sys_base]
}
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_implicit** *int*: Change the maximum number of iterations for the CG (Conjugate Gradient) algorithm when solving the diffusion implication of the equation.
- **seuil_diffusion_implicit** *float*: Change the threshold convergence value used by default for the CG resolution for the diffusion implication of this equation.
- **solveur** *solveur_sys_base* (11.16): Method (different from the default one, Conjugate Gradient) to solve the linear system.

5.8 Conduction_ibm

Description: IBM Heat equation.

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

See also: Conduction ([5.1](#))

Usage:

Conduction_ibm *str*

```
Read str {  
    [ correction_variable_initiale int]  
    [ disable_equation_residual int]  
    [ convection bloc_convection]  
    [ diffusion bloc_diffusion]  
    [ boundary_conditions|conditions_limites condlims]  
    [ initial_conditions|conditions_initiales condinits]  
    [ sources sources]  
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]  
    [ parametre_equation parametre_equation_base]  
    [ equation_non_resolue str]  
    [ renommer_equation str]  
}  
where
```

- **correction_variable_initiale** *int*: Modify initial variable
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* ([5.2](#)) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* ([5.3](#)) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* ([5.4](#)) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* ([5.5](#)) for inheritance: Initial conditions.
- **sources** *sources* ([5.6](#)) 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* ([3.40](#)) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* ([5.7](#)) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.9 Convection_diffusion_espece_binaire_turbulent_qc

Description: Species conservation equation for a binary quasi-compressible fluid as well as the associated turbulence model equations.

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

See also: convection_diffusion_espece_binaire_QC ([5.23](#))

Usage:

Convection_Diffusion_Espece_Binaire_Turbulent_QC *str*

```
Read str {  
    [ modele_turbulence modele_turbulence_scal_base]
```

```

[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
}
where

```

- **modele_turbulence** *modele_turbulence_scal_base* (23): Turbulence model for the species conservation equation.
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.10 Echelle_temporelle_turbulente

Description: Turbulent Dissipation time scale equation for a turbulent mono/multi-phase problem (available in TrioCFD)

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

See also: eqn_base (5.32)

Usage:

Echelle_temporelle_turbulente *str*

Read *str* {

```

[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]

```

```

[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
}
where

```

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.11 Energie_multiphase

Description: Internal energy conservation equation for a multi-phase problem where the unknown is the temperature

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

See also: eqn_base (5.32)

Usage:

Energie_Multiphase *str*

Read *str* {

```

[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]

```

```

}
where

```

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limit** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.12 Energie_multiphase_h

Description: Internal energy conservation equation for a multi-phase problem where the unknown is the enthalpy

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

See also: eqn_base (5.32)

Usage:

Energie_Multiphase_h *str*

```
Read str {
    [ disable_equation_residual int]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ boundary_conditions|conditions_limit
```

```
}]
```

where

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limit** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.13 Energie_cinetique_turbulente

Description: Turbulent kinetic Energy conservation equation for a turbulent mono/multi-phase problem (available in TrioCFD)

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

See also: eqn_base (5.32)

Usage:

Energie_cinetique_turbulente *str*

```
Read str {
    [ disable_equation_residual int]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ boundary_conditions|conditions_limites condlims]
    [ initial_conditions|conditions_initiales condinits]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
    [ renommer_equation str]
}
```

where

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }

- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.14 Energie_cinetique_turbulente_wit

Description: Bubble Induced Turbulent kinetic Energy equation for a turbulent multi-phase problem (available in TrioCFD)

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

See also: eqn_base (5.32)

Usage:

Energie_cinetique_turbulente_WIT *str*

```
Read str {
    [ disable_equation_residual int]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ boundary_conditions|conditions_limite condlims]
    [ initial_conditions|conditions_initiales condinits]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
    [ renommer_equation str]
}
```

where

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limite** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
 Navier_Sokes_Standard
 { equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.15 Masse_multiphase

Description: Mass conservation equation for a multi-phase problem where the unknown is the alpha (void fraction)

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

See also: `eqn_base` (5.32)

Usage:

Masse_Multiphase *str*

Read *str* {

```
[ disable_equation_residual int]  
[ convection bloc_convection]  
[ diffusion bloc_diffusion]  
[ boundary_conditions|conditions_limites condlims]  
[ initial_conditions|conditions_initiales condinits]  
[ sources sources]  
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]  
[ parametre_equation parametre_equation_base]  
[ equation_non_resolue str]  
[ renommer_equation str]
```

}

where

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if `equation_non_resolue` keyword is used. Exemple: The Navier-Stokes equations are not solved between time `t0` and `t1`.
`Navier_Sokes_Standard`
{ `equation_non_resolue` (`t>t0`)*(`t<t1`) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.16 Qdm_multiphase

Description: Momentum conservation equation for a multi-phase problem where the unknown is the velocity

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

See also: `eqn_base` (5.32)

Usage:

QDM_Multiphase *str*

Read *str* {

```
[ solveur_pression solveur_sys_base]  
[ evanescence bloc_lecture]
```

```

[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
}
where

```

- **solveur_pression** *solveur_sys_base* (11.16): Linear pressure system resolution method.
- **evanescence** *bloc_lecture* (3.60): Management of the vanishing phase (when alpha tends to 0 or 1)
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.17 Taux_dissipation_turbulent

Description: Turbulent Dissipation frequency equation for a turbulent mono/multi-phase problem (available in TrioCFD)

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

See also: eqn_base (5.32)

Usage:

Taux_dissipation_turbulent *str*

Read *str* {

```

[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]

```



```

[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
}
where

```

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.18 Convection_diffusion_chaleur_qc

Description: Temperature equation for a quasi-compressible fluid.

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

See also: eqn_base (5.32) convection_diffusion_chaleur_turbulent_qc (5.20)

Usage:

convection_diffusion_chaleur_QC *str*

Read *str* {

```

[ mode_calcul_convection str into ['ancien', 'divuT_moins_Tdivu', 'divrhout_moins_Tdivrhout']]
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]

```

```

}
where

```

- **mode_calcul_convection** *str* into [*'ancien'*, *'divuT_moins_Tdivu'*, *'divrhout_moins_Tdivrhout'*]: Option to set the form of the convective operator
divrhout_moins_Tdivrhout (the default since 1.6.8): $\rho u \cdot \text{grad} T = \text{div}(\rho u \cdot T) - T \text{div}(\rho u)$
ancien: $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \text{div}(u)$
divuT_moins_Tdivu : $u \cdot \text{grad} T = \text{div}(u \cdot T) - T \text{div}(u)$
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.19 Convection_diffusion_chaleur_wc

Description: Temperature equation for a weakly-compressible fluid.

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

See also: eqn_base (5.32)

Usage:

convection_diffusion_chaleur_WC *str*

Read *str* {

```
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
```

}

where

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.

- **boundary_conditions|conditions_limit** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.20 Convection_diffusion_chaleur_turbulent_qc

Description: Temperature equation for a quasi-compressible fluid as well as the associated turbulence model equations.

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

See also: convection_diffusion_chaleur_QC (5.18)

Usage:

convection_diffusion_chaleur_turbulent_qc *str*

Read *str* {

```
[ modele_turbulence modele_turbulence_scal_base]
[ mode_calcul_convection str into ['ancien', 'divuT_moins_Tdivu', 'divrhout_moins_Tdivrhout']]
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limit
```

}

where

- **modele_turbulence** *modele_turbulence_scal_base* (23): Turbulence model for the temperature (energy) conservation equation.
- **mode_calcul_convection** *str into ['ancien', 'divuT_moins_Tdivu', 'divrhout_moins_Tdivrhout']* for inheritance: Option to set the form of the convective operator
divrhout_moins_Tdivrhout (the default since 1.6.8): $\rho \cdot u \cdot \text{grad}T = \text{div}(\rho \cdot u \cdot T) - T \cdot \text{div}(\rho \cdot u)$
ancien: $u \cdot \text{grad}T = \text{div}(u \cdot T) - T \cdot \text{div}(u)$
divuT_moins_Tdivu : $u \cdot \text{grad}T = \text{div}(u \cdot T) - T \cdot \text{div}(u)$
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.

- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.21 Convection_diffusion_concentration

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

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

See also: eqn_base (5.32) convection_diffusion_concentration_turbulent (5.22)

Usage:

convection_diffusion_concentration *str*

Read *str* {

```
[ nom_inconnue str ]
[ alias str ]
[ masse_molaire float ]
[ is_multi_scalar_diffusion|is_multi_scalar ]
[ disable_equation_residual int ]
[ convection bloc_convection ]
[ diffusion bloc_diffusion ]
[ boundary_conditions|conditions_limites condlims ]
[ initial_conditions|conditions_initiales condinits ]
[ sources sources ]
[ écrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur ]
[ parametre_equation parametre_equation_base ]
[ equation_non_resolue str ]
[ renommer_equation 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).
- **alias** *str*
- **masse_molaire** *float*
- **is_multi_scalar_diffusion|is_multi_scalar** : Flag to activate the multi_scalar diffusion operator

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.22 Convection_diffusion_concentration_turbulent

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

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

See also: convection_diffusion_concentration (5.21)

Usage:

convection_diffusion_concentration_turbulent *str*

Read *str* {

```
[ modele_turbulence modele_turbulence_scal_base]
[ nom_inconnue str]
[ alias str]
[ masse_molaire float]
[ is_multi_scalar_diffusion|is_multi_scalar ]
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
```

}

where

- **modele_turbulence** *modele_turbulence_scal_base* (23): Turbulence model to be used in the constituent transport equations. The only model currently available is Schmidt.

- **nom_inconnue** *str* for inheritance: Keyword `Nom_inconnue` will rename the unknown of this equation with the given name. In the postprocessing part, the concentration field will be accessible with this name. This is useful if you want to track more than one concentration (otherwise, only the concentration field in the first concentration equation can be accessed).
- **alias** *str* for inheritance
- **masse_molaire** *float* for inheritance
- **is_multi_scalar_diffusion** *is_multi_scalar* for inheritance: Flag to activate the multi_scalar diffusion operator
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if `equation_non_resolue` keyword is used. Exemple: The Navier-Stokes equations are not solved between time `t0` and `t1`.
`Navier_Sokes_Standard`
`{ equation_non_resolue (t>t0)*(t<t1) }`
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.23 Convection_diffusion_espece_binaire_qc

Description: Species conservation equation for a binary quasi-compressible fluid.

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

See also: `eqn_base` (5.32) `Convection_Diffusion_Espece_Binaire_Turbulent_QC` (5.9)

Usage:

convection_diffusion_espece_binaire_QC *str*

```
Read str {
    [ disable_equation_residual int]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ boundary_conditions|conditions_limites condlims]
    [ initial_conditions|conditions_initiales condinits]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
    [ renommer_equation str]
}
```

where

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step

- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.24 Convection_diffusion_espece_binaire_wc

Description: Species conservation equation for a binary weakly-compressible fluid.

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

See also: eqn_base (5.32)

Usage:

convection_diffusion_espece_binaire_WC *str*

Read *str* {

```
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
```

}

where

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file

- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.25 Convection_diffusion_espece_multi_qc

Description: Species conservation equation for a multi-species quasi-compressible fluid.

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

See also: eqn_base (5.32)

Usage:

convection_diffusion_espece_multi_QC *str*

```
Read str {
    [ espece espece]
    [ disable_equation_residual int]
    [ convection bloc_convection]
    [ diffusion bloc_diffusion]
    [ boundary_conditions|conditions_limites condlims]
    [ initial_conditions|conditions_initiales condinits]
    [ sources sources]
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
    [ parametre_equation parametre_equation_base]
    [ equation_non_resolue str]
    [ renommer_equation str]
}
```

where

- **espece** *espece* (3.42): Associate a species (with its properties) to the equation
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.26 Convection_diffusion_espece_multi_wc

Description: Species conservation equation for a multi-species weakly-compressible fluid.

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

See also: eqn_base (5.32)

Usage:

convection_diffusion_espece_multi_WC *str*

```
Read str {  
    [ disable_equation_residual int]  
    [ convection bloc_convection]  
    [ diffusion bloc_diffusion]  
    [ boundary_conditions|conditions_limites condlims]  
    [ initial_conditions|conditions_initiales condinits]  
    [ sources sources]  
    [ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]  
    [ parametre_equation parametre_equation_base]  
    [ equation_non_resolue str]  
    [ renommer_equation str]  
}
```

where

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.27 Convection_diffusion_espece_multi_turbulent_qc

Description: not_set

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

See also: eqn_base (5.32)

Usage:

convection_diffusion_espece_multi_turbulent_qc *str*

```
Read str {
```

```

[ modele_turbulence modele_turbulence_scal_base]
espece espece
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
}
where

```

- **modele_turbulence** *modele_turbulence_scal_base* (23): Turbulence model to be used.
- **espece** *espece* (3.42)
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.28 Convection_diffusion_temperature

Description: Energy equation (temperature diffusion convection).

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

See also: eqn_base (5.32) convection_diffusion_temperature_ibm (5.29)

Usage:

convection_diffusion_temperature *str*

Read *str* {

```

[ penalisation_l2_ftd bloc_lecture]
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]

```

```

[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
}
where

```

- **penalisation_l2_ftd** *bloc_lecture* (3.60): 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.
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.29 Convection_diffusion_temperature_ibm

Description: IBM Energy equation (temperature diffusion convection).

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

See also: convection_diffusion_temperature (5.28)

Usage:

convection_diffusion_temperature_ibm *str*

Read *str* {

```

[ correction_variable_initiale int]
[ penalisation_l2_ftd bloc_lecture]
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]

```

```

[ equation_non_resolue str]
[ renommer_equation str]
}

```

where

- **correction_variable_initiale** *int*: Modify initial variable
- **penalisation_l2_ftd** *bloc_lecture* (3.60) 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.
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.30 Convection_diffusion_temperature_ibm_turbulent

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

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

See also: eqn_base (5.32)

Usage:

convection_diffusion_temperature_ibm_turbulent *str*

Read *str* {

```

[ modele_turbulence modele_turbulence_scal_base]
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]

```

}
where

- **modele_turbulence** *modele_turbulence_scal_base* (23): Turbulence model for the energy equation.
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.31 Convection_diffusion_temperature_turbulent

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

Keyword Discretize should have already been used to read the object.
See also: eqn_base (5.32)

Usage:

convection_diffusion_temperature_turbulent *str*

Read *str* {

```
[ modele_turbulence modele_turbulence_scal_base]
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
```

}
where

- **modele_turbulence** *modele_turbulence_scal_base* (23): Turbulence model for the energy equation.

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.32 Eqn_base

Description: Basic class for equations.

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

See also: mor_eqn (5) Conduction (5.1) convection_diffusion_temperature (5.28) navier_stokes_standard (5.41) convection_diffusion_temperature_ibm_turbulent (5.30) Energie_Multiphase (5.11) Energie_Multiphase_h (5.12) Masse_Multiphase (5.15) QDM_Multiphase (5.16) Echelle_temporelle_turbulente (5.10) Energie_cinetique_turbulente (5.13) Energie_cinetique_turbulente_WIT (5.14) Taux_dissipation_turbulent (5.17) convection_diffusion_espece_multi_turbulent_qc (5.27) convection_diffusion_concentration (5.21) convection_diffusion_chaleur_QC (5.18) convection_diffusion_temperature_turbulent (5.31) convection_diffusion_espece_binaire_QC (5.23) convection_diffusion_chaleur_WC (5.19) convection_diffusion_espece_multi_QC (5.25) convection_diffusion_espece_binaire_WC (5.24) convection_diffusion_espece_multi_WC (5.26)

Usage:

eqn_base *str*

Read *str* {

```
[ disable_equation_residual int ]
[ convection bloc_convection ]
[ diffusion bloc_diffusion ]
[ boundary_conditions|conditions_limites condlims ]
[ initial_conditions|conditions_initiales condinits ]
[ sources sources ]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur ]
[ parametre_equation parametre_equation_base ]
[ equation_non_resolue str ]
[ renommer_equation str ]
```

}

where

- **disable_equation_residual** *int*: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step

- **convection** *bloc_convection* (5.2): Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3): Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limit** *condlims* (5.4): Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5): Initial conditions.
- **sources** *sources* (5.6): 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* (3.40): This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7): Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str*: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str*: Rename the equation with a specific name.

5.33 Navier_stokes_qc

Description: Navier-Stokes equation for a quasi-compressible fluid.

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

See also: *navier_stokes_standard* (5.41)

Usage:

navier_stokes_QC *str*

Read *str* {

```
[ solveur_pression solveur_sys_base]
[ dt_projection deuxmots]
[ traitement_particulier traitement_particulier]
[ seuil_divU floatfloat]
[ solveur_bar solveur_sys_base]
[ projection_initiale int]
[ postraiter_gradient_pression_sans_masse ]
[ methode_calcul_pression_initiale str into ['avec_les_cl', 'avec_sources', 'avec_sources_et-
_operateurs', 'sans_rien']]
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limit
```

}

where

- **solveur_pression** *solveur_sys_base* (11.16) for inheritance: Linear pressure system resolution method.

- **dt_projection** *deuxmots* (5.34) 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.
- **traitement_particulier** *traitement_particulier* (5.35) for inheritance: Keyword to post-process particular values.
- **seuil_divU** *floatfloat* (5.36) 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)
- **solveur_bar** *solveur_sys_base* (11.16) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source_Qdm_lambdaup). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **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.
- **postraiter_gradient_pression_sans_masse** for inheritance: Avoid mass matrix multiplication for the gradient postprocessing
- **methode_calcul_pression_initiale** *str into* ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien'] for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : avec_les_cl (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), avec_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limit** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.
 Navier_Sokes_Standard
 { equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.34 Deuxmots

Description: Two words.

See also: [objet_lecture \(39\)](#)

Usage:

mot_1 mot_2

where

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

5.35 Traitement_particulier

Description: Auxiliary class to post-process particular values.

See also: [objet_lecture \(39\)](#)

Usage:

aco trait_part acof

where

- **aco** *str* into ['']: Opening curly bracket.
- **trait_part** *traitement_particulier_base (5.35.1)*: Type of *traitement_particulier*.
- **acof** *str* into ['']: Closing curly bracket.

5.35.1 Traitement_particulier_base

Description: Basic class to post-process particular values.

See also: [objet_lecture \(39\)](#) [profils_thermo \(5.35.2\)](#) [temperature \(5.35.3\)](#) [canal \(5.35.4\)](#) [chmoy_faceperio \(5.35.5\)](#) [ec \(5.35.6\)](#) [thi \(5.35.7\)](#)

Usage:

5.35.2 Profils_thermo

Description: non documente

See also: [traitement_particulier_base \(5.35.1\)](#)

Usage:

profils_thermo bloc

where

- **bloc** *bloc_lecture (3.60)*

5.35.3 Temperature

Description: not_set

See also: [traitement_particulier_base \(5.35.1\)](#)

Usage:

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

where

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

5.35.4 Canal

Description: Keyword for statistics on a periodic plane channel.

See also: `traitement_particulier_base` ([5.35.1](#))

Usage:

```
canal {  
    [ dt_impr_moy_spat float]  
    [ dt_impr_moy_temp float]  
    [ debut_stat float]  
    [ fin_stat float]  
    [ pulsation_w float]  
    [ nb_points_par_phase int]  
    [ reprise str]  
}
```

where

- **dt_impr_moy_spat** *float*: Period to print the spatial average (default value is 1e6).
- **dt_impr_moy_temp** *float*: Period to print the temporal average (default value is 1e6).
- **debut_stat** *float*: Time to start the temporal averaging (default value is 1e6).
- **fin_stat** *float*: Time to end the temporal averaging (default value is 1e6).
- **pulsation_w** *float*: Pulsation for phase averaging (in case of pulsating forcing term) (no default value).
- **nb_points_par_phase** *int*: Number of samples to represent phase average all along a period (no default value).
- **reprise** *str*: `val_moy_temp_xxxxxx.sauv` : Keyword to resume a calculation with previous averaged quantities.

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

5.35.5 Chmoy_faceperio

Description: non documente

See also: `traitement_particulier_base` ([5.35.1](#))

Usage:

chmoy_faceperio bloc

where

- **bloc** *bloc_lecture* ([3.60](#))

5.35.6 Ec

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

See also: *traitement_particulier_base* ([5.35.1](#))

Usage:

```
ec {  
  
    [ Ec ]  
    [ Ec_dans_repere_fixe ]  
    [ periode float ]  
  
}
```

where

- **Ec**
- **Ec_dans_repere_fixe**
- **periode** *float*: periode is the keyword to set the period of printing into the file *datafile_Ec.son* or *datafile_Ec_dans_repere_fixe.son*.

5.35.7 Thi

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

See also: *traitement_particulier_base* ([5.35.1](#))

Usage:

```
thi {  
  
    init_Ec int  
    [ val_Ec float ]  
    [ facon_init int into [0, 1]]  
    [ calc_spectre int into [0, 1]]  
    [ periode_calc_spectre float ]  
    [ spectre_3D int into [0, 1]]  
    [ spectre_1D int into [0, 1]]  
    [ conservation_Ec ]  
    [ longueur_boite float ]  
  
}
```

where

- **init_Ec** *int*: Keyword to renormalize initial velocity so that kinetic energy equals to the value given by keyword *val_Ec*.
- **val_Ec** *float*: Keyword to impose a value for kinetic energy by velocity renormalized if *init_Ec* value is 1.

- **facon_init** *int into [0, 1]*: Keyword to specify how kinetic energy is computed (0 or 1).
- **calc_spectre** *int into [0, 1]*: Calculate or not the spectrum of kinetic energy.
Files called Sorties_THI are written with inside four columns :
time:t global_kinetic_energy:Ec enstrophy:D skewness:S
If calc_spectre is set to 1, a file Sorties_THI2_2 is written with three columns :
time:t kinetic_energy_at_kc=32 enstrophy_at_kc=32
If calc_spectre is set to 1, a file spectre_XXXXX is written with two columns at each time XXXXX :
frequency:k energy:E(k).
- **periode_calc_spectre** *float*: Period for calculating spectrum of kinetic energy
- **spectre_3D** *int into [0, 1]*: Calculate or not the 3D spectrum
- **spectre_1D** *int into [0, 1]*: Calculate or not the 1D spectrum
- **conservation_Ec** : If set to 1, velocity field will be changed as to have a constant kinetic energy (default 0)
- **longueur_boite** *float*: Length of the calculation domain

5.36 Floatfloat

Description: Two reals.

See also: objet_lecture (39)

Usage:

a b

where

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

5.37 Navier_stokes_wc

Description: Navier-Stokes equation for a weakly-compressible fluid.

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

See also: navier_stokes_standard (5.41)

Usage:

navier_stokes_WC *str*

Read *str* {

```
[ mass_source mass_source]
[ solveur_pression solveur_sys_base]
[ dt_projection deuxmots]
[ traitement_particulier traitement_particulier]
[ seuil_divU floatfloat]
[ solveur_bar solveur_sys_base]
[ projection_initiale int]
[ postraiter_gradient_pression_sans_masse ]
[ methode_calcul_pression_initiale str into ['avec_les_cl', 'avec_sources', 'avec_sources_et-
_operateurs', 'sans_rien']]
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
```

```

[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
}
where

```

- **mass_source** *mass_source* (3.74): Mass source used in a dilatable simulation to add/reduce a mass at the boundary (volumetric source in the first cell of a given boundary).
- **solveur_pression** *solveur_sys_base* (11.16) for inheritance: Linear pressure system resolution method.
- **dt_projection** *deuxmots* (5.34) 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.
- **traitement_particulier** *traitement_particulier* (5.35) for inheritance: Keyword to post-process particular values.
- **seuil_divU** *floatfloat* (5.36) 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)
- **solveur_bar** *solveur_sys_base* (11.16) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source_Qdm_lambdaup). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **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.
- **postraiter_gradient_pression_sans_masse** for inheritance: Avoid mass matrix multiplication for the gradient postprocessing
- **methode_calcul_pression_initiale** *str into* ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien'] for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : avec_les_cl (default option $\text{lapP}=0$ is solved with Neuman boundary conditions on pressure if any), avec_sources ($\text{lapP}=f$ is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec_sources_et_operateurs ($\text{lapP}=f$ is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.38 Navier_stokes_ibm

Description: IBM Navier-Stokes equations.

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

See also: *navier_stokes_standard* (5.41)

Usage:

navier_stokes_ibm *str*

Read *str* {

```
[ correction_matrice_projection_initiale int]
[ correction_calcul_pression_initiale int]
[ correction_vitesse_projection_initiale int]
[ correction_matrice_pression int]
[ matrice_pression_penalisee_H1 int]
[ correction_vitesse_modifie int]
[ correction_pression_modifie int]
[ gradient_pression_qdm_modifie int]
[ correction_variable_initiale int]
[ solveur_pression solveur_sys_base]
[ dt_projection deuxmots]
[ traitement_particulier traitement_particulier]
[ seuil_divU floatfloat]
[ solveur_bar solveur_sys_base]
[ projection_initiale int]
[ postraiter_gradient_pression_sans_masse ]
[ methode_calcul_pression_initiale str into ['avec_les_cl', 'avec_sources', 'avec_sources_et-
_operateurs', 'sans_rien']]
[ disable_equation_residual int]
[ convection bloc_convection]
[ diffusion bloc_diffusion]
[ boundary_conditions|conditions_limites condlims]
[ initial_conditions|conditions_initiales condinits]
[ sources sources]
[ ecrire_fichier_xyz_valeur ecrire_fichier_xyz_valeur]
[ parametre_equation parametre_equation_base]
[ equation_non_resolue str]
[ renommer_equation str]
```

}

where

- **correction_matrice_projection_initiale** *int*: (IBM advanced) fix matrix of initial projection for PDF
 - **correction_calcul_pression_initiale** *int*: (IBM advanced) fix initial pressure computation for PDF
 - **correction_vitesse_projection_initiale** *int*: (IBM advanced) fix initial velocity computation for PDF
 - **correction_matrice_pression** *int*: (IBM advanced) fix pressure matrix for PDF
 - **matrice_pression_penalisee_H1** *int*: (IBM advanced) fix pressure matrix for PDF
 - **correction_vitesse_modifie** *int*: (IBM advanced) fix velocity for PDF
 - **correction_pression_modifie** *int*: (IBM advanced) fix pressure for PDF
 - **gradient_pression_qdm_modifie** *int*: (IBM advanced) fix pressure gradient
 - **correction_variable_initiale** *int*: Modify initial variable
 - **solveur_pression** *solveur_sys_base* (11.16) for inheritance: Linear pressure system resolution method.
-
- **dt_projection** *deuxmots* (5.34) 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.
 - **traitement_particulier** *traitement_particulier* (5.35) for inheritance: Keyword to post-process particular values.
 - **seuil_divU** *floatfloat* (5.36) 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)
 - **solveur_bar** *solveur_sys_base* (11.16) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and *Source_Qdm_lambdaup*). A file (*solveur.bar*) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
 - **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.
 - **postraiter_gradient_pression_sans_masse** for inheritance: Avoid mass matrix multiplication for the gradient postprocessing
 - **methode_calcul_pression_initiale** *str into* ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien'] for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : *avec_les_cl* (default option $\text{lapP}=0$ is solved with Neuman boundary conditions on pressure if any), *avec_sources* ($\text{lapP}=f$ is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and *avec_sources_et_operateurs* ($\text{lapP}=f$ is solved as with the previous option *avec_sources* but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
 - **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
 - **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
 - **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
 - **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
 - **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
 - **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.39 Navier_stokes_ibm_turbulent

Description: IBM Navier-Stokes equations as well as the associated turbulence model equations.

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

See also: navier_stokes_standard (5.41)

Usage:

navier_stokes_ibm_turbulent *str*

Read *str* {

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

}

where

- **modele_turbulence** *modele_turbulence_hyd_deriv* (5.40): Turbulence model for Navier-Stokes equations.
- **solveur_pression** *solveur_sys_base* (11.16) for inheritance: Linear pressure system resolution method.
- **dt_projection** *deuxmots* (5.34) 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.

- **traitement_particulier** *traitement_particulier* (5.35) for inheritance: Keyword to post-process particular values.
- **seuil_divU** *floatfloat* (5.36) 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)*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)
- **solveur_bar** *solveur_sys_base* (11.16) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source_Qdm_lambdaup). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **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.
- **postraiter_gradient_pression_sans_masse** for inheritance: Avoid mass matrix multiplication for the gradient postprocessing
- **methode_calcul_pression_initiale** *str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']* for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : avec_les_cl (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), avec_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limite** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t_0 and t_1 .

```
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
```
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.40 Modele_turbulence_hyd_deriv

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

See also: objet_lecture (39) mod_turb_hyd_ss_maille (5.40.2) null (5.40.7)

Usage:

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

where

- **turbulence_paroit** *turbulence_paroit_base* (36): Keyword to set the wall law.
- **dt_impr_ustar** *float*: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.40.1): This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float*: Upper limitation of turbulent viscosity (default value 1.e8).
- **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]

5.40.1 Dt_impr_ustar_mean_only

Description: not_set

See also: objet_lecture (39)

Usage:

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

where

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

5.40.2 Mod_turb_hyd_ss_maille

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

See also: modele_turbulence_hyd_deriv (5.40) sous_maille_smago (5.40.4) sous_maille_wale (5.40.5) longueur_melange (5.40.6)

Usage:

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

- **formulation_a_nb_points form_a_nb_points** (5.40.3): The structure fonction is calculated on nb points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homogeneity planes. Example for channel flows, planes parallel to the walls.
- **longueur_maille str into ['volume', 'volume_sans_lissage', 'scotti', 'arrete']**: Different ways to calculate the characteristic length may be specified :
volume : It is the default option. Characteristic length is based on the cubic root of the volume cells. A smoothing procedure is applied to avoid discontinuities of this quantity in VEF from a cell to another.
volume_sans_lissage : For VEF only. Characteristic length is based on the cubic root of the volume cells (without smoothing procedure).
scotti : Characteristic length is based on the cubic root of the volume cells and the Scotti correction is applied to take into account the stretching of the cell in the case of anisotropic meshes.
arete : For VEF only. Characteristic length relies on the max edge (+ smoothing procedure) is taken into account.
- **turbulence_paroit turbulence_paroit_base** (36) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar float** for inheritance: This keyword is used to print the values (U^+ , d^+ , u^*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only dt_impr_ustar_mean_only** (5.40.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max float** for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **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]

5.40.3 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 \(39\)](#)

Usage:

nb dir1 dir2

where

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

5.40.4 Sous_maille_smago

Description: Smagorinsky sub-grid turbulence model.

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

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

See also: [mod_turb_hyd_ss_maille \(5.40.2\)](#)

Usage:

sous_maille_smago {

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

}

where

- **cs** *float*: This is an optional keyword and the value is used to set the constant used in the Smagorinsky model (This is currently only valid for Smagorinsky models and it is set to 0.18 by default) .
- **formulation_a_nb_points** *form_a_nb_points (5.40.3)* 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.

- **turbulence_paro** *turbulence_paro_base* (36) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.40.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **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]

5.40.5 Sous_maille_wale

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

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

See also: `mod_turb_hyd_ss_maille` (5.40.2)

Usage:

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

where

- **cw** *float*: The unique parameter (constant) of the WALE-model (by default value 0.5).
- **formulation_a_nb_points** *form_a_nb_points* (5.40.3) 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.
- **turbulence_paro** *turbulence_paro_base* (36) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.40.1) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **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]

5.40.6 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 (5.40.2)

Usage:

```
longueur_melange {
    [ canalx float]
    [ tuyauz float]
    [ 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']]
}
```

```

[ turbulence_paroi turbulence_paroi_base]
[ dt_impr_ustar float]
[ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]
[ nut_max float]
[ correction_visco_turb_pour_controle_pas_de_temps ]
[ correction_visco_turb_pour_controle_pas_de_temps_parametre 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).
- **dmax** *float*: Maximum distance.
- **fichier** *str*
- **fichier_ecriture_K_Eps** *str*: When a resume with k-epsilon model is envisaged, this keyword allows to generate external MED-format file with evaluation of k and epsilon quantities (based on eddy turbulent viscosity and turbulent characteristic length returned by mixing length model). The frequency of the MED file print is set equal to dt_impr_ustar. Moreover, k-eps MED field is automatically saved at the last time step. MED file is then used for resuming a K-Epsilon calculation with the Champ_Fonc_Med keyword.
- **formulation_a_nb_points** *form_a_nb_points* (5.40.3) 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.
- **turbulence_paro**i *turbulence_paro*i_base (36) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named datafile_ProblemName_Ustar.face and periode refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.40.1) for inheritance: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_Ustar_mean_only.out. periode refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword boundaries, all the boundaries will be considered. If you use it, you must specify nb_boundaries which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **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]

5.40.7 Null

Description: Null turbulence model (turbulent viscosity = 0) which can be used with a turbulent problem.

See also: `modele_turbulence_hyd_deriv` (5.40)

Usage:

```
null {  
    [ turbulence_paroi turbulence_paro_base]  
    [ dt_impr_ustar float]  
    [ dt_impr_ustar_mean_only dt_impr_ustar_mean_only]  
    [ nut_max float]  
    [ correction_visco_turb_pour_controle_pas_de_temps ]  
    [ correction_visco_turb_pour_controle_pas_de_temps_parametre float]  
}
```

where

- **turbulence_paro**i *turbulence_paro_base* (36) for inheritance: Keyword to set the wall law.
- **dt_impr_ustar** *float* for inheritance: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named `datafile_ProblemName_Ustar.face` and `periode` refers to the printing period, this value is expressed in seconds.
- **dt_impr_ustar_mean_only** *dt_impr_ustar_mean_only* (5.40.1) for inheritance: This keyword is used to print the mean values of u^* (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_Ustar_mean_only.out`. `periode` refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword `boundaries`, all the boundaries will be considered. If you use it, you must specify `nb_boundaries` which is the number of boundaries on which you want to calculate the mean values of u^* , then you have to specify their names.
- **nut_max** *float* for inheritance: Upper limitation of turbulent viscosity (default value 1.e8).
- **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]

5.41 Navier_stokes_standard

Description: Navier-Stokes equations.

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

See also: `eqn_base` (5.32) `navier_stokes_ibm_turbulent` (5.39) `navier_stokes_ibm` (5.38) `navier_stokes_turbulent` (5.42) `navier_stokes_QC` (5.33) `navier_stokes_WC` (5.37)

Usage:

```
navier_stokes_standard str  
Read str {
```



```

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

```

- **solveur_pression** *solveur_sys_base* (11.16): Linear pressure system resolution method.
- **dt_projection** *deuxmots* (5.34): 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.
- **traitement_particulier** *traitement_particulier* (5.35): Keyword to post-process particular values.
- **seuil_divU** *floatfloat* (5.36): value factor : this keyword is intended to minimise the number of iterations during the pressure system resolution. The convergence criteria during this step ('seuil' in solveur_pression) is dynamically adapted according to the mass conservation. At t_n , the linear system $Ax=B$ is considered as solved if the residual $\|Ax-B\| < \text{seuil}(t_n)$. For t_{n+1} , the threshold value $\text{seuil}(t_{n+1})$ will be evaluated as:
 If ($\text{lmax}(\text{DivU}) * dt < \text{value}$)
 $\text{Seuil}(t_{n+1}) = \text{Seuil}(t_n) * \text{factor}$
 Else
 $\text{Seuil}(t_{n+1}) = \text{Seuil}(t_n) * \text{factor}$
 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)
- **solveur_bar** *solveur_sys_base* (11.16): This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source_Qdm_lambdaup). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **projection_initiale** *int*: Keyword to suppress, if boolean equals 0, the initial projection which checks $\text{DivU}=0$. By default, boolean equals 1.
- **postraiter_gradient_pression_sans_masse** : Avoid mass matrix multiplication for the gradient postprocessing
- **methode_calcul_pression_initiale** *str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']*: Keyword to select an option for the pressure calculation before the first time step. Options are : avec_les_cl (default option $\text{lapP}=0$ is solved with Neuman boundary conditions on pressure if any), avec_sources ($\text{lapP}=f$ is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec_sources_et_operateurs ($\text{lapP}=f$ is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes

equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.

- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Exemple: The Navier-Stokes equations are not solved between time t0 and t1.
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.42 Navier_stokes_turbulent

Description: Navier-Stokes equations as well as the associated turbulence model equations.

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

See also: *navier_stokes_standard* (5.41) *navier_stokes_turbulent_qc* (5.43)

Usage:

navier_stokes_turbulent *str*

Read *str* {

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

}
where

- **modele_turbulence** *modele_turbulence_hyd_deriv* (5.40): Turbulence model for Navier-Stokes equations.
- **solveur_pression** *solveur_sys_base* (11.16) for inheritance: Linear pressure system resolution method.
- **dt_projection** *deuxmots* (5.34) 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.
- **traitement_particulier** *traitement_particulier* (5.35) for inheritance: Keyword to post-process particular values.
- **seuil_divU** *floatfloat* (5.36) 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 ($\text{lmax}(\text{DivU}) \cdot dt < \text{value}$)
 Seuil(t_{n+1}) = Seuil(t_n) * factor
 Else
 Seuil(t_{n+1}) = Seuil(t_n) * factor
 Endif
 The first parameter (value) is the mass evolution the user is ready to accept per timestep, and the second one (factor) is the factor of evolution for 'seuil' (for example 1.1, so 10)
- **solveur_bar** *solveur_sys_base* (11.16) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and *Source_Qdm_lambdaup*). A file (*solveur.bar*) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **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.
- **postraiter_gradient_pression_sans_masse** for inheritance: Avoid mass matrix multiplication for the gradient postprocessing
- **methode_calcul_pression_initiale** *str into ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien']* for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : *avec_les_cl* (default option $\text{lapP}=0$ is solved with Neuman boundary conditions on pressure if any), *avec_sources* ($\text{lapP}=f$ is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and *avec_sources_et_operateurs* ($\text{lapP}=f$ is solved as with the previous option *avec_sources* but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limtes** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is

verified if `equation_non_resolue` keyword is used. Example: The Navier-Stokes equations are not solved between time t_0 and t_1 .

`Navier_Sokes_Standard`

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

- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

5.43 Navier_stokes_turbulent_qc

Description: Navier-Stokes equations under low Mach number as well as the associated turbulence model equations.

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

See also: `navier_stokes_turbulent` (5.42)

Usage:

navier_stokes_turbulent_qc *str*

Read *str* {

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

}

where

- **modele_turbulence** *modele_turbulence_hyd_deriv* (5.40) for inheritance: Turbulence model for Navier-Stokes equations.
- **solveur_pression** *solveur_sys_base* (11.16) for inheritance: Linear pressure system resolution method.
- **dt_projection** *deuxmots* (5.34) 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.
- **traitement_particulier** *traitement_particulier* (5.35) for inheritance: Keyword to post-process particular values.
- **seuil_divU** *floatfloat* (5.36) 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 `seuil(tn+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

- **solveur_bar** *solveur_sys_base* (11.16) for inheritance: This keyword is used to define when filtering operation is called (typically for EF convective scheme, standard diffusion operator and Source_Qdm_lambdaup). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).
- **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.
- **postraiter_gradient_pression_sans_masse** for inheritance: Avoid mass matrix multiplication for the gradient postprocessing
- **methode_calcul_pression_initiale** *str into* ['avec_les_cl', 'avec_sources', 'avec_sources_et_operateurs', 'sans_rien'] for inheritance: Keyword to select an option for the pressure calculation before the first time step. Options are : avec_les_cl (default option lapP=0 is solved with Neuman boundary conditions on pressure if any), avec_sources (lapP=f is solved with Neuman boundaries conditions and f integrating the source terms of the Navier-Stokes equations) and avec_sources_et_operateurs (lapP=f is solved as with the previous option avec_sources but f integrating also some operators of the Navier-Stokes equations). The two last options are useful and sometime necessary when source terms are implicated when using an implicit time scheme to solve the Navier-Stokes equations.
- **disable_equation_residual** *int* for inheritance: The equation residual will not be used for the problem residual used when checking time convergence or computing dynamic time-step
- **convection** *bloc_convection* (5.2) for inheritance: Keyword to alter the convection scheme.
- **diffusion** *bloc_diffusion* (5.3) for inheritance: Keyword to specify the diffusion operator.
- **boundary_conditions|conditions_limites** *condlims* (5.4) for inheritance: Boundary conditions.
- **initial_conditions|conditions_initiales** *condinits* (5.5) for inheritance: Initial conditions.
- **sources** *sources* (5.6) 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* (3.40) for inheritance: This keyword is used to write the values of a field only for some boundaries in a text file
- **parametre_equation** *parametre_equation_base* (5.7) for inheritance: Keyword used to specify additional parameters for the equation
- **equation_non_resolue** *str* for inheritance: The equation will not be solved while condition(t) is verified if equation_non_resolue keyword is used. Example: The Navier-Stokes equations are not solved between time t0 and t1.

```
Navier_Sokes_Standard
{ equation_non_resolue (t>t0)*(t<t1) }
```
- **renommer_equation** *str* for inheritance: Rename the equation with a specific name.

6 domaine_base

Description: base for most domains

See also: objet_u (40) domaine_ijk (6.1)

Usage:

6.1 Domaine_ijk

Description: domain for IJK simulation (used in TrioCFD)

See also: [Domaine_base \(6\)](#)

Usage:

domaine_ijk *str*

Read *str* {

```
    nbelem n1 n2 (n3)
    size_dom x1 x2 (x3)
    perio n1 n2 (n3)
    nproc n1 n2 (n3)
    [ origin x1 x2 (x3) ]
    [ ijk_splitting_ft_extension int ]
    [ file_coords troismots ]
```

}

where

- **nbelem** *n1 n2 (n3)*: Number of elements in each direction (integers, 2 or 3 values depending on dimension)
- **size_dom** *x1 x2 (x3)*: Domain size in each direction (floats, 2 or 3 values depending on dimension)
- **perio** *n1 n2 (n3)*: Is the direction periodic ? (0 or 1, 2 or 3 values depending on dimension)
- **nproc** *n1 n2 (n3)*: Number of procs in each direction (integers, 2 or 3 values depending on dimension)
- **origin** *x1 x2 (x3)*: Domain origin in each direction (floats, 2 or 3 values depending on dimension)
- **ijk_splitting_ft_extension** *int*
- **file_coords** *troismots* ([6.2](#))

6.2 Troismots

Description: Three words.

See also: [objet_lecture \(39\)](#)

Usage:

mot_1 mot_2 mot_3

where

- **mot_1** *str*: First word.
- **mot_2** *str*: Snd word.
- **mot_3** *str*: Third word.

7 interface_base

Description: Basic class for a liquid-gas interface (used in pb_multiphase)

See also: [objet_u \(40\)](#) [saturation_base \(7.2\)](#) [Interface_sigma_constant \(7.1\)](#)

Usage:

Interface_base *str*

Read *str* {

```

    [ surface_tension/tension_superficielle float]
}
where

    • surface_tension/tension_superficielle float: surface tension

```

7.1 Interface_sigma_constant

Description: Liquid-gas interface with a constant surface tension sigma

See also: Interface_base (7)

Usage:

Interface_sigma_constant *str*

Read *str* {

```

    [ surface_tension/tension_superficielle float]
}
where

```

- surface_tension/tension_superficielle float for inheritance: surface tension

7.2 Saturation_base

Description: fluide-gas interface with phase change (used in pb_multiphase)

See also: Interface_base (7) saturation_sodium (7.4) saturation_constant (7.3)

Usage:

saturation_base *str*

Read *str* {

```

    [ p_ref float]
    [ t_ref float]
    [ surface_tension/tension_superficielle float]
}
where

```

- p_ref float
- t_ref float
- surface_tension/tension_superficielle float for inheritance: surface tension

7.3 Saturation_constant

Description: Class for saturation constant

See also: saturation_base (7.2)

Usage:

saturation_constant *str*

Read *str* {

```

    [ P_sat float]
    [ T_sat float]
    [ Lvap float]
    [ Hlsat float]
    [ Hvsat float]
    [ p_ref float]
    [ t_ref float]
    [ surface_tension/tension_superficielle float]
}
where

```

- **P_sat** *float*: Define the saturation pressure value (this is a required parameter)
- **T_sat** *float*: Define the saturation temperature value (this is a required parameter)
- **Lvap** *float*: Latent heat of vaporization
- **Hlsat** *float*: Liquid saturation enthalpy
- **Hvsat** *float*: Vapor saturation enthalpy
- **p_ref** *float* for inheritance
- **t_ref** *float* for inheritance
- **surface_tension/tension_superficielle** *float* for inheritance: surface tension

7.4 Saturation_sodium

Description: Class for saturation sodium

See also: [saturation_base \(7.2\)](#)

Usage:

saturation_sodium *str*

Read *str* {

```

    [ P_ref float]
    [ T_ref float]
    [ p_ref float]
    [ t_ref float]
    [ surface_tension/tension_superficielle float]

```

```

}
where

```

- **P_ref** *float*: Use to fix the pressure value in the closure law. If not specified, the value of the pressure unknown will be used
- **T_ref** *float*: Use to fix the temperature value in the closure law. If not specified, the value of the temperature unknown will be used
- **p_ref** *float* for inheritance
- **t_ref** *float* for inheritance
- **surface_tension/tension_superficielle** *float* for inheritance: surface tension

8 /*

8.1 /*

Description: bloc of Comment in a data file.

See also: `objet_u` (40)

Usage:

```
/* comm
where
```

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

9 champ_generique_base

Description: `not_set`

See also: `objet_u` (40) `champ_post_de_champs_post` (9.1) `champ_post_refchamp` (9.17) `predefini` (9.15)

Usage:

9.1 Champ_post_de_champs_post

Description: `not_set`

See also: `champ_generique_base` (9) `champ_post_tparoi_veh` (9.18) `champ_post_statistiques_base` (9.6) `champ_post_extraction` (9.10) `champ_post_transformation` (9.19) `champ_post_operateur_base` (9.4) `champ_post_morceau_equation` (9.13) `interpolation` (9.12) `champ_post_reduction_0d` (9.16) `champ_post_operateur_eqn` (9.5)

Usage:

```
champ_post_de_champs_post str
```

```
Read str {
```

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

```
}
```

where

- **source** *champ_generique_base* (9): the source field.
- **sources** *listchamp_generique* (9.2): sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **nom_source** *str*: To name a source field with the `nom_source` keyword
- **source_reference** *str*
- **sources_reference** *list_nom_virgule* (9.3)

9.2 Listchamp_generique

Description: XXX

See also: `listobj` (38.5)

Usage:

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

list of *champ_generique_base* (9) separated with ,

9.3 List_nom_virgule

Description: List of name.

See also: listobj (38.5)

Usage:

{ object1 , object2 }

list of *nom_anonyme* (25.1) separated with ,

9.4 Champ_post_operateur_base

Description: not_set

See also: champ_post_de_champs_post (9.1) champ_post_operateur_gradient (9.11) champ_post_operateur-_divergence (9.8)

Usage:

champ_post_operateur_base *str*

Read *str* {

[**source** *champ_generique_base*]

[**sources** *listchamp_generique*]

[**nom_source** *str*]

[**source_reference** *str*]

[**sources_reference** *list_nom_virgule*]

}

where

- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

9.5 Champ_post_operateur_eqn

Synonymous: **operateur_eqn**

Description: Post-process equation operators/sources

See also: champ_post_de_champs_post (9.1)

Usage:

champ_post_operateur_eqn *str*

Read *str* {

[**numero_source** *int*]

[**numero_op** *int*]

[**numero_masse** *int*]

[**sans_solveur_masse**]

[**compo** *int*]

[**source** *champ_generique_base*]

```

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

```

- **numero_source** *int*: the source to be post-processed (its number). If you have only one source term, numero_source will correspond to 0 if you want to post-process that unique source
- **numero_op** *int*: numero_op will be 0 (diffusive operator) or 1 (convective operator) or 2 (gradient operator) or 3 (divergence operator).
- **numero_masse** *int*: numero_masse will be 0 for the mass equation operator in Pb_multiphase.
- **sans_solveur_masse**
- **compo** *int*: If you want to post-process only one component of a vector field, you can specify the number of the component after compo keyword. By default, it is set to -1 which means that all the components will be post-processed. This feature is not available in VDF discretization.
- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

9.6 Champ_post_statistiques_base

Description: not_set

See also: champ_post_de_champs_post (9.1) moyenne (9.14) ecart_type (9.9) correlation (9.7)

Usage:

champ_post_statistiques_base *str*

Read *str* {

```

t_deb float
t_fin float
[ source champ_generique_base]
[ sources listchamp_generique]
[ nom_source str]
[ source_reference str]
[ sources_reference list_nom_virgule]

```

}
where

- **t_deb** *float*: Start of integration time
- **t_fin** *float*: End of integration time
- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

9.7 Correlation

Synonymous: **champ_post_statistiques_correlation**

Description: to calculate the correlation between the two fields.

See also: `champ_post_statistiques_base` (9.6)

Usage:

correlation *str*

Read *str* {

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

}

where

- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

9.8 Champ_post_operateur_divergence

Synonymous: **divergence**

Description: To calculate divergency of a given field.

See also: `champ_post_operateur_base` (9.4)

Usage:

champ_post_operateur_divergence *str*

Read *str* {

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

}

where

- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }

- **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* (9.3) for inheritance

9.9 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` (9.6)

Usage:

ecart_type *str*

Read *str* {

```

    t_deb float
    t_fin float
    [ source champ_generique_base]
    [ sources listchamp_generique]
    [ nom_source str]
    [ source_reference str]
    [ sources_reference list_nom_virgule]

```

}

where

- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

9.10 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` (9.1)

Usage:

champ_post_extraction *str*

Read *str* {

```

    domaine str
    nom_frontiere str
    [ methode str into ['trace', 'champ_frontiere']]
    [ source champ_generique_base]
    [ sources listchamp_generique]
    [ nom_source str]

```

```

    [ source_reference str]
    [ sources_reference list_nom_virgule]
}

```

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* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

9.11 Champ_post_operateur_gradient

Synonymous: **gradient**

Description: To calculate gradient of a given field.

See also: champ_post_operateur_base (9.4)

Usage:

champ_post_operateur_gradient *str*

Read *str* {

```

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

```

}

where

- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

9.12 Interpolation

Synonymous: **champ_post_interpolation**

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

See also: champ_post_de_champs_post (9.1)

Usage:

interpolation *str*

Read *str* {

```

localisation str
[ methode str]
[ domaine str]
[ optimisation_sous_maillage str into ['default', 'yes', 'no']]
[ source champ_generique_base]
[ sources listchamp_generique]
[ nom_source str]
[ source_reference str]
[ sources_reference list_nom_virgule]
}
where

```

- **localisation** *str*: type_loc indicate where is done the interpolation (elem for element or som for node).
- **methode** *str*: The optional keyword methode is limited to calculer_champ_post for the moment.
- **domaine** *str*: the domain name where the interpolation is done (by default, the calculation domain)
- **optimisation_sous_maillage** *str* into ['default', 'yes', 'no']
- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

9.13 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 (9.1)

Usage:

champ_post_morceau_equation *str*
Read *str* {

```

type str
[ numero int]
[ unite str]
option str into ['stabilite', 'flux_bords', 'flux_surfacique_bords']
[ compo int]
[ source champ_generique_base]
[ sources listchamp_generique]
[ nom_source str]
[ source_reference str]
[ sources_reference list_nom_virgule]
}
where

```

- **type** *str*: can only be operateur for equation operators.

- **numero** *int*: numero will be 0 (diffusive operator) or 1 (convective operator) or 2 (gradient operator) or 3 (divergence operator).
- **unite** *str*: will specify the field unit
- **option** *str* into ['stabilite', 'flux_bords', 'flux_surfacique_bords']: option is stability for time steps or flux_bords for boundary fluxes or flux_surfacique_bords for boundary surfacic fluxes
- **compo** *int*: compo will specify the number component of the boundary flux (for boundary fluxes, in this case compo permits to specify the number component of the boundary flux choosen).
- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

9.14 Moyenne

Synonymous: **champ_post_statistiques_moyenne**

Description: to calculate the average of the field over time

See also: **champ_post_statistiques_base** (9.6)

Usage:

moyenne *str*

Read *str* {

```
[ moyenne_convergee champ_base]
t_deb float
t_fin float
[ source champ_generique_base]
[ sources listchamp_generique]
[ nom_source str]
[ source_reference str]
[ sources_reference list_nom_virgule]
```

}

where

- **moyenne_convergee** *champ_base* (16.1): This option allows to read a converged time averaged field in a .xyz file in order to calculate, when resuming the calculation, the statistics fields (rms, correlation) which depend on this average. In that case, the time averaged field is not updated during the resume of calculation. In this case, the time averaged field must be fully converged to avoid errors when calculating high order statistics.
- **t_deb** *float* for inheritance: Start of integration time
- **t_fin** *float* for inheritance: End of integration time
- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

9.15 Predefini

Description: This keyword is used to post process predefined postprocessing fields.

See also: `champ_generique_base` (9)

Usage:

predefini *str*

Read *str* {

pb_champ *deuxmots*

}

where

- **pb_champ** *deuxmots* (5.34): { `Pb_champ` `nom_pb` `nom_champ` } : `nom_pb` is the problem name and `nom_champ` is the selected field name. The available keywords for the field name are: `energie_cinetique_totale`, `energie_cinetique_elem`, `viscosite_turbulente`, `viscous_force_x`, `viscous_force_y`, `viscous_force_z`, `pressure_force_x`, `pressure_force_y`, `pressure_force_z`, `total_force_x`, `total_force_y`, `total_force_z`, `viscous_force`, `pressure_force`, `total_force`

9.16 Champ_post_reduction_0d

Synonymous: **reduction_0d**

Description: To calculate the min, max, sum, average, weighted sum, weighted average, weighted sum by porosity, weighted average by porosity, euclidian norm, normalized euclidian norm, L1 norm, L2 norm of a field.

See also: `champ_post_de_champs_post` (9.1)

Usage:

champ_post_reduction_0d *str*

Read *str* {

methode *str* into ['min', 'max', 'moyenne', 'average', 'moyenne_ponderee', 'weighted_average', 'somme', 'sum', 'somme_ponderee', 'weighted_sum', 'somme_ponderee_porosite', 'weighted_sum_porosity', 'euclidian_norm', 'normalized_euclidian_norm', 'L1_norm', 'L2_norm', 'valeur_a_gauche', 'left_value']

 [**source** *champ_generique_base*]

 [**sources** *listchamp_generique*]

 [**nom_source** *str*]

 [**source_reference** *str*]

 [**sources_reference** *list_nom_virgule*]

}

where

- **methode** *str* into ['min', 'max', 'moyenne', 'average', 'moyenne_ponderee', 'weighted_average', 'somme', 'sum', 'somme_ponderee', 'weighted_sum', 'somme_ponderee_porosite', 'weighted_sum_porosity', 'euclidian_norm', 'normalized_euclidian_norm', 'L1_norm', 'L2_norm', 'valeur_a_gauche', 'left_value']: name of the reduction method:
 - min for the minimum value,
 - max for the maximum value,
 - average (or moyenne) for a mean,

- `weighted_average` (or `moyenne_ponderee`) for a mean ponderated by integration volumes, e.g: cell volumes for temperature and pressure in VDF, volumes around faces for velocity and temperature in VEF,
- `sum` (or `somme`) for the sum of all the values of the field,
- `weighted_sum` (or `somme_ponderee`) for a weighted sum (integral),
- `weighted_average_porosity` (or `moyenne_ponderee_porosite`) and `weighted_sum_porosity` (or `somme_ponderee_porosite`) for the mean and sum weighted by the volumes of the elements, only for ELEM localisation,
- `euclidian_norm` for the euclidian norm,
- `normalized_euclidian_norm` for the euclidian norm normalized,
- `L1_norm` for norm L1,
- `L2_norm` for norm L2
- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

9.17 Champ_post_refchamp

Synonymous: **refchamp**

Description: Field of prolem

See also: `champ_generique_base` (9)

Usage:

champ_post_refchamp *str*

Read *str* {

[**nom_source** *str*]
pb_champ *deuxmots*

}

where

- **nom_source** *str*: The alias name for the field
- **pb_champ** *deuxmots* (5.34): { Pb_champ nom_pb nom_champ } : `nom_pb` is the problem name and `nom_champ` is the selected field name.

9.18 Champ_post_tparoi_vef

Synonymous: **tparoi_vef**

Description: This keyword is used to post process (only for VEF discretization) the temperature field with a slight difference on boundaries with Neumann condition where law of the wall is applied on the temperature field. `nom_pb` is the problem name and `field_name` is the selected field name. A keyword (`temperature_physique`) is available to post process this field without using `Definition_champs`.

See also: `champ_post_de_champs_post` (9.1)

Usage:

champ_post_tparoi_vof *str*

Read *str* {

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

}

where

- **source** *champ_generique_base* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post..
{ ... }}
- **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* (9.3) for inheritance

9.19 Champ_post_transformation

Synonymous: **transformation**

Description: To create a field with a transformation using source fields and x, y, z, t. If you use in your datafile source refChamp { Pb_champ pb pression }, the field pression may be used in the expression with the name pression_natif_dom; this latter is the same as pression. If you specify nom_source in refChamp bloc, you should use the alias given to pressure field. This is avail for all equations unknowns in transformation.

See also: champ_post_de_champs_post (9.1)

Usage:

champ_post_transformation *str*

Read *str* {

methode *str* into ['produit_scalaire', 'norme', 'vecteur', 'formule', 'composante']
[**unite** *str*]
[**expression** *n word1 word2 ... wordn*]
[**numero** *int*]
[**localisation** *str*]
[**source** *champ_generique_base*]
[**sources** *listchamp_generique*]
[**nom_source** *str*]
[**source_reference** *str*]
[**sources_reference** *list_nom_virgule*]

}

where

- **methode** *str* into ['produit_scalaire', 'norme', 'vecteur', 'formule', 'composante']: methode 0
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 $N \text{ f1}(x,y,z,t) \text{ fN}(x,y,z,t)$: 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.

- **unite** *str*: will specify the field unit
- **expression** *n word1 word2 ... wordn*: expression 1 see methodes formule and vecteur
- **numero** *int*: numero 1 see methode composante
- **localisation** *str*: localisation 1 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* (9) for inheritance: the source field.
- **sources** *listchamp_generique* (9.2) for inheritance: sources { Champ_Post.... { ... } Champ_Post.. { ... } }
- **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* (9.3) for inheritance

10 chimie

Description: Keyword to describe the chmical reactions

See also: objet_u (40)

Usage:

chimie *str*

Read *str* {

```

    reactions reactions
    [ modele_micro_melange int]
    [ constante_modele_micro_melange float]
    [ espece_en_competition_micro_melange str]

```

}

where

- **reactions** *reactions* (10.1): list of reactions
- **modele_micro_melange** *int*: modele_micro_melange (0 by default)
- **constante_modele_micro_melange** *float*: constante of modele (1 by default)
- **espece_en_competition_micro_melange** *str*: espece in competition in reactions

10.1 Reactions

Description: list of reactions

See also: listobj (38.5)

Usage:

{ object1 , object2 }

list of *reaction* (10.1.1) separeted with ,

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

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

See also: `objet_lecture` (39)

Usage:

```
{
    reactifs str
    produits str
    [ constante_taux_reaction float]
    enthalpie_reaction float
    energie_activation float
    exposant_beta float
    [ coefficients_activites bloc_lecture]
    [ 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
- **enthalpie_reaction** *float*: DH
- **energie_activation** *float*: Ea
- **exposant_beta** *float*: Beta
- **coefficients_activites** *bloc_lecture* (3.60): coefficients of activity (exemple { CH4 1 O2 2 })
- **contre_reaction** *float*: K_inv
- **contre_energie_activation** *float*: c_r_Ea

11 class_generic

Description: `not_set`

See also: `objet_u` (40) `solveur_sys_base` (11.16) `dt_start` (11.8)

Usage:

11.1 Amg

Description: Wrapper for AMG preconditioner-based solver which switch for the best one on CPU/GPU Nvidia/GPU AMD

See also: `solveur_sys_base` (11.16)

Usage:

amg solveur option_solveur
where

- **solveur** *str*
- **option_solveur** *bloc_lecture* (3.60)

11.2 Amgx

Description: Solver via AmgX API

See also: `petsc` ([11.13](#))

Usage:

amgx solveur option_solveur

where

- **solveur** *str*
- **option_solveur** *bloc_lecture* ([3.60](#))

11.3 Cholesky

Description: Cholesky direct method.

See also: `solveur_sys_base` ([11.16](#))

Usage:

cholesky str

Read str {

[impr]
 [quiet]

}

where

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

11.4 Cudss

Description: Solver via cuDSS API

See also: `petsc` ([11.13](#))

Usage:

cuDSS solveur option_solveur

where

- **solveur** *str*
- **option_solveur** *bloc_lecture* ([3.60](#))

11.5 Dt_calc

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

See also: `dt_start` ([11.8](#))

Usage:

dt_calc

11.6 Dt_fixe

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

See also: [dt_start \(11.8\)](#)

Usage:

dt_fixe **value**

where

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

11.7 Dt_min

Description: The first iteration is based on dt_min.

See also: [dt_start \(11.8\)](#)

Usage:

dt_min

11.8 Dt_start

Description: not_set

See also: [class_generic \(11\)](#) [dt_calc \(11.5\)](#) [dt_min \(11.7\)](#) [dt_fixe \(11.6\)](#)

Usage:

dt_start

11.9 Gcp_ns

Description: not_set

See also: [gcp \(11.15\)](#)

Usage:

gcp_ns *str*

Read *str* {

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

}

where

- **solveur0** *solveur_sys_base* (11.16): Solver type.
- **solveur1** *solveur_sys_base* (11.16): Solver type.
- **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.
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gcp.
- **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** *int* for inheritance: to save the matrix in a file.
- **precond** *precond_base* (29) 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.
- **precond_diagonal** for inheritance: Keyword to use diagonal preconditioning.
- **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.

11.10 Gen

Description: not_set

See also: *solveur_sys_base* (11.16)

Usage:

gen *str*

Read *str* {

```

    solv_elem str
    precond precond_base
    [ seuil float ]
    [ impr ]
    [ save_matrix|save_matrice ]
    [ quiet ]
    [ nb_it_max int ]
    [ force ]

```

}

where

- **solv_elem** *str*: To specify a solver among gmres or bicgstab.
- **precond** *precond_base* (29): The only preconditionner that we can specify is ilu.

- **seuil** *float*: Value of the final residue. The solver ceases iterations when the Euclidean residue standard $\|Ax-B\|$ is less than this value. default value 1e-12.
- **impr** : Keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **save_matrix|save_matrice** : To save the matrix in a file.
- **quiet** : To not displaying any outputs of the solver.
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the GEN solver.
- **force** : Keyword to set `ipar[5]=-1` in the GEN solver. This is helpful if you notice that the solver does not perform more than 100 iterations. If this keyword is specified in the datafile, you should provide `nb_it_max`.

11.11 Gmres

Description: Gmres method (for non symmetric matrix).

See also: `solveur_sys_base` ([11.16](#))

Usage:

gmres *str*

Read *str* {

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

}

where

- **impr** : Keyword which may be used to print the convergence.
- **quiet** : To disable printing of information
- **seuil** *float*: Convergence value.
- **diag** : Keyword to use diagonal preconditionner (in place of pilut that is not parallel).
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** *int into [0, 1]*: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.
- **save_matrix|save_matrice** : to save the matrix in a file.
- **dim_espace_krilov** *int*

11.12 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` ([11.16](#))

Usage:

optimal *str*

Read *str* {

```

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

```

- **seuil** *float*: Convergence threshold
- **impr** : To print the convergency of the fastest solver
- **quiet** : To disable printing of information
- **save_matrix|save_matrice** : To save the linear system (A, x, B) into a file
- **frequence_recalc** *int*: To set a time step period (by default, 100) for re-checking the fastest solver
- **nom_fichier_solveur** *str*: To specify the file containing the list of the tested solvers
- **fichier_solveur_non_recre** : To avoid the creation of the file containing the list

11.13 Petsc

Description: Solver via Petsc API

See also: `solveur_sys_base` (11.16) `amgx` (11.2) `petsc_gpu` (11.14) `cuDSS` (11.4)

Usage:

petsc solveur

where

- **solveur** *solveur_petsc_deriv* (33): solver type and options

11.14 Petsc_gpu

Description: GPU solver via Petsc API

See also: `petsc` (11.13)

Usage:

petsc_gpu solveur option_solveur [atol] [rtol]

where

- **solveur** *str*
- **option_solveur** *bloc_lecture* (3.60)
- **atol** *float*: Absolute threshold for convergence (same as `seuil` option)
- **rtol** *float*: Relative threshold for convergence

11.15 Gcp

Description: Preconditioned conjugated gradient.

See also: `solveur_sys_base` (11.16) `gcp_ns` (11.9)

Usage:

```

gcp str
Read str {
    seuil float
    [ nb_it_max int ]
    [ impr ]
    [ quiet ]
    [ save_matrix|save_matrice int ]
    [ precond precond_base ]
    [ precond_nul ]
    [ precond_diagonal ]
    [ optimized ]
}
where

```

- **seuil** *float*: Value of the final residue. The gradient ceases iteration when the Euclidean residue standard $\|Ax-B\|$ is less than this value.
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the Gcp.
- **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** *int*: to save the matrix in a file.
- **precond** *precond_base* (29): 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.
- **precond_diagonal** : Keyword to use diagonal preconditioning.
- **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.

11.16 Solveur_sys_base

Description: Basic class to solve the linear system.

See also: class_generic (11) gen (11.10) petsc (11.13) gcp (11.15) optimal (11.12) cholesky (11.3) gmres (11.11) amg (11.1)

Usage:

12

12.1

Description: Comments in a data file.

See also: `objet_u` (40)

Usage:

comm

where

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

13 condlim_base

Description: Basic class of boundary conditions.

See also: `objet_u` (40) `Paroi_echange_interne_global_impose` (13.2) `Paroi_echange_interne_global_parfait` (13.3) `paroi_echange_global_impose` (13.39) `neumann` (13.28) `paroi_echange_contact_vdf` (13.36) `paroi_echange_contact_correlation_vdf` (13.34) `Paroi_echange_interne_parfait` (13.5) `Paroi_echange_interne_impose` (13.4) `paroi_decalee_robin` (13.32) `dirichlet` (13.10) `paroi_echange_externe_impose` (13.37) `paroi_fixe` (13.40) `Paroi` (13.9) `Neumann_homogene` (13.6) `robin_vdf` (13.46) `paroi_echange_contact_correlation_vdf` (13.35) `periodique` (13.45) `paroi_echange_externe_radiatif` (13.11) `paroi_adiabatique` (13.29) `paroi_contact` (13.30) `frontiere_ouverte_fraction_massique_imposee` (13.16) `paroi_contact_fictif` (13.31) `Neumann-paroi` (13.7) `symetrie` (13.49) `paroi_flux_impose` (13.42)

Usage:

condlim_base

13.1 Echange_couplage_thermique

Description: Thermal coupling boundary condition

See also: `paroi_echange_global_impose` (13.39)

Usage:

Echange_couplage_thermique *str*

Read *str* {

[**temperature_pari** *champ_base*]

[**flux_pari** *champ_base*]

}

where

- **temperature_pari** *champ_base* (16.1): Temperature
- **flux_pari** *champ_base* (16.1): Wall heat flux

13.2 Paroi_echange_interne_global_impose

Description: Internal heat exchange boundary condition with global exchange coefficient.

See also: `condlim_base` (13)

Usage:

Paroi_echange_interne_global_impose h_imp ch

where

- **h_imp** *str*: Global exchange coefficient value. The global exchange coefficient value is expressed in $\text{W.m}^{-2}.\text{K}^{-1}$.
- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.3 Paroi_echange_interne_global_parfait

Description: Internal heat exchange boundary condition with perfect (infinite) exchange coefficient.

See also: [condlim_base \(13\)](#)

Usage:

Paroi_echange_interne_global_parfait

13.4 Paroi_echange_interne_impose

Description: Internal heat exchange boundary condition with exchange coefficient.

See also: [condlim_base \(13\)](#)

Usage:

Paroi_echange_interne_impose h_imp ch

where

- **h_imp** *str*: Exchange coefficient value expressed in $\text{W.m}^{-2}.\text{K}^{-1}$.
- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.5 Paroi_echange_interne_parfait

Description: Internal heat exchange boundary condition with perfect (infinite) exchange coefficient.

See also: [condlim_base \(13\)](#)

Usage:

Paroi_echange_interne_parfait

13.6 Neumann_homogene

Description: Homogeneous neumann boundary condition

See also: [condlim_base \(13\)](#) [Neumann_pari_adiabatique \(13.8\)](#)

Usage:

Neumann_homogene

13.7 Neumann_paro

Description: Neumann boundary condition for mass equation (multiphase problem)

See also: [condlim_base \(13\)](#)

Usage:

Neumann_paro ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.8 Neumann_paro_adiabatique

Description: Adiabatic wall neumann boundary condition

See also: [Neumann_homogene \(13.6\)](#)

Usage:

Neumann_paro_adiabatique

13.9 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 \(13\)](#)

Usage:

Paroi

13.10 Dirichlet

Description: Dirichlet condition at the boundary called bord (edge) : 1). For Navier-Stokes equations, velocity imposed at the boundary; 2). For scalar transport equation, scalar imposed at the boundary.

See also: [condlim_base \(13\)](#) [frontiere_ouverte_vitesse_imposee \(13.26\)](#) [frontiere_ouverte_enthalpie_imposee \(13.25\)](#) [paroi_knudsen_non_negligeable \(13.43\)](#) [paroi_temperature_imposee \(13.44\)](#) [frontiere_ouverte_concentration_imposee \(13.15\)](#) [frontiere_ouverte_alpha_impose \(13.14\)](#) [paroi_defilante \(13.33\)](#) [scalaire_impose_paro \(13.47\)](#)

Usage:

dirichlet

13.11 Paroi_echange_externe_radiatif

Synonymous: **echange_externe_radiatif**

Description: Combines radiative ($\sigma * \epsilon * (T^4 - T_{ext}^4)$) and convective ($h * (T - T_{ext})$) heat transfer boundary conditions, where sigma is the Stefan-Boltzmann constant, epsilon is the emi

See also: [condlim_base \(13\)](#)

Usage:

paroi_echange_externe_radiatif h_imp himpc emissivite emissivitebc t_ext ch temp_unit temp_unit_val

where

- **h_imp** *str* into ['h_imp', 't_ext', 'emissivite']: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ_front_base* (17.1): Boundary field type.
- **emissivite** *str* into ['emissivite', 'h_imp', 't_ext']: Emissivity coefficient value.
- **emissivitebc** *champ_front_base* (17.1): Boundary field type.
- **t_ext** *str* into ['t_ext', 'h_imp', 'emissivite']: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* (17.1): Boundary field type.
- **temp_unit** *str* into ['temperature_unit']: Temperature unit
- **temp_unit_val** *str* into ['kelvin', 'celsius']: Temperature unit

13.12 Entree_temperature_imposee_h

Description: Particular case of class *frontiere_ouverte_temperature_imposee* for enthalpy equation.

See also: *frontiere_ouverte_enthalpie_imposee* (13.25)

Usage:

entree_temperature_imposee_h ch

where

- **ch** *champ_front_base* (17.1): Boundary field type.

13.13 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* (13.28)

Usage:

frontiere_ouverte var_name ch

where

- **var_name** *str* into ['T_ext', 'C_ext', 'Y_ext', 'K_Eps_ext', 'K_Omega_ext', 'Fluctu_Temperature_ext', 'Flux_Chaleur_Turb_ext', 'V2_ext', 'a_ext', 'tau_ext', 'k_ext', 'omega_ext', 'H_ext', 'A_i_ext']: Field name.
- **ch** *champ_front_base* (17.1): Boundary field type.

13.14 Frontiere_ouverte_alpha_impose

Description: Imposed alpha condition at the open boundary.

See also: *dirichlet* (13.10)

Usage:

frontiere_ouverte_alpha_impose ch

where

- **ch** *champ_front_base* (17.1): Boundary field type.

13.15 Frontiere_ouverte_concentration_imposee

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

See also: [dirichlet \(13.10\)](#)

Usage:

frontiere_ouverte_concentration_imposee ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.16 Frontiere_ouverte_fraction_massique_imposee

Description: not_set

See also: [condlim_base \(13\)](#)

Usage:

frontiere_ouverte_fraction_massique_imposee ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.17 Frontiere_ouverte_gradient_pression_impose

Description: Normal imposed pressure gradient condition on the open boundary called bord (edge). This boundary condition may be only used in VDF discretization. The imposed $\partial P/\partial n$ value is expressed in Pa.m-1.

See also: [neumann \(13.28\)](#) [frontiere_ouverte_gradient_pression_impose_vefprep1b \(13.18\)](#)

Usage:

frontiere_ouverte_gradient_pression_impose ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.18 Frontiere_ouverte_gradient_pression_impose_vefprep1b

Description: Keyword for an outlet boundary condition in VEF P1B/P1NC on the gradient of the pressure.

See also: [frontiere_ouverte_gradient_pression_impose \(13.17\)](#)

Usage:

frontiere_ouverte_gradient_pression_impose_vefprep1b ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.19 **Frontiere_ouverte_gradient_pression_libre_vef**

Description: Class for outlet boundary condition in VEF like Orlansky. There is no reference for pressure for these boundary conditions so it is better to add pressure condition (with `Frontiere_ouverte_pression_imposee`) on one or two cells (for symmetry in a channel) of the boundary where Orlansky conditions are imposed.

See also: `neumann` ([13.28](#))

Usage:

frontiere_ouverte_gradient_pression_libre_vef

13.20 **Frontiere_ouverte_gradient_pression_libre_vefprep1b**

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

See also: `neumann` ([13.28](#))

Usage:

frontiere_ouverte_gradient_pression_libre_vefprep1b

13.21 **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` ([13.28](#))

Usage:

frontiere_ouverte_pression_imposee ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.22 **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` ([13.28](#))

Usage:

frontiere_ouverte_pression_imposee_orlansky

13.23 **Frontiere_ouverte_pression_moyenne_imposee**

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

See also: `neumann` ([13.28](#))

Usage:

frontiere_ouverte_pression_moyenne_imposee pext

where

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

13.24 Frontiere_ouverte_rho_u_imposee

Description: This keyword is used to designate a condition of imposed mass rate at an open boundary called bord (edge). The imposed mass rate field at the inlet is vectorial and the imposed velocity values are expressed in kg.s-1. This boundary condition can be used only with the Quasi compressible model.

See also: `frontiere_ouverte_vitesse_imposee_sortie` ([13.27](#))

Usage:

frontiere_ouverte_rho_u_imposee **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.25 Frontiere_ouverte_enthalpie_imposee

Synonymous: **frontiere_ouverte_temperature_imposee**

Description: Imposed temperature condition at the open boundary called bord (edge) (in the case of fluid inlet). This condition must be associated with an imposed inlet velocity condition. The imposed temperature value is expressed in oC or K.

See also: `dirichlet` ([13.10](#)) `entree_temperature_imposee_h` ([13.12](#))

Usage:

frontiere_ouverte_enthalpie_imposee **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.26 Frontiere_ouverte_vitesse_imposee

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

See also: `dirichlet` ([13.10](#)) `frontiere_ouverte_vitesse_imposee_sortie` ([13.27](#))

Usage:

frontiere_ouverte_vitesse_imposee **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.27 **Frontiere_ouverte_vitesse_imposee_sortie**

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

See also: `frontiere_ouverte_vitesse_imposee` (13.26) `frontiere_ouverte_rho_u_imposee` (13.24)

Usage:

frontiere_ouverte_vitesse_imposee_sortie **ch**

where

- **ch** `champ_front_base` (17.1): Boundary field type.

13.28 **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` (13) `frontiere_ouverte_pression_imposee_orlansky` (13.22) `frontiere_ouverte_gradient_pression_imposee` (13.17) `sortie_libre_temperature_imposee_h` (13.48) `frontiere_ouverte_pression_imposee` (13.21) `frontiere_ouverte` (13.13) `frontiere_ouverte_pression_moyenne_imposee` (13.23) `frontiere_ouverte_gradient_pression_libre_vefprep1b` (13.20) `frontiere_ouverte_gradient_pression_libre_vef` (13.19)

Usage:

neumann

13.29 **Paroi_adiabatique**

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

See also: `condlim_base` (13)

Usage:

paroi_adiabatique

13.30 **Paroi_contact**

Description: Thermal condition between two domains. Important: the name of the boundaries in the two domains should be the same. (Warning: there is also an old limitation not yet fixed on the sequential algorithm in VDF to detect the matching faces on the two boundaries: faces should be ordered in the same way). The kind of condition depends on the discretization. In VDF, it is a heat exchange condition, and in VEF, a temperature condition.

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

In 3D, for a first preliminary cut domain with HOMARD (fluid for instance), the second problem associated to `pb_name` (solide in a fluid/solid coupling problem) has to be submitted to HOMARD cutting procedure with `connectivity_failed_pb_name.med`.

Such a procedure works as while the primary refined mesh (fluid in our example) impacts the fluid/solid interface with a compact shape as described below (values 2 or 4 indicates the number of division from primary faces obtained in fluid domain at the interface after HOMARD cutting):

2-2-2-2-2-2

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

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

2-2-2-2-2 2-2

OK

2-2 2-2-2

2-4-2 2-2

2-2 2-2

NOT OK

See also: `condlim_base` ([13](#))

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

13.31 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` ([13](#))

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

13.32 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` ([13](#))

Usage:

paroi_decalee_robin *str*

Read *str* {

delta *float*

}

where

- **delta** *float*

13.33 Paroi_defilante

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

See also: [dirichlet \(13.10\)](#)

Usage:

paroi_defilante **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.34 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 \(13\)](#)

Usage:

paroi_echange_contact_correlation_vdf *str*

Read *str* {

```
[ dir int]  
[ tin float]  
[ tsup float]  
[ lambda str]  
[ rho str]  
[ dt_impr float]  
[ cp 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.
- **dt_impr** *float*: Printing period in name_of_data_file_time.dat files of the 1D model results.
- **cp** *float*: Calorific capacity value at a constant pressure of the fluid (J.kg-1.K-1).
- **mu** *str*: Dynamic viscosity of the fluid (kg.m-1.s-1) which may be a function of the temperature T.

- **debit** *float*: Surface flow rate (kg.s-1.m-2) of the fluid into the channel.
- **dh** *float*: Hydraulic diameter may be a function f(x) with x position along the 1D axis ($x_{inf} \leq x \leq x_{sup}$)
- **volume** *str*: Exact volume of the 1D domain (m3) which may be a function of the hydraulic diameter (Dh) and the lateral surface (S) of the meshed boundary.
- **nu** *str*: Nusselt number which may be a function of the Reynolds number (Re) and the Prandtl number (Pr).
- **reprise_correlation** : Keyword in the case of a resuming calculation with this correlation.

13.35 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 \(13\)](#)

Usage:

paroi_echange_contact_correlation_vef *str*

Read *str* {

```
[ dir int]
[ tin float]
[ tsup float]
[ lambda str]
[ rho str]
[ dt_impr float]
[ cp float]
[ mu str]
[ debit float]
[ n int]
[ dh str]
[ surface str]
[ xinf float]
[ xsup float]
[ nu str]
[ 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.
- **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.
- **dt_impr** *float*: Printing period in name_of_data_file_time.dat files of the 1D model results.
- **cp** *float*: Calorific capacity value at a constant pressure of the fluid (J.kg-1.K-1).
- **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.
- **n** *int*: Number of 1D cells of the 1D mesh.

- **dh** *str*: Hydraulic diameter may be a function $f(x)$ with x position along the 1D axis ($x_{inf} \leq x \leq x_{sup}$)
- **surface** *str*: Section surface of the channel which may be function $f(Dh, x)$ of the hydraulic diameter (Dh) and x position along the 1D axis ($x_{inf} \leq x \leq x_{sup}$)
- **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.
- **nu** *str*: Nusselt number which may be a function of the Reynolds number (Re) and the Prandtl number (Pr).
- **emissivite_pour_rayonnement_entre_deux_plaques_quasi_infinies** *float*: Coefficient of emissivity for radiation between two quasi infinite plates.
- **reprise_correlation** : Keyword in the case of a resuming calculation with this correlation.

13.36 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` (13)

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 $W.K^{-1}m^{-2}$) 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 :
$$q = h (T_1 - T_2)$$
 where $1/h = d_1/\lambda_1 + 1/val_h_contact + d_2/\lambda_2$
where d_i : distance between the node where T_i and the wall is found.

13.37 Paroi_echange_externe_impose

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

See also: `condlim_base` (13) `paroi_echange_externe_impose_h` (13.38)

Usage:

paroi_echange_externe_impose **h_or_t** **himpc** **t_or_h** **ch**

where

- **h_or_t** *str* into ['h_imp', 't_ext']: Heat exchange coefficient value (expressed in $W.m^{-2}.K^{-1}$).
- **himpc** *champ_front_base* (17.1): Boundary field type.
- **t_or_h** *str* into ['t_ext', 'h_imp']: External temperature value (expressed in $^{\circ}C$ or K).
- **ch** *champ_front_base* (17.1): Boundary field type.

13.38 Paroi_echange_externe_impose_h

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

See also: `paroi_echange_externe_impose` ([13.37](#))

Usage:

paroi_echange_externe_impose_h h_or_t himpc t_or_h ch

where

- **h_or_t** *str* into ['h_imp', 't_ext']: Heat exchange coefficient value (expressed in W.m-2.K-1).
- **himpc** *champ_front_base* ([17.1](#)): Boundary field type.
- **t_or_h** *str* into ['t_ext', 'h_imp']: External temperature value (expressed in oC or K).
- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.39 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` ([13](#)) `Echange_couplage_thermique` ([13.1](#))

Usage:

paroi_echange_global_impose h_imp himpc text ch

where

- **h_imp** *str*: Global exchange coefficient value. The global exchange coefficient value is expressed in W.m-2.K-1.
- **himpc** *champ_front_base* ([17.1](#)): Boundary field type.
- **text** *str*: External temperature value. The external temperature value is expressed in oC or K.
- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.40 Paroi_fixe

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

See also: `condlim_base` ([13](#)) `paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets` ([13.41](#))

Usage:

paroi_fixe

13.41 Paroi_fixe_iso_genepi2_sans_contribution_aux_vitesses_sommets

Description: Boundary condition to obtain iso Geneppi2, without interest

See also: `paroi_fixe` ([13.40](#))

Usage:

paroi_fixe_iso_Genepi2_sans_contribution_aux_vitesses_sommets

13.42 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` ([13](#))

Usage:

paroi_flux_impose ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.43 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.

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

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` ([13.10](#))

Usage:

paroi_knudsen_non_negligeable name_champ_1 champ_1 name_champ_2 champ_2

where

- **name_champ_1** *str into ['vitesse_paro', 'k']*: Field name.
- **champ_1** *champ_front_base* ([17.1](#)): Boundary field type.
- **name_champ_2** *str into ['vitesse_paro', 'k']*: Field name.
- **champ_2** *champ_front_base* ([17.1](#)): Boundary field type.

13.44 Paroi_temperature_imposee

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

See also: `dirichlet` ([13.10](#)) `enthalpie_imposee_paro` ([13.50](#))

Usage:

paroi_temperature_imposee ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.45 Periodique

Description: 1). For Navier-Stokes equations, this keyword is used to indicate that the horizontal inlet velocity values are the same as the outlet velocity values, at every moment. As regards meshing, the inlet and outlet edges bear the same name.; 2). For scalar transport equation, this keyword is used to set a periodic condition on scalar. The two edges dealing with this periodic condition bear the same name.

See also: `condlim_base` ([13](#))

Usage:

periodique

13.46 Robin_vef

Description: Robin condition at the boundary (edge)

See also: `condlim_base` ([13](#))

Usage:

robin_vef *str*

Read *str* {

alpha *float*

beta *float*

champ_front_normal_et_tangentiel_robin *champ_front_base*

}

where

- **alpha** *float*: Robin coefficient for the normal field
- **beta** *float*: Robin coefficient for the tangent field
- **champ_front_normal_et_tangentiel_robin** *champ_front_base* ([17.1](#)): The boundary field

13.47 Scalaire_impose_paro

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

See also: `dirichlet` ([13.10](#))

Usage:

scalaire_impose_paro **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.48 Sortie_libre_temperature_imposee_h

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

See also: `neumann` ([13.28](#))

Usage:

sortie_libre_temperature_imposee_h **ch**

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

13.49 Symetrie

Description: 1). For Navier-Stokes equations, this keyword is used to designate a symmetry condition concerning the velocity at the boundary called bord (edge) (normal velocity at the edge equal to zero and tangential velocity gradient at the edge equal to zero); 2). For scalar transport equation, this keyword is used to set a symmetry condition on scalar on the boundary named bord (edge).

See also: [condlim_base \(13\)](#)

Usage:

symetrie

13.50 Enthalpie_imposee_paro

Synonymous: **temperature_imposee_paro**

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

See also: [paroi_temperature_imposee \(13.44\)](#)

Usage:

enthalpie_imposee_paro ch

where

- **ch** *champ_front_base* ([17.1](#)): Boundary field type.

14 discretisation_base

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

See also: [objet_u \(40\)](#) [vdf \(14.8\)](#) [polymac \(14.5\)](#) [polymac_POPINC \(14.6\)](#) [polymac_p0 \(14.7\)](#) [DG \(14.1\)](#) [vef \(14.9\)](#) [ijk \(14.4\)](#) [EF_axi \(14.2\)](#) [ef \(14.3\)](#)

Usage:

14.1 Dg

Description: DG discretization

See also: [discretisation_base \(14\)](#)

Usage:

14.2 Ef_axi

Description: Element Finite discretization.

See also: [discretisation_base \(14\)](#)

Usage:

14.3 Ef

Description: Element Finite discretization.

See also: [discretisation_base \(14\)](#)

Usage:

14.4 Ijk

Description: IJK discretization.

See also: [discretisation_base \(14\)](#)

Usage:

14.5 Polymac

Description: polymac discretization (polymac discretization that is not compatible with pb_multi).

See also: [discretisation_base \(14\)](#)

Usage:

14.6 Polymac_p0p1nc

Description: polymac_P0P1NC discretization (previously polymac discretization compatible with pb_multi).

See also: [discretisation_base \(14\)](#)

Usage:

14.7 Polymac_p0

Description: polymac_p0 discretization (previously covimac discretization compatible with pb_multi).

See also: [discretisation_base \(14\)](#)

Usage:

14.8 Vdf

Description: Finite difference volume discretization.

See also: [discretisation_base \(14\)](#)

Usage:

14.9 Vef

Synonymous: **vefprep1b**

Description: Finite element volume discretization (P1NC/P1-bubble element). Since the 1.5.5 version, several new discretizations are available thanks to the optional keyword Read. By default, the VEFPreP1B

keyword is equivalent to the former VEFPreP1B formulation (v1.5.4 and sooner). P0P1 (if used with the strong formulation for imposed pressure boundary) is equivalent to VEFPreP1B but the convergence is slower. VEFPreP1B dis is equivalent to VEFPreP1B dis Read dis { P0 P1 Changement_de_base_P1Bulle 1 Cl_pression_sommet_faible 0 }

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

Usage:

vef *str*

Read *str* {

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

}

where

- **changement_de_base_p1bulle** *int into [0, 1]*: changement_de_base_p1bulle 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).
- **p0** : Pressure nodes are added on element centres
- **p1** : Pressure nodes are added on vertices
- **pa** : Only available in 3D, pressure nodes are added on bones
- **rt** : For P1NCP1B (in TrioCFD)
- **modif_div_face_dirichlet** *int into [0, 1]*: This option (by default 0) is used to extend control volumes for the momentum equation.
- **cl_pression_sommet_faible** *int into [0, 1]*: This option is used to specify a strong formulation (value set to 0, the default) or a weak formulation (value set to 1) for an imposed pressure boundary condition. The first formulation converges quicker and is stable in general cases. The second formulation should be used if there are several outlet boundaries with Neumann condition (see Ecoulement_Neumann test case for example).

15 domaine

Description: Keyword to create a domain.

See also: objet_u ([40](#)) DomaineAxi1d ([15.1](#)) IJK_Grid_Geometry ([15.2](#))

Usage:

15.1 Domaineaxi1d

Description: 1D domain

See also: domaine ([15](#))

Usage:

15.2 Ijk_grid_geometry

Description: Object to define the grid that will represent the domain of the simulation in IJK discretization

See also: [domaine \(15\)](#)

Usage:

IJK_Grid_Geometry *str*

```
Read str {  
    [ perio_i ]  
    [ perio_j ]  
    [ perio_k ]  
    [ nbelem_i int]  
    [ nbelem_j int]  
    [ nbelem_k int]  
    [ uniform_domain_size_i float]  
    [ uniform_domain_size_j float]  
    [ uniform_domain_size_k float]  
    [ origin_i float]  
    [ origin_j float]  
    [ origin_k float]  
}
```

where

- **perio_i** : rien to specify the border along the I direction is periodic
- **perio_j** : rien to specify the border along the J direction is periodic
- **perio_k** : rien to specify the border along the K direction is periodic
- **nbelem_i** *int*: the number of elements of the grid in the I direction
- **nbelem_j** *int*: the number of elements of the grid in the J direction
- **nbelem_k** *int*: the number of elements of the grid in the K direction
- **uniform_domain_size_i** *float*: the size of the elements along the I direction
- **uniform_domain_size_j** *float*: the size of the elements along the J direction
- **uniform_domain_size_k** *float*: the size of the elements along the K direction
- **origin_i** *float*: I-coordinate of the origin of the grid
- **origin_j** *float*: J-coordinate of the origin of the grid
- **origin_k** *float*: K-coordinate of the origin of the grid

16 champ_base

16.1 Champ_base

Description: Basic class of fields.

See also: [objet_u \(40\)](#) [champ_don_base \(16.9\)](#) [champ_ostwald \(16.25\)](#) [champ_fonc_med \(16.14\)](#) [champ_input_base \(16.21\)](#)

Usage:

16.2 Champ_fonc_interp

Description: Field that is interpolated from a distant domain via MEDCoupling (remapper).

See also: [champ_don_base \(16.9\)](#)

Usage:

Champ_Fonc_Interp *str*

```
Read str {  
  
    nom_champ str  
    pb_loc str  
    pb_dist str  
    [ dom_loc str]  
    [ dom_dist str]  
    [ default_value str]  
    nature str  
    [ use_overlapdec str]  
  
}
```

where

- **nom_champ** *str*: Name of the field (for example: temperature).
- **pb_loc** *str*: Name of the local problem.
- **pb_dist** *str*: Name of the distant problem.
- **dom_loc** *str*: Name of the local domain.
- **dom_dist** *str*: Name of the distant domain.
- **default_value** *str*: Name of the distant domain.
- **nature** *str*: Nature of the field (knowledge from MEDCoupling is required; IntensiveMaximum, IntensiveConservation, ...).
- **use_overlapdec** *str*: Nature of the field (knowledge from MEDCoupling is required; IntensiveMaximum, IntensiveConservation, ...).

16.3 Champ_fonc_med_table_temps

Description: Field defined as a fixed spatial shape scaled by a temporal coefficient

See also: `champ_fonc_med` ([16.14](#))

Usage:

Champ_Fonc_MED_Table_Temps *str*

```
Read str {  
  
    [ table_temps bloc_lecture]  
    [ table_temps_lue str]  
    [ use_existing_domain ]  
    [ last_time ]  
    [ decoup str]  
    [ mesh str]  
    domain str  
    file str  
    field str  
    [ loc str into [ 'som', 'elem' ]]  
    [ time float]  
  
}
```

where

- **table_temps** *bloc_lecture* ([3.60](#)): Table containing the temporal coefficient used to scale the field

- **table_temps_lue** *str*: Name of the file containing the values of the temporal coefficient used to scale the field
- **use_existing_domain** for inheritance: whether to optimize the field loading by indicating that the field is supported by the same mesh that was initially loaded as the domain
- **last_time** for inheritance: to use the last time of the MED file instead of the specified time. Mutually exclusive with 'time' parameter.
- **decoup** *str* for inheritance: specify a partition file.
- **mesh** *str* for inheritance: Name of the mesh supporting the field. This is the name of the mesh in the MED file, and if this mesh was also used to create the TRUST domain, loading can be optimized with option 'use_existing_domain'.
- **domain** *str* for inheritance: Name of the domain supporting the field. This is the name of the mesh in the MED file, and if this mesh was also used to create the TRUST domain, loading can be optimized with option 'use_existing_domain'.
- **file** *str* for inheritance: Name of the .med file.
- **field** *str* for inheritance: Name of field to load.
- **loc** *str* into ['som', 'elem'] for inheritance: To indicate where the field is localised. Default to 'elem'.
- **time** *float* for inheritance: Timestep to load from the MED file. Mutually exclusive with 'last_time' flag.

16.4 Champ_fonc_med_tabule

Description: not_set

See also: champ_fonc_med ([16.14](#))

Usage:

Champ_Fonc_MED_Tabule *str*

Read *str* {

```

    [ use_existing_domain ]
    [ last_time ]
    [ decoup str ]
    [ mesh str ]
    domain str
    file str
    field str
    [ loc str into ['som', 'elem']]
    [ time float ]

```

}

where

- **use_existing_domain** for inheritance: whether to optimize the field loading by indicating that the field is supported by the same mesh that was initially loaded as the domain
- **last_time** for inheritance: to use the last time of the MED file instead of the specified time. Mutually exclusive with 'time' parameter.
- **decoup** *str* for inheritance: specify a partition file.
- **mesh** *str* for inheritance: Name of the mesh supporting the field. This is the name of the mesh in the MED file, and if this mesh was also used to create the TRUST domain, loading can be optimized with option 'use_existing_domain'.
- **domain** *str* for inheritance: Name of the domain supporting the field. This is the name of the mesh in the MED file, and if this mesh was also used to create the TRUST domain, loading can be optimized with option 'use_existing_domain'.

- **file** *str* for inheritance: Name of the .med file.
- **field** *str* for inheritance: Name of field to load.
- **loc** *str into* [*'som'*, *'elem'*] for inheritance: To indicate where the field is localised. Default to *'elem'*.
- **time** *float* for inheritance: Timestep to load from the MED file. Mutually exclusive with *'last_time'* flag.

16.5 Champ_tabule_morceaux

Description: Field defined by tabulated data in each sub-domaine. It makes possible the definition of a field which is a function of other fields.

See also: `champ_don_base` ([16.9](#)) `Champ_Fonc_Tabule_Morceaux_Interp` ([16.6](#))

Usage:

Champ_Tabule_Morceaux **domain_name** **nb_comp** **data**

where

- **domain_name** *str*: Name of the domain.
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* ([3.60](#)): { Defaut *val_def* *sous_domaine_1 val_1 ... sous_domaine_i val_i* } By default, the value *val_def* is assigned to the field. It takes the *sous_domaine_i* identifier *Sous_Domaine* (*sub_area*) type object function, *val_i*. *Sous_Domaine* (*sub_area*) type objects must have been previously defined if the operator wishes to use a `champ_fonc_tabule_morceaux` type object.

16.6 Champ_fonc_tabule_morceaux_interp

Description: Field defined by tabulated data in each sub-domaine. It makes possible the definition of a field which is a function of other fields. Here we use MEDCoupling to interpolate fields between the two domains.

See also: `Champ_Tabule_Morceaux` ([16.5](#))

Usage:

Champ_Fonc_Tabule_Morceaux_Interp **problem_name** **nb_comp** **data**

where

- **problem_name** *str*: Name of the problem.
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* ([3.60](#)): { Defaut *val_def* *sous_domaine_1 val_1 ... sous_domaine_i val_i* } By default, the value *val_def* is assigned to the field. It takes the *sous_domaine_i* identifier *Sous_Domaine* (*sub_area*) type object function, *val_i*. *Sous_Domaine* (*sub_area*) type objects must have been previously defined if the operator wishes to use a `champ_fonc_tabule_morceaux` type object.

16.7 Champ_parametrique

Description: Parametric field

See also: `champ_don_base` ([16.9](#))

Usage:

Champ_Parametrique *str*

Read *str* {

```

    fichier str
}
where

```

- **fichier** *str*: Filename where fields are read

16.8 Champ_composite

Description: Composite field. Used in multiphase problems to associate data to each phase.

See also: champ_don_base (16.9) champ_musig (16.24)

Usage:

```

champ_composite dim bloc
where

```

- **dim** *int*: Number of field components.
- **bloc** *bloc_lecture* (3.60): Values Various pieces of the field, defined per phase. Part 1 goes to phase 1, etc...

16.9 Champ_don_base

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

See also: champ_base (16.1) champ_som_lu_vdf (16.26) champ_som_lu_vef (16.27) champ_fonc_tabule (16.18) champ_tabule_temps (16.29) champ_uniforme_morceaux (16.30) champ_fonc_t (16.17) tayl_green (16.35) champ_don_lu (16.10) Champ_Tabule_Morceaux (16.5) champ_init_canal_sinal (16.19) init_partie (16.34) uniform_field (16.36) champ_composite (16.8) champ_fonc_txyz (16.32) champ_fonc_xyz (16.33) champ_fonc_fonction_txyz_morceaux (16.13) champ_fonc_reprise (16.15) Champ_Parametrique (16.7) Champ_Fonc_Interp (16.2)

Usage:

16.10 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 (16.9)

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

16.11 Champ_fonc_fonction

Description: Field that is a function of another field.

See also: `champ_fonc_tabule` (16.18) `champ_fonc_fonction_txyz` (16.12)

Usage:

champ_fonc_fonction **problem_name** **inco** **expression**

where

- **problem_name** *str*: Name of problem.
- **inco** *str*: Name of the field (for example: temperature).
- **expression** *n word1 word2 ... wordn*: Number of field components followed by the analytical expression for each field component.

16.12 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` (16.11)

Usage:

champ_fonc_fonction_txyz **problem_name** **inco** **expression**

where

- **problem_name** *str*: Name of problem.
- **inco** *str*: Name of the field (for example: temperature).
- **expression** *n word1 word2 ... wordn*: Number of field components followed by the analytical expression for each field component.

16.13 Champ_fonc_fonction_txyz_morceaux

Description: Field defined by analytical functions in each sub-domaine. On each zone, the value is defined as a function of x,y,z,t and of scalar value taken from a parameter field. This values is associated to the variable 'val' in the expression.

See also: `champ_don_base` (16.9)

Usage:

champ_fonc_fonction_txyz_morceaux **problem_name** **inco** **nb_comp** **data**

where

- **problem_name** *str*: Name of the problem.
- **inco** *str*: Name of the field (for example: temperature).
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* (3.60): { Defaut val_def sous_domaine_1 val_1 ... sous_domaine_i val_i } By default, the value val_def is assigned to the field. It takes the sous_domaine_i identifier Sous_Domaine (sub_area) type object function, val_i. Sous_Domaine (sub_area) type objects must have been previously defined if the operator wishes to use a `champ_fonc_fonction_txyz_morceaux` type object.

16.14 Champ_fonc_med

Description: Field to read a data field in a MED-format file .med at a specified time. It is very useful, for example, to resume a calculation with a new or refined geometry. The field post-processed on the new geometry at med format is used as initial condition for the resume.

See also: [champ_base \(16.1\)](#) [Champ_Fonc_MED_Table_Temps \(16.3\)](#) [Champ_Fonc_MED_Tabule \(16.4\)](#)

Usage:

champ_fonc_med *str*

Read *str* {

```
[ use_existing_domain ]
[ last_time ]
[ decoup str]
[ mesh str]
domain str
file str
field str
[ loc str into ['som', 'elem']]
[ time float]
```

}

where

- **use_existing_domain** : whether to optimize the field loading by indicating that the field is supported by the same mesh that was initially loaded as the domain
- **last_time** : to use the last time of the MED file instead of the specified time. Mutually exclusive with 'time' parameter.
- **decoup *str***: specify a partition file.
- **mesh *str***: Name of the mesh supporting the field. This is the name of the mesh in the MED file, and if this mesh was also used to create the TRUST domain, loading can be optimized with option 'use_existing_domain'.
- **domain *str***: Name of the domain supporting the field. This is the name of the mesh in the MED file, and if this mesh was also used to create the TRUST domain, loading can be optimized with option 'use_existing_domain'.
- **file *str***: Name of the .med file.
- **field *str***: Name of field to load.
- **loc *str* into ['som', 'elem']**: To indicate where the field is localised. Default to 'elem'.
- **time *float***: Timestep to load from the MED file. Mutually exclusive with 'last_time' flag.

16.15 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 \(16.9\)](#)

Usage:

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

where

- **format *str* into ['binaire', 'formatte', 'xyz', 'single_hdf', 'pdi']**: 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. If pdi is used, the same constraints/advantages as binaire apply, but it produces one (HDF5) file per node on the filesystem instead of having one file per processor. The single_hdf format is still supported but is obsolete, the PDI format is recommended.

- **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* (16.16): 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.

16.16 Fonction_champ_reprise

Description: *not_set*

See also: *objet_lecture* (39)

Usage:

mot fonction
where

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

16.17 Champ_fonc_t

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

See also: *champ_don_base* (16.9)

Usage:

champ_fonc_t val
where

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

16.18 Champ_fonc_tabule

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

See also: *champ_don_base* (16.9) *champ_fonc_fonction* (16.11)

Usage:

champ_fonc_tabule pb_field dim bloc
where

- **pb_field** *bloc_lecture* (3.60): block similar to { *pb1 field1* } or { *pb1 field1 ... pbN fieldN* }
- **dim** *int*: Number of field components.

- **bloc** *bloc_lecture* (3.60): 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)).

16.19 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* (16.9)

Usage:

champ_init_canal_sinal *dim bloc*

where

- **dim** *int*: Number of field components.
- **bloc** *bloc_lec_champ_init_canal_sinal* (16.20): Parameters for the class *champ_init_canal_sinal*.

16.20 Bloc_lec_champ_init_canal_sinal

Description: Parameters for the class *champ_init_canal_sinal*.

in 2D:

$U = ucent * y(2h - y) / h / h$

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

rand: unpredictable value between -1 and 1.

in 3D:

$U = ucent * y(2h - y) / h / h$

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

$W = ampli_bruit * rand2$

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

See also: *objet_lecture* (39)

Usage:

```
{
    ucent float
    h float
    ampli_bruit float
    [ ampli_sin float]
    omega float
    [ dir_flow int into [0, 1, 2]]
    [ dir_wall int into [0, 1, 2]]
    [ min_dir_flow float]
    [ min_dir_wall float]
}
```

where

- **ucent** *float*: Velocity value at the center of the channel.
- **h** *float*: Half length of the channel.
- **ampli_bruit** *float*: Amplitude for the disturbance.
- **ampli_sin** *float*: Amplitude for the sinusoidal disturbance (by default equals to ucent/10).
- **omega** *float*: Value of pulsation for the of the sinusoidal disturbance.

- **dir_flow** *int into [0, 1, 2]*: Flow direction for the initialization of the flow in a channel.
 - if dir_flow=0, the flow direction is X
 - if dir_flow=1, the flow direction is Y
 - if dir_flow=2, the flow direction is Z
 Default value for dir_flow is 0
- **dir_wall** *int into [0, 1, 2]*: Wall direction for the initialization of the flow in a channel.
 - if dir_wall=0, the normal to the wall is in X direction
 - if dir_wall=1, the normal to the wall is in Y direction
 - if dir_wall=2, the normal to the wall is in Z direction
 Default value for dir_flow is 1
- **min_dir_flow** *float*: Value of the minimum coordinate in the flow direction for the initialization of the flow in a channel. Default value for dir_flow is 0.
- **min_dir_wall** *float*: Value of the minimum coordinate in the wall direction for the initialization of the flow in a channel. Default value for dir_flow is 0.

16.21 Champ_input_base

Description: not_set

See also: champ_base (16.1) champ_input_p0 (16.22) champ_input_p0_composite (16.23)

Usage:

champ_input_base *str*

```
Read str {
    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*

16.22 Champ_input_p0

Description: not_set

See also: champ_input_base (16.21)

Usage:

champ_input_p0 *str*

```
Read str {
    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

16.23 Champ_input_p0_composite

Description: Field used to define a classical champ input p0 field (for ICoCo), but with a predefined field for the initial state.

See also: `champ_input_base` ([16.21](#))

Usage:

champ_input_p0_composite *str*

Read *str* {

```

    [ initial_field  champ_base]
    [ input_field  champ_input_p0]
    nb_comp  int
    nom  str
    [ initial_value  n x1 x2 ... xn]
    probleme  str
    [ sous_zone  str]

```

```

}
where

```

- **initial_field** *champ_base* ([16.1](#)): The field used for initialization
- **input_field** *champ_input_p0* ([16.22](#)): The input field for ICoCo
- **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

16.24 Champ_musig

Description: MUSIG field. Used in multiphase problems to associate data to each phase.

See also: `champ_composite` ([16.8](#))

Usage:

champ_musig **bloc**

where

- **bloc** *bloc_lecture* ([3.60](#)): Not set

16.25 Champ_ostwald

Description: This keyword is used to define the viscosity variation law:

$$\mu(T) = K(T) \cdot (D:D/2)^{((n-1)/2)}$$

See also: `champ_base` ([16.1](#))

Usage:

champ_ostwald

16.26 Champ_som_lu_vdf

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

See also: `champ_don_base` ([16.9](#))

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 ...

16.27 Champ_som_lu_vef

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

See also: `champ_don_base` ([16.9](#))

Usage:

champ_som_lu_vef domain_name dim tolerance file

where

- **domain_name** *str*: Name of the domain.
- **dim** *int*: Value of the dimension of the field.
- **tolerance** *float*: Value of the tolerance to check the coordinates of the nodes.
- **file** *str*: Name of the file.

This file has the following format:

Xi Yi Zi -> Coordinates of the node

Ui Vi Wi -> Value of the field on this node

Xi+1 Yi+1 Zi+1 -> Next point

Ui+1 Vi+1 Zi+1 -> Next value ...

16.28 Champ_tabule_lu

Description: Uniform field, tabulated from a specified column file. Lines starting with # are ignored.

See also: champ_tabule_temps ([16.29](#))

Usage:

champ_tabule_lu nb_comp column_file dim

where

- **nb_comp** *int*: Number of field components.
- **column_file** *str*: Name of the column file.
- **dim** *int*: Number of field components.

16.29 Champ_tabule_temps

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

See also: champ_don_base ([16.9](#)) champ_tabule_lu ([16.28](#))

Usage:

champ_tabule_temps dim bloc

where

- **dim** *int*: Number of field components.
- **bloc** *bloc_lecture* ([3.60](#)): Values as a table. The value of the field at any time is calculated by linear interpolation from this table.

16.30 Champ_uniforme_morceaux

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

See also: champ_don_base ([16.9](#)) valeur_totale_sur_volume ([16.37](#)) champ_uniforme_morceaux_tabule_temps ([16.31](#))

Usage:

champ_uniforme_morceaux nom_dom nb_comp data

where

- **nom_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* ([3.60](#)): { Defaut val_def sous_zone_1 val_1 ... sous_zone_i val_i } By default, the value val_def is assigned to the field. It takes the sous_zone_i identifier Sous_Zone (sub_area) type object value, val_i. Sous_Zone (sub_area) type objects must have been previously defined if the operator wishes to use a Champ_Uniforme_Morceaux(partly_uniform_field) type object.

16.31 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 ([16.30](#))

Usage:

champ_uniforme_morceaux_tabule_temps **nom_dom** **nb_comp** **data**

where

- **nom_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* (3.60): { Defaut val_def sous_zone_1 val_1 ... sous_zone_i val_i } By default, the value val_def is assigned to the field. It takes the sous_zone_i identifier Sous_Zone (sub_area) type object value, val_i. Sous_Zone (sub_area) type objects must have been previously defined if the operator wishes to use a Champ_Uniforme_Morceaux(partly_uniform_field) type object.

16.32 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 (16.9)

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).

16.33 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 (16.9)

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).

16.34 Init_par_partie

Description: ne marche que pour n_comp=1

See also: champ_don_base (16.9)

Usage:

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

where

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

16.35 Tayl_green

Description: Class Tayl_green.

See also: champ_don_base ([16.9](#))

Usage:

tayl_green dim

where

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

16.36 Uniform_field

Synonymous: **champ_uniforme**

Description: Field that is constant in space and stationary.

See also: champ_don_base ([16.9](#))

Usage:

uniform_field val

where

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

16.37 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 ([16.30](#))

Usage:

valeur_totale_sur_volume nom_dom nb_comp data

where

- **nom_dom** *str*: Name of the domain to which the sub-areas belong.
- **nb_comp** *int*: Number of field components.
- **data** *bloc_lecture* ([3.60](#)): { Default val_def sous_zone_1 val_1 ... sous_zone_i val_i } By default, the value val_def is assigned to the field. It takes the sous_zone_i identifier Sous_Zone (sub_area) type object value, val_i. Sous_Zone (sub_area) type objects must have been previously defined if the operator wishes to use a Champ_Uniforme_Morceaux(partly_uniform_field) type object.

17 champ_front_base

17.1 Champ_front_base

Description: Basic class for fields at domain boundaries.

See also: objet_u ([40](#)) Champ_front_debit_QC_VDF_fonc_t ([17.5](#)) Champ_front_debit_QC_VDF ([17.4](#)) champ_front_pression_from_u ([17.25](#)) champ_front_contact_vef ([17.13](#)) champ_front_tangentiel_vef ([17.29](#))

champ_front_MED (17.9) champ_front_uniforme (17.30) champ_front_fonction (17.21) champ_front_debit-
_massique (17.15) champ_front_tabule (17.27) ch_front_input (17.7) champ_front_debit (17.14) champ-
_front_xyz_debit (17.31) champ_front_lu (17.22) boundary_field_inward (17.6) champ_front_normal_vef
(17.24) champ_front_fonc_pois_tube (17.17) champ_front_bruite (17.10) champ_front_fonc_txyz (17.19)
champ_front_fonc_pois_ipsn (17.16) champ_front_calc (17.11) champ_front_composite (17.12) champ-
_front_fonc_t (17.18) champ_front_fonc_xyz (17.20) champ_front_recyclage (17.26) Champ_front_Parametrique
(17.3)

Usage:

17.2 Champ_front_xyz_tabule

Description: Space dependent field on the boundary, tabulated as a function of time.

See also: champ_front_fonc_txyz (17.19)

Usage:

Champ_Front_xyz_Tabule **val bloc**
where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).
- **bloc** *bloc_lecture* (3.60): {nt1 t2 t3 ...tn u1 [v1 w1 ...] u2 [v2 w2 ...] u3 [v3 w3 ...] ... un [vn wn ...]
}
Values are entered into a table based on n couples (ti, ui) if nb_comp value is 1. The value of a field
at a given time is calculated by linear interpolation from this table.

17.3 Champ_front_parametrique

Description: Parametric boundary field

See also: champ_front_base (17.1)

Usage:

Champ_front_Parametrique *str*
Read *str* {

fichier *str*

}

where

- **fichier** *str*: Filename where boundary fields are read

17.4 Champ_front_debit_qc_vdf

Description: This keyword is used to define a flow rate field for quasi-compressible fluids in VDF discretization. The flow rate is kept constant during a transient.

See also: champ_front_base (17.1)

Usage:

Champ_front_debit_QC_VDF **dimension liste [moyen] pb_name**
where

- **dimension** *int*: Problem dimension
- **liste** *bloc_lecture* (3.60): List of the mass flow rate values [kg/s/m2] with the following syntaxe: { val1 ... valdim }
- **moyen** *str*: Option to use rho mean value
- **pb_name** *str*: Problem name

17.5 Champ_front_debit_qc_vdf_fonc_t

Description: This keyword is used to define a flow rate field for quasi-compressible fluids in VDF discretization. The flow rate could be constant or time-dependent.

See also: `champ_front_base` (17.1)

Usage:

Champ_front_debit_QC_VDF_fonc_t **dimension** **liste** [**moyen**] **pb_name**
where

- **dimension** *int*: Problem dimension
- **liste** *bloc_lecture* (3.60): List of the mass flow rate values [kg/s/m2] with the following syntaxe: { val1 ... valdim } where val1 ... valdim are constant or function of time.
- **moyen** *str*: Option to use rho mean value
- **pb_name** *str*: Problem name

17.6 Boundary_field_inward

Description: this field is used to define the normal vector field standard at the boundary in VDF or VEF discretization.

See also: `champ_front_base` (17.1)

Usage:

boundary_field_inward *str*
Read *str* {

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.

17.7 Ch_front_input

Description: `not_set`

See also: `champ_front_base` (17.1) `ch_front_input_uniforme` (17.8)

Usage:

ch_front_input *str*
Read *str* {

nb_comp *int*

```

    nom str
    [ initial_value n x1 x2 ... xn]
    probleme str
    [ sous_zone str]
}
where

```

- **nb_comp** *int*
- **nom** *str*
- **initial_value** *n x1 x2 ... xn*
- **probleme** *str*
- **sous_zone** *str*

17.8 Ch_front_input_uniforme

Description: for coupling, you can use `ch_front_input_uniforme` which is a `champ_front_uniforme`, which use an external value. It must be used with `Problem.setInputField`.

See also: `ch_front_input` ([17.7](#))

Usage:

```

ch_front_input_uniforme str
Read str {

```

```

    nb_comp int
    nom str
    [ initial_value n x1 x2 ... xn]
    probleme str
    [ sous_zone str]

```

```

}
where

```

- **nb_comp** *int* for inheritance
- **nom** *str* for inheritance
- **initial_value** *n x1 x2 ... xn* for inheritance
- **probleme** *str* for inheritance
- **sous_zone** *str* for inheritance

17.9 Champ_front_med

Description: Field allowing the loading of a boundary condition from a MED file using `Champ_fonc_med`

See also: `champ_front_base` ([17.1](#))

Usage:

```

champ_front_MED champ_fonc_med
where

```

- **champ_fonc_med** *champ_base* ([16.1](#)): a `champ_fonc_med` loading the values of the unknown on a domain boundary

17.10 Champ_front_bruite

Description: Field which is variable in time and space in a random manner.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_bruite nb_comp bloc

where

- **nb_comp** *int*: Number of field components.
- **bloc** *bloc_lecture* ([3.60](#)): { [N val L val] Moyenne m_1.....[m_i] Amplitude A_1.....[A_i] }:
Random noise: If N and L are not defined, the ith component of the field varies randomly around an average value m_i with a maximum amplitude A_i.
White noise: If N and L are defined, these two additional parameters correspond to L, the domain length and N, the number of nodes in the domain. Noise frequency will be between $2\pi/L$ and $2\pi N/(4L)$.
For example, formula for velocity: $u=U0(t)$ $v=U1(t)$ $Uj(t)=Mj+2\cdot Aj\cdot \text{bruit_blanc}$ where `bruit_blanc` (white_noise) is the formula given in the `mettre_a_jour` (update) method of the `Champ_front_bruite` (`noise_boundary_field`) (Refer to the `Champ_front_bruite.cpp` file).

17.11 Champ_front_calc

Description: This keyword is used on a boundary to get a field from another boundary. The local and remote boundaries should have the same mesh. If not, the `Champ_front_recyclage` keyword could be used instead. It is used in the condition block at the limits of equation which itself refers to a problem called pb1. We are working under the supposition that pb1 is coupled to another problem.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_calc problem_name bord field_name

where

- **problem_name** *str*: Name of the other problem to which pb1 is coupled.
- **bord** *str*: Name of the side which is the boundary between the 2 domains in the domain object description associated with the `problem_name` object.
- **field_name** *str*: Name of the field containing the value that the user wishes to use at the boundary. The `field_name` object must be recognized by the `problem_name` object.

17.12 Champ_front_composite

Description: Composite front field. Used in multiphase problems to associate data to each phase.

See also: `champ_front_base` ([17.1](#)) `champ_front_musig` ([17.23](#))

Usage:

champ_front_composite dim bloc

where

- **dim** *int*: Number of field components.
- **bloc** *bloc_lecture* ([3.60](#)): Values Various pieces of the field, defined per phase. Part 1 goes to phase 1, etc...

17.13 Champ_front_contact_vef

Description: This field is used on a boundary between a solid and fluid domain to exchange a calculated temperature at the contact face of the two domains according to the flux of the two problems.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_contact_vef local_pb local_boundary remote_pb remote_boundary
where

- **local_pb** *str*: Name of the problem.
- **local_boundary** *str*: Name of the boundary.
- **remote_pb** *str*: Name of the second problem.
- **remote_boundary** *str*: Name of the boundary in the second problem.

17.14 Champ_front_debit

Description: This field is used to define a flow rate field instead of a velocity field for a Dirichlet boundary condition on Navier-Stokes equations.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_debit ch
where

- **ch** *champ_front_base* ([17.1](#)): uniform field in space to define the flow rate. It could be, for example, `champ_front_uniforme`, `ch_front_input_uniform` or `champ_front_fonc_txyz` that depends only on time.

17.15 Champ_front_debit_massique

Description: This field is used to define a flow rate field using the density

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_debit_massique ch
where

- **ch** *champ_front_base* ([17.1](#)): uniform field in space to define the flow rate. It could be, for example, `champ_front_uniforme`, `ch_front_input_uniform` or `champ_front_fonc_txyz` that depends only on time.

17.16 Champ_front_fonc_pois_ipsn

Description: Boundary field `champ_front_fonc_pois_ipsn`.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_fonc_pois_ipsn r_tube umoy r_loc
where

- **r_tube** *float*
- **umoy** *n x1 x2 ... xn*
- **r_loc** *x1 x2 (x3)*

17.17 Champ_front_fonc_pois_tube

Description: Boundary field champ_front_fonc_pois_tube.

See also: champ_front_base ([17.1](#))

Usage:

champ_front_fonc_pois_tube **r_tube** **umoy** **r_loc** **r_loc_mult**

where

- **r_tube** *float*
- **umoy** *n x1 x2 ... xn*
- **r_loc** *x1 x2 (x3)*
- **r_loc_mult** *n1 n2 (n3)*

17.18 Champ_front_fonc_t

Description: Boundary field that depends only on time.

See also: champ_front_base ([17.1](#))

Usage:

champ_front_fonc_t **val**

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

17.19 Champ_front_fonc_txyz

Description: Boundary field which is not constant in space and in time.

See also: champ_front_base ([17.1](#)) Champ_Front_xyz_Tabule ([17.2](#))

Usage:

champ_front_fonc_txyz **val**

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

17.20 Champ_front_fonc_xyz

Description: Boundary field which is not constant in space.

See also: champ_front_base ([17.1](#))

Usage:

champ_front_fonc_xyz **val**

where

- **val** *n word1 word2 ... wordn*: Values of field components (mathematical expressions).

17.21 Champ_front_fonction

Description: boundary field that is function of another field

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_fonction dim inco expression

where

- **dim** *int*: Number of field components.
- **inco** *str*: Name of the field (for example: temperature).
- **expression** *str*: keyword to use a analytical expression like `10.*EXP(-0.1*val)` where val be the keyword for the field.

17.22 Champ_front_lu

Description: boundary field which is given from data issued from a read file. The format of this file has to be the same that the one generated by `Ecrire_fichier_xyz_valeur`

Example for K and epsilon quantities to be defined for inlet condition in a boundary named 'entree':

`entree frontiere_ouverte_K_Eps_impose Champ_Front_lu dom 2pb_K_EPS_PERIO_1006.306198.dat`

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_lu domaine dim file

where

- **domaine** *str*: Name of domain
- **dim** *int*: number of components
- **file** *str*: path for the read file

17.23 Champ_front_musig

Description: MUSIG front field. Used in multiphase problems to associate data to each phase.

See also: `champ_front_composite` ([17.12](#))

Usage:

champ_front_musig bloc

where

- **bloc** *bloc_lecture* ([3.60](#)): Not set

17.24 Champ_front_normal_vef

Description: Field to define the normal vector field standard at the boundary in VEF discretization.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_normal_vef mot vit_tan

where

- **mot** *str* into ['valeur_normale']: Name of vector field.
- **vit_tan** *float*: normal vector value (positive value for a vector oriented outside to inside).

17.25 Champ_front_pression_from_u

Description: this field is used to define a pressure field depending of a velocity field.

See also: `champ_front_base` (17.1)

Usage:

champ_front_pression_from_u *expression*

where

- **expression** *str*: value depending of a velocity (like $2 * u_{moy}^2$).

17.26 Champ_front_recyclage

Description: This keyword is used on a boundary to get a field from another boundary.

It is to use, in a general way, on a boundary of a local_pb problem, a field calculated from a linear combination of an imposed field $g(x,y,z,t)$ with an instantaneous $f(x,y,z,t)$ and a spatial mean field $\langle f \rangle(t)$ or a temporal mean field $\langle f \rangle(x,y,z)$ extracted from a plane of a problem named pb (pb may be local_pb itself):

For each component i , the field F applied on the boundary will be:

$$F_i(x,y,z,t) = \alpha_i g_i(x,y,z,t) + \chi_i [f_i(x,y,z,t) - \beta_i \langle f_i \rangle]$$

See also: `champ_front_base` (17.1)

Usage:

champ_front_recyclage *str*

Read *str* {

```

pb_champ_evaluateur pb_champ_evaluateur
[ distance_plan x1 x2 (x3)]
[ ampli_moyenne_imposee n x1 x2 ... xn]
[ ampli_moyenne_recyclee n x1 x2 ... xn]
[ ampli_fluctuation n x1 x2 ... xn]
[ direction_anisotrope int into [1, 2, 3]]
[ moyenne_imposee moyenne_imposee_deriv]
[ moyenne_recyclee str]
[ fichier str]

```

}

where

- **pb_champ_evaluateur** *pb_champ_evaluateur* (27)
- **distance_plan** *x1 x2 (x3)*: Vector which gives the distance between the boundary and the plane from where the field F will be extracted. By default, the vector is zero, that should imply the two domains have coincident boundaries.
- **ampli_moyenne_imposee** *n x1 x2 ... xn*: 2|3 $\alpha(0) \alpha(1) [\alpha(2)]$: α_i coefficients (by default =1)
- **ampli_moyenne_recyclee** *n x1 x2 ... xn*: 2|3 $\beta(0) \beta(1) [\beta(2)]$: β_i coefficients (by default =1)
- **ampli_fluctuation** *n x1 x2 ... xn*: 2|3 $\gamma(0) \gamma(1) [\gamma(2)]$: γ_i coefficients (by default =1)

- **direction_anisotrope** *int* into [1, 2, 3]: If an integer is given for direction (X:1, Y:2, Z:3, by default, direction is negative), the imposed field *g* will be 0 for the 2 other directions.
- **moyenne_imposee** *moyenne_imposee_deriv* (24): Value of the imposed *g* field.
- **moyenne_recyclee** *str*: Method used to perform a spatial or a temporal averaging of *f* field to specify <f>. <f> can be the surface mean of *f* on the plane (surface option, see below) or it can be read from several files (for example generated by the *chmoy_faceperio* option of the *Traitement_particulier* keyword to obtain a temporal mean field). The option *methode_recyc* can be: *surfacique*, Surface mean for <f> from *f* values on the plane ; Or one of the following *methode_moy* options applied to read a temporal mean field <f>(x,y,z): *interpolation*, *connexion_approchee* or *connexion_exacte*
- **fichier** *str*

17.27 Champ_front_tabule

Description: Constant field on the boundary, tabulated as a function of time.

See also: *champ_front_base* (17.1) *champ_front_tabule_lu* (17.28)

Usage:

champ_front_tabule **nb_comp** **bloc**

where

- **nb_comp** *int*: Number of field components.
 - **bloc** *bloc_lecture* (3.60): {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.

17.28 Champ_front_tabule_lu

Description: Constant field on the boundary, tabulated from a specified column file. Lines starting with # are ignored.

See also: *champ_front_tabule* (17.27)

Usage:

champ_front_tabule_lu **nb_comp** **column_file**

where

- **nb_comp** *int*: Number of field components.
- **column_file** *str*: Name of the column file.

17.29 Champ_front_tangentiel_vef

Description: Field to define the tangential velocity vector field standard at the boundary in VEF discretization.

See also: *champ_front_base* (17.1)

Usage:

champ_front_tangentiel_vef **mot** **vit_tan**

where

- **mot** *str* into ['vitesse_tangentielle']: Name of vector field.
- **vit_tan** *float*: Vector field standard [m/s].

17.30 Champ_front_uniforme

Description: Boundary field which is constant in space and stationary.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_uniforme *val*

where

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

17.31 Champ_front_xyz_debit

Description: This field is used to define a flow rate field with a velocity profil which will be normalized to match the flow rate chosen.

See also: `champ_front_base` ([17.1](#))

Usage:

champ_front_xyz_debit *str*

Read *str* {

 [**velocity_profil** *champ_front_base*]
 flow_rate *champ_front_base*

}

where

- **velocity_profil** *champ_front_base* ([17.1](#)): *velocity_profil* 0 velocity field to define the profil of velocity.
- **flow_rate** *champ_front_base* ([17.1](#)): *flow_rate* 1 uniform field in space to define the flow rate. It could be, for example, `champ_front_uniforme`, `ch_front_input_uniform` or `champ_front_fonc_t`

18 interpolation_ibm_base

Description: Base class for all the interpolation methods available in the Immersed Boundary Method (IBM).

See also: `objet_u` ([40](#)) `ibm_element_fluide` ([18.3](#)) `ibm_gradient_moyen` ([18.5](#)) `ibm_aucune` ([18.2](#))

Usage:

interpolation_ibm_base [*impr*] [*nb_histo_boxes_impr*]

where

- **impr** : To print IBM-related data
- **nb_histo_boxes_impr** *int*: number of histogram boxes for printed data

18.1 Interpolation_ibm_power_law_tbl_u_star

Description: Immersed Boundary Method (IBM): law u star.

See also: `ibm_gradient_moyen` ([18.5](#))

Usage:

Interpolation_IBM_power_law_tbl_u_star *str*

Read *str* {

points_solides *champ_base*
est_dirichlet *champ_base*
correspondance_elements *champ_base*
elements_solides *champ_base*
[**impr**]
[**nb_histo_boxes_impr** *int*]

}

where

- **points_solides** *champ_base* (16.1): Node field giving the projection of the node on the immersed boundary
- **est_dirichlet** *champ_base* (16.1): Node field of booleans indicating whether the node belong to an element where the interface is
- **correspondance_elements** *champ_base* (16.1): Cell field giving the SALOME cell number
- **elements_solides** *champ_base* (16.1): Node field giving the element number containing the solid point
- **impr** for inheritance: To print IBM-related data
- **nb_histo_boxes_impr** *int* for inheritance: number of histogram boxes for printed data

18.2 Ibm_aucune

Synonymous: **interpolation_ibm_aucune**

Description: Immersed Boundary Method (IBM): no interpolation.

See also: interpolation_ibm_base (18)

Usage:

ibm_aucune [**impr**] [**nb_histo_boxes_impr**]

where

- **impr** : To print IBM-related data
- **nb_histo_boxes_impr** *int*: number of histogram boxes for printed data

18.3 Ibm_element_fluide

Synonymous: **interpolation_ibm_element_fluide**

Description: Immersed Boundary Method (IBM): fluid element interpolation.

See also: interpolation_ibm_base (18) ibm_hybride (18.4) ibm_power_law_tbl (18.6)

Usage:

ibm_element_fluide *str*

Read *str* {

points_fluides *champ_base*
points_solides *champ_base*

```

    elements_fluides champ_base
    correspondance_elements champ_base
    [ impr ]
    [ nb_histo_boxes_impr int]
}
where

```

- **points_fluides** *champ_base* (16.1): Node field giving the projection of the point below (points_solides) falling into the pure cell fluid
- **points_solides** *champ_base* (16.1): Node field giving the projection of the node on the immersed boundary
- **elements_fluides** *champ_base* (16.1): Node field giving the number of the element (cell) containing the pure fluid point
- **correspondance_elements** *champ_base* (16.1): Cell field giving the SALOME cell number
- **impr** for inheritance: To print IBM-related data
- **nb_histo_boxes_impr** *int* for inheritance: number of histogram boxes for printed data

18.4 Ibm_hybride

Synonymous: **interpolation_ibm_hybride**

Description: Immersed Boundary Method (IBM): hybrid (fluid/mean gradient) interpolation.

See also: **ibm_element_fluide** (18.3)

Usage:

ibm_hybride *str*

Read *str* {

```

    est_dirichlet champ_base
    elements_solides champ_base
    points_fluides champ_base
    points_solides champ_base
    elements_fluides champ_base
    correspondance_elements champ_base
    [ impr ]
    [ nb_histo_boxes_impr int]

```

}

where

- **est_dirichlet** *champ_base* (16.1): Node field of booleans indicating whether the node belong to an element where the interface is
- **elements_solides** *champ_base* (16.1): Node field giving the element number containing the solid point
- **points_fluides** *champ_base* (16.1) for inheritance: Node field giving the projection of the point below (points_solides) falling into the pure cell fluid
- **points_solides** *champ_base* (16.1) for inheritance: Node field giving the projection of the node on the immersed boundary
- **elements_fluides** *champ_base* (16.1) for inheritance: Node field giving the number of the element (cell) containing the pure fluid point
- **correspondance_elements** *champ_base* (16.1) for inheritance: Cell field giving the SALOME cell number
- **impr** for inheritance: To print IBM-related data
- **nb_histo_boxes_impr** *int* for inheritance: number of histogram boxes for printed data

18.5 Ibm_gradient_moyen

Synonymous: **interpolation_ibm_gradient_moyen**

Description: Immersed Boundary Method (IBM): mean gradient interpolation.

See also: **interpolation_ibm_base** ([18](#)) **Interpolation_IBM_power_law_tbl_u_star** ([18.1](#))

Usage:

ibm_gradient_moyen *str*

Read *str* {

```
    points_solides champ_base
    est_dirichlet champ_base
    correspondance_elements champ_base
    elements_solides champ_base
    [ impr ]
    [ nb_histo_boxes_impr int]
```

}

where

- **points_solides** *champ_base* ([16.1](#)): Node field giving the projection of the node on the immersed boundary
- **est_dirichlet** *champ_base* ([16.1](#)): Node field of booleans indicating whether the node belong to an element where the interface is
- **correspondance_elements** *champ_base* ([16.1](#)): Cell field giving the SALOME cell number
- **elements_solides** *champ_base* ([16.1](#)): Node field giving the element number containing the solid point
- **impr** for inheritance: To print IBM-related data
- **nb_histo_boxes_impr** *int* for inheritance: number of histogram boxes for printed data

18.6 Ibm_power_law_tbl

Synonymous: **interpolation_ibm_power_law_tbl**

Description: Immersed Boundary Method (IBM): power law interpolation.

See also: **ibm_element_fluide** ([18.3](#))

Usage:

ibm_power_law_tbl *str*

Read *str* {

```
    [ formulation_linear_pwl int ]
    points_fluides champ_base
    points_solides champ_base
    elements_fluides champ_base
    correspondance_elements champ_base
    [ impr ]
    [ nb_histo_boxes_impr int]
```

}

where

- **formulation_linear_pwl** *int*: Choix formulation lineaire ou non

- **points_fluides** *champ_base* (16.1) for inheritance: Node field giving the projection of the point below (points_solides) falling into the pure cell fluid
- **points_solides** *champ_base* (16.1) for inheritance: Node field giving the projection of the node on the immersed boundary
- **elements_fluides** *champ_base* (16.1) for inheritance: Node field giving the number of the element (cell) containing the pure fluid point
- **correspondance_elements** *champ_base* (16.1) for inheritance: Cell field giving the SALOME cell number
- **impr** for inheritance: To print IBM-related data
- **nb_histo_boxes_impr** *int* for inheritance: number of histogram boxes for printed data

19 loi_etat_base

Description: Basic class for state laws used with a dilatable fluid.

See also: [objet_u](#) (40) [loi_etat_gaz_reel_base](#) (19.8) [loi_etat_gaz_parfait_base](#) (19.7) [loi_etat_tppi_base](#) (19.9)

Usage:

19.1 Eos_qc

Description: Class for using EOS with QC problem

See also: [loi_etat_tppi_base](#) (19.9)

Usage:

EOS_QC *str*

Read *str* {

Cp *float*

fluid *str*

model *str*

}

where

- **Cp** *float*: Specific heat at constant pressure (J/kg/K).
- **fluid** *str*: Fluid name in the EOS model
- **model** *str*: EOS model name

19.2 Eos_wc

Description: Class for using EOS with WC problem

See also: [loi_etat_tppi_base](#) (19.9)

Usage:

EOS_WC *str*

Read *str* {

Cp *float*

fluid *str*

model *str*

```
}
```

where

- **Cp** *float*: Specific heat at constant pressure (J/kg/K).
- **fluid** *str*: Fluid name in the EOS model
- **model** *str*: EOS model name

19.3 Binaire_gaz_parfait_qc

Description: Class for perfect gas binary mixtures state law used with a quasi-compressible fluid under the iso-thermal and iso-bar assumptions.

See also: `loi_etat_gaz_parfait_base` ([19.7](#))

Usage:

binaire_gaz_parfait_QC *str*

Read *str* {

```
    molar_mass1 float
    molar_mass2 float
    mu1 float
    mu2 float
    temperature float
    diffusion_coeff float
```

```
}
```

where

- **molar_mass1** *float*: Molar mass of species 1 (in kg/mol).
- **molar_mass2** *float*: Molar mass of species 2 (in kg/mol).
- **mu1** *float*: Dynamic viscosity of species 1 (in kg/m.s).
- **mu2** *float*: Dynamic viscosity of species 2 (in kg/m.s).
- **temperature** *float*: Temperature (in Kelvin) which will be constant during the simulation since this state law only works for iso-thermal conditions.
- **diffusion_coeff** *float*: Diffusion coefficient assumed the same for both species (in m²/s).

19.4 Binaire_gaz_parfait_wc

Description: Class for perfect gas binary mixtures state law used with a weakly-compressible fluid under the iso-thermal and iso-bar assumptions.

See also: `loi_etat_gaz_parfait_base` ([19.7](#))

Usage:

binaire_gaz_parfait_WC *str*

Read *str* {

```
    molar_mass1 float
    molar_mass2 float
    mu1 float
    mu2 float
    temperature float
    diffusion_coeff float
```

}
where

- **molar_mass1** *float*: Molar mass of species 1 (in kg/mol).
- **molar_mass2** *float*: Molar mass of species 2 (in kg/mol).
- **mu1** *float*: Dynamic viscosity of species 1 (in kg/m.s).
- **mu2** *float*: Dynamic viscosity of species 2 (in kg/m.s).
- **temperature** *float*: Temperature (in Kelvin) which will be constant during the simulation since this state law only works for iso-thermal conditions.
- **diffusion_coeff** *float*: Diffusion coefficient assumed the same for both species (in m²/s).

19.5 Coolprop_qc

Description: Class for using CoolProp with QC problem

See also: `loi_etat_tppi_base` ([19.9](#))

Usage:

coolprop_QC *str*

Read *str* {

Cp *float*

fluid *str*

model *str*

}

where

- **Cp** *float*: Specific heat at constant pressure (J/kg/K).
- **fluid** *str*: Fluid name in the CoolProp model
- **model** *str*: CoolProp model name

19.6 Coolprop_wc

Description: Class for using CoolProp with WC problem

See also: `loi_etat_tppi_base` ([19.9](#))

Usage:

coolprop_WC *str*

Read *str* {

Cp *float*

fluid *str*

model *str*

}

where

- **Cp** *float*: Specific heat at constant pressure (J/kg/K).
- **fluid** *str*: Fluid name in the CoolProp model
- **model** *str*: CoolProp model name

19.7 Loi_etat_gaz_parfait_base

Description: Basic class for perfect gases state laws used with a dilatable fluid.

See also: [loi_etat_base \(19\)](#) [rhoT_gaz_parfait_QC \(19.14\)](#) [binaire_gaz_parfait_QC \(19.3\)](#) [multi_gaz_parfait_QC \(19.10\)](#) [gaz_parfait_QC \(19.12\)](#) [multi_gaz_parfait_WC \(19.11\)](#) [binaire_gaz_parfait_WC \(19.4\)](#) [gaz_parfait_WC \(19.13\)](#)

Usage:

19.8 Loi_etat_gaz_reel_base

Description: Basic class for real gases state laws used with a dilatable fluid.

See also: [loi_etat_base \(19\)](#) [rhoT_gaz_reel_QC \(19.15\)](#)

Usage:

19.9 Loi_etat_tppi_base

Description: Basic class for thermo-physical properties interface (TPPI) used for dilatable problems

See also: [loi_etat_base \(19\)](#) [coolprop_QC \(19.5\)](#) [EOS_QC \(19.1\)](#) [EOS_WC \(19.2\)](#) [coolprop_WC \(19.6\)](#)

Usage:

19.10 Multi_gaz_parfait_qc

Description: Class for perfect gas multi-species mixtures state law used with a quasi-compressible fluid.

See also: [loi_etat_gaz_parfait_base \(19.7\)](#)

Usage:

multi_gaz_parfait_QC *str*

Read *str* {

```
    sc float
    prandtl float
    [ cp float ]
    [ dtol_fraction float ]
    [ correction_fraction ]
    [ ignore_check_fraction ]
```

}

where

- **sc** *float*: Schmidt number of the gas $Sc = \nu/D$ (D : diffusion coefficient of the mixing).
- **prandtl** *float*: Prandtl number of the gas $Pr = \mu * Cp / \lambda$
- **cp** *float*: Specific heat at constant pressure of the gas Cp .
- **dtol_fraction** *float*: Delta tolerance on mass fractions for check testing (default value 1.e-6).
- **correction_fraction** : To force mass fractions between 0. and 1.
- **ignore_check_fraction** : Not to check if mass fractions between 0. and 1.

19.11 Multi_gaz_parfait_wc

Description: Class for perfect gas multi-species mixtures state law used with a weakly-compressible fluid.

See also: `loi_etat_gaz_parfait_base` ([19.7](#))

Usage:

multi_gaz_parfait_WC *str*

```
Read str {  
    species_number int  
    diffusion_coeff champ_base  
    molar_mass champ_base  
    mu champ_base  
    cp champ_base  
    prandtl float
```

```
}
```

where

- **species_number** *int*: Number of species you are considering in your problem.
- **diffusion_coeff** *champ_base* ([16.1](#)): Diffusion coefficient of each species, defined with a `Champ_uniforme` of dimension equals to the `species_number`.
- **molar_mass** *champ_base* ([16.1](#)): Molar mass of each species, defined with a `Champ_uniforme` of dimension equals to the `species_number`.
- **mu** *champ_base* ([16.1](#)): Dynamic viscosity of each species, defined with a `Champ_uniforme` of dimension equals to the `species_number`.
- **cp** *champ_base* ([16.1](#)): Specific heat at constant pressure of the gas C_p , defined with a `Champ_uniforme` of dimension equals to the `species_number`.
- **prandtl** *float*: Prandtl number of the gas $Pr = \mu * C_p / \lambda$.

19.12 Gaz_parfait_qc

Description: Class for perfect gas state law used with a quasi-compressible fluid.

See also: `loi_etat_gaz_parfait_base` ([19.7](#))

Usage:

gaz_parfait_QC *str*

```
Read str {  
    Cp float  
    [ Cv float ]  
    [ gamma float ]  
    Prandtl float  
    [ rho_constant_pour_debug champ_base ]
```

```
}
```

where

- **Cp** *float*: Specific heat at constant pressure (J/kg/K).
- **Cv** *float*: Specific heat at constant volume (J/kg/K).
- **gamma** *float*: C_p / C_v
- **Prandtl** *float*: Prandtl number of the gas $Pr = \mu * C_p / \lambda$
- **rho_constant_pour_debug** *champ_base* ([16.1](#)): For developers to debug the code with a constant ρ .

19.13 Gaz_parfait_wc

Description: Class for perfect gas state law used with a weakly-compressible fluid.

See also: [loi_etat_gaz_parfait_base \(19.7\)](#)

Usage:

gaz_parfait_WC *str*

Read *str* {

Cp *float*
[**Cv** *float*]
[**gamma** *float*]
Prandtl *float*

}

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 $Pr = \mu * C_p / \lambda$

19.14 Rhot_gaz_parfait_qc

Description: Class for perfect gas used with a quasi-compressible fluid where the state equation is defined as $\rho = f(T)$.

See also: [loi_etat_gaz_parfait_base \(19.7\)](#)

Usage:

rhoT_gaz_parfait_QC *str*

Read *str* {

cp *float*
[**prandtl** *float*]
[**rho_xyz** *champ_base*]
[**rho_t** *str*]
[**t_min** *float*]

}

where

- **cp** *float*: Specific heat at constant pressure of the gas C_p .
- **prandtl** *float*: Prandtl number of the gas $Pr = \mu * C_p / \lambda$
- **rho_xyz** *champ_base* (16.1): Defined with a *Champ_Fonc_xyz* to define a constant ρ with time (space dependent)
- **rho_t** *str*: Expression of T used to calculate ρ . This can lead to a variable ρ , both in space and in time.
- **t_min** *float*: Temperature may, in some cases, locally and temporarily be very small (and negative) even though computation converges. *T_min* keyword allows to set a lower limit of temperature (in Kelvin, -1000 by default). WARNING: DO NOT USE THIS KEYWORD WITHOUT CHECKING CAREFULLY YOUR RESULTS!

19.15 Rhot_gaz_reel_qc

Description: Class for real gas state law used with a quasi-compressible fluid.

See also: `loi_etat_gaz_reel_base` ([19.8](#))

Usage:

rhoT_gaz_reel_QC **bloc**

where

- **bloc** *bloc_lecture* ([3.60](#)): Description.

20 loi_fermeture_base

Description: Class for appends fermeture to problem

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

See also: `objet_u` ([40](#)) `loi_fermeture_test` ([20.1](#))

Usage:

20.1 Loi_fermeture_test

Description: Loi for test only

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

See also: `loi_fermeture_base` ([20](#))

Usage:

loi_fermeture_test *str*

Read *str* {

 [**coef** *float*]

}

where

- **coef** *float*: coefficient

21 loi_horaire

Description: to define the movement with a time-dependant law for the solid interface.

See also: `objet_u` ([40](#))

Usage:

loi_horaire *str*

Read *str* {

position *n word1 word2 ... wordn*

vitesse *n word1 word2 ... wordn*

 [**rotation** *n word1 word2 ... wordn*]

 [**derivee_rotation** *n word1 word2 ... wordn*]

 [**verification_derivee** *int*]


```
[ impr int]
```

```
}
```

where

- **position** *n word1 word2 ... wordn*: Vecteur position
- **vitesse** *n word1 word2 ... wordn*: Vecteur vitesse
- **rotation** *n word1 word2 ... wordn*: Matrice de passage
- **derivee_rotation** *n word1 word2 ... wordn*: Derivee matrice de passage
- **verification_derivee** *int*
- **impr** *int*: Whether to print output

22 milieu_base

Description: Basic class for medium (physics properties of medium).

See also: objet_u (40) constituant (22.1) solide (22.13) fluide_base (22.2)

Usage:

milieu_base *str*

Read *str* {

```
[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]
[ porosites porosites]
[ rho champ_base]
[ lambda champ_base]
[ cp champ_base]
```

```
}
```

where

- **gravite** *champ_base* (16.1): Gravity field (optional).
- **porosites_champ** *champ_base* (16.1): 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 $\Psi(\text{elem1})$, $\Psi(\text{elem2})$: $\Psi(\text{face}) = 2 / (1/\Psi(\text{elem1}) + 1/\Psi(\text{elem2}))$. This keyword is optional.
- **diametre_hyd_champ** *champ_base* (16.1): Hydraulic diameter field (optional).
- **porosites** *porosites* (28): Porosities.
- **rho** *champ_base* (16.1): Density (kg.m-3).
- **lambda** *champ_base* (16.1): Conductivity (W.m-1.K-1).
- **cp** *champ_base* (16.1): Specific heat (J.kg-1.K-1).

22.1 Constituant

Description: Constituent.

See also: milieu_base (22)

Usage:

constituant *str*

Read *str* {

```
[ coefficient_diffusion champ_base]
```

```

[ is_multi_scalar ]
[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]
[ porosites porosites]
[ rho champ_base]
[ lambda champ_base]
[ cp champ_base]
}
where

```

- **coefficient_diffusion** *champ_base* (16.1): Constituent diffusion coefficient value (m².s⁻¹). If a multi-constituent problem is being processed, the diffusivity will be a vectorial and each components will be the diffusion of the constituent.
- **is_multi_scalar** : Flag to activate the multi_scalar diffusion operator
- **gravite** *champ_base* (16.1) for inheritance: Gravity field (optional).
- **porosites_champ** *champ_base* (16.1) for inheritance: 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 $\Psi(\text{elem1})$, $\Psi(\text{elem2})$: $\Psi(\text{face}) = 2 / (1/\Psi(\text{elem1}) + 1/\Psi(\text{elem2}))$. This keyword is optional.
- **diametre_hyd_champ** *champ_base* (16.1) for inheritance: Hydraulic diameter field (optional).
- **porosites** *porosites* (28) for inheritance: Porosities.
- **rho** *champ_base* (16.1) for inheritance: Density (kg.m⁻³).
- **lambda** *champ_base* (16.1) for inheritance: Conductivity (W.m⁻¹.K⁻¹).
- **cp** *champ_base* (16.1) for inheritance: Specific heat (J.kg⁻¹.K⁻¹).

22.2 Fluide_base

Description: Basic class for fluids.

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

See also: milieu_base (22) fluide_incompressible (22.4) fluide_reel_base (22.8) fluide_dilatable_base (22.3)

Usage:

fluide_base *str*

Read *str* {

```

[ indice champ_base]
[ kappa champ_base]
[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]
[ porosites porosites]
[ rho champ_base]
[ lambda champ_base]
[ cp champ_base]

```

}
where

- **indice** *champ_base* (16.1): Refractivity of fluid.
- **kappa** *champ_base* (16.1): Absorptivity of fluid (m⁻¹).
- **gravite** *champ_base* (16.1) for inheritance: Gravity field (optional).

- **porosites_champ** *champ_base* (16.1) for inheritance: 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 $\Psi(\text{elem1})$, $\Psi(\text{elem2})$: $\Psi(\text{face})=2/(1/\Psi(\text{elem1})+1/\Psi(\text{elem2}))$. This keyword is optional.
- **diametre_hyd_champ** *champ_base* (16.1) for inheritance: Hydraulic diameter field (optional).
- **porosites** *porosites* (28) for inheritance: Porosities.
- **rho** *champ_base* (16.1) for inheritance: Density (kg.m-3).
- **lambda** *champ_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).
- **cp** *champ_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).

22.3 Fluides_dilatable_base

Description: Basic class for dilatable fluids.

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

See also: *fluides_base* (22.2) *fluides_quasi_compressible* (22.6) *fluides_weakly_compressible* (22.12)

Usage:

fluides_dilatable_base *str*

Read *str* {

```
[ indice champ_base]
[ kappa champ_base]
[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]
[ porosites porosites]
[ rho champ_base]
[ lambda champ_base]
[ cp champ_base]
```

}

where

- **indice** *champ_base* (16.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (16.1) for inheritance: Absorptivity of fluid (m-1).
- **gravite** *champ_base* (16.1) for inheritance: Gravity field (optional).
- **porosites_champ** *champ_base* (16.1) for inheritance: 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 $\Psi(\text{elem1})$, $\Psi(\text{elem2})$: $\Psi(\text{face})=2/(1/\Psi(\text{elem1})+1/\Psi(\text{elem2}))$. This keyword is optional.
- **diametre_hyd_champ** *champ_base* (16.1) for inheritance: Hydraulic diameter field (optional).
- **porosites** *porosites* (28) for inheritance: Porosities.
- **rho** *champ_base* (16.1) for inheritance: Density (kg.m-3).
- **lambda** *champ_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).
- **cp** *champ_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).

22.4 Fluides_incompressible

Description: Class for non-compressible fluids.

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

See also: `fluide_base` (22.2) `fluide_ostwald` (22.5)

Usage:

fluide_incompressible *str*

Read *str* {

```
[ beta_th champ_base]  
[ mu champ_base]  
[ beta_co champ_base]  
[ rho champ_base]  
[ cp champ_base]  
[ lambda champ_base]  
[ porosites bloc_lecture]  
[ indice champ_base]  
[ kappa champ_base]  
[ gravite champ_base]  
[ porosites_champ champ_base]  
[ diametre_hyd_champ champ_base]
```

}

where

- **beta_th** *champ_base* (16.1): Thermal expansion (K-1).
- **mu** *champ_base* (16.1): Dynamic viscosity (kg.m-1.s-1).
- **beta_co** *champ_base* (16.1): Volume expansion coefficient values in concentration.
- **rho** *champ_base* (16.1): Density (kg.m-3).
- **cp** *champ_base* (16.1): Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (16.1): Conductivity (W.m-1.K-1).
- **porosites** *bloc_lecture* (3.60): Porosity (optional)
- **indice** *champ_base* (16.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (16.1) for inheritance: Absorptivity of fluid (m-1).
- **gravite** *champ_base* (16.1) for inheritance: Gravity field (optional).
- **porosites_champ** *champ_base* (16.1) for inheritance: 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 $\Psi(\text{elem1})$, $\Psi(\text{elem2})$: $\Psi(\text{face}) = 2 / (1/\Psi(\text{elem1}) + 1/\Psi(\text{elem2}))$. This keyword is optional.
- **diametre_hyd_champ** *champ_base* (16.1) for inheritance: Hydraulic diameter field (optional).

22.5 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 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.

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

See also: `fluide_incompressible` (22.4)

Usage:

fluide_ostwald *str*

Read *str* {

```
[ k champ_base]
```

```

[ n champ_base]
[ beta_th champ_base]
[ mu champ_base]
[ beta_co champ_base]
[ rho champ_base]
[ cp champ_base]
[ lambda champ_base]
[ porosites bloc_lecture]
[ indice champ_base]
[ kappa champ_base]
[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]
}
where

```

- **k** *champ_base* (16.1): Fluid consistency.
- **n** *champ_base* (16.1): Fluid structure index.
- **beta_th** *champ_base* (16.1) for inheritance: Thermal expansion (K-1).
- **mu** *champ_base* (16.1) for inheritance: Dynamic viscosity (kg.m-1.s-1).
- **beta_co** *champ_base* (16.1) for inheritance: Volume expansion coefficient values in concentration.
- **rho** *champ_base* (16.1) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).
- **porosites** *bloc_lecture* (3.60) for inheritance: Porosity (optional)
- **indice** *champ_base* (16.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (16.1) for inheritance: Absorptivity of fluid (m-1).
- **gravite** *champ_base* (16.1) for inheritance: Gravity field (optional).
- **porosites_champ** *champ_base* (16.1) for inheritance: 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 $\Psi(\text{elem1})$, $\Psi(\text{elem2})$: $\Psi(\text{face}) = 2 / (1/\Psi(\text{elem1}) + 1/\Psi(\text{elem2}))$. This keyword is optional.
- **diametre_hyd_champ** *champ_base* (16.1) for inheritance: Hydraulic diameter field (optional).

22.6 **Fluide_quasi_compressible**

Description: Quasi-compressible flow with a low mach number assumption; this means that the thermodynamic pressure (used in state law) is uniform in space.

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

See also: **fluide_dilatable_base** (22.3)

Usage:

fluide_quasi_compressible *str*

Read *str* {

```

[ 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]

```

```

[ lambda champ_base]
[ mu champ_base]
[ indice champ_base]
[ kappa champ_base]
[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]
[ porosites porosites]
[ rho champ_base]
[ cp champ_base]
}
where

```

- **sutherland** *bloc_sutherland* (22.7): Sutherland law for viscosity and for conductivity.
- **pression** *float*: Initial thermo-dynamic pressure used in the associated state law.
- **loi_etat** *loi_etat_base* (19): The state law that will be associated to the Quasi-compressible fluid.
- **traitement_pth** *str* into ['edo', 'constant', 'conservation_masse']: Particular treatment for the thermodynamic pressure Pth ; there are three possibilities:
 - 1) with the keyword 'edo' the code computes Pth solving an O.D.E. ; in this case, the mass is not strictly conserved (it is the default case for quasi compressible computation);
 - 2) the keyword 'conservation_masse' forces the conservation of the mass (closed geometry or with periodic boundaries condition)
 - 3) the keyword 'constant' makes it possible to have a constant Pth ; it's the good choice when the flow is open (e.g. with pressure boundary conditions).
 It is possible to monitor the volume averaged value for temperature and density, plus Pth evolution in the .evol_glob file.
- **traitement_rho_gravite** *str* into ['standard', 'moins_rho_moyen']: It may be :1) 'standard': the gravity term is evaluated with $\rho \cdot g$ (It is the default). 2) 'moins_rho_moyen': the gravity term is evaluated with $(\rho - \rho_{\text{moy}}) \cdot g$. Unknown pressure is then $P^* = P + \rho_{\text{moy}} \cdot g \cdot z$. It is useful when you apply uniform pressure boundary condition like $P^* = 0$.
- **temps_debut_prise_en_compte_drho_dt** *float*: While time < value, $d\rho/dt$ is set to zero (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.
- **lambda** *champ_base* (16.1): Conductivity ($\text{W.m}^{-1}.\text{K}^{-1}$).
- **mu** *champ_base* (16.1): Dynamic viscosity ($\text{kg.m}^{-1}.\text{s}^{-1}$).
- **indice** *champ_base* (16.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (16.1) for inheritance: Absorptivity of fluid (m^{-1}).
- **gravite** *champ_base* (16.1) for inheritance: Gravity field (optional).
- **porosites_champ** *champ_base* (16.1) for inheritance: The porosity is given at each element and the porosity at each face, $\text{Psi}(\text{face})$, is calculated by the average of the porosities of the two neighbour elements $\text{Psi}(\text{elem1})$, $\text{Psi}(\text{elem2})$: $\text{Psi}(\text{face}) = 2 / (1/\text{Psi}(\text{elem1}) + 1/\text{Psi}(\text{elem2}))$. This keyword is optional.
- **diametre_hyd_champ** *champ_base* (16.1) for inheritance: Hydraulic diameter field (optional).
- **porosites** *porosites* (28) for inheritance: Porosities.
- **rho** *champ_base* (16.1) for inheritance: Density (kg.m^{-3}).
- **cp** *champ_base* (16.1) for inheritance: Specific heat ($\text{J.kg}^{-1}.\text{K}^{-1}$).

22.7 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 $\lambda(T) = \mu_0 \cdot \text{Cp} / \text{Prandtl} \cdot ((T_0 + S\lambda)/(T + S\lambda)) \cdot (T/T_0)^{1.5}$

See also: [objet_lecture \(39\)](#)

Usage:

problem_name mu0 mu0_val t0 t0_val [Slambda] [s] C c_val

where

- **problem_name** *str*: Name of problem.
- **mu0** *str* into ['mu0']
- **mu0_val** *float*
- **t0** *str* into ['T0']
- **t0_val** *float*
- **Slambda** *str* into ['Slambda']
- **s** *float*
- **C** *str* into ['C']
- **c_val** *float*

22.8 Fluide_reel_base

Description: Class for real fluids.

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

See also: [fluide_base \(22.2\)](#) [fluide_sodium_gaz \(22.9\)](#) [fluide_stiffened_gas \(22.11\)](#) [fluide_sodium_liquide \(22.10\)](#)

Usage:

fluide_reel_base *str*

Read *str* {

```
[ indice champ_base]
[ kappa champ_base]
[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]
[ porosites porosites]
[ rho champ_base]
[ lambda champ_base]
[ cp champ_base]
```

}

where

- **indice** *champ_base* [\(16.1\)](#) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* [\(16.1\)](#) for inheritance: Absorptivity of fluid (m-1).
- **gravite** *champ_base* [\(16.1\)](#) for inheritance: Gravity field (optional).
- **porosites_champ** *champ_base* [\(16.1\)](#) for inheritance: The porosity is given at each element and the porosity at each face, Psi(face), is calculated by the average of the porosities of the two neighbour elements Psi(elem1), Psi(elem2) : Psi(face)=2/(1/Psi(elem1)+1/Psi(elem2)). This keyword is optional.
- **diametre_hyd_champ** *champ_base* [\(16.1\)](#) for inheritance: Hydraulic diameter field (optional).
- **porosites** *porosites* [\(28\)](#) for inheritance: Porosities.
- **rho** *champ_base* [\(16.1\)](#) for inheritance: Density (kg.m-3).
- **lambda** *champ_base* [\(16.1\)](#) for inheritance: Conductivity (W.m-1.K-1).
- **cp** *champ_base* [\(16.1\)](#) for inheritance: Specific heat (J.kg-1.K-1).

22.9 **Fluide_sodium_gaz**

Description: Class for `Fluide_sodium_liquide`

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

See also: `fluide_reel_base` ([22.8](#))

Usage:

fluide_sodium_gaz *str*

Read *str* {

```
[ P_ref float]
[ T_ref float]
[ indice champ_base]
[ kappa champ_base]
[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]
[ porosites porosites]
[ rho champ_base]
[ lambda champ_base]
[ cp champ_base]
```

}

where

- **P_ref** *float*: Use to set the pressure value in the closure law. If not specified, the value of the pressure unknown will be used
- **T_ref** *float*: Use to set the temperature value in the closure law. If not specified, the value of the temperature unknown will be used
- **indice** *champ_base* ([16.1](#)) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* ([16.1](#)) for inheritance: Absorptivity of fluid (m-1).
- **gravite** *champ_base* ([16.1](#)) for inheritance: Gravity field (optional).
- **porosites_champ** *champ_base* ([16.1](#)) for inheritance: 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 $\Psi(\text{elem1})$, $\Psi(\text{elem2})$: $\Psi(\text{face}) = 2 / (1/\Psi(\text{elem1}) + 1/\Psi(\text{elem2}))$. This keyword is optional.
- **diametre_hyd_champ** *champ_base* ([16.1](#)) for inheritance: Hydraulic diameter field (optional).
- **porosites** *porosites* ([28](#)) for inheritance: Porosities.
- **rho** *champ_base* ([16.1](#)) for inheritance: Density (kg.m-3).
- **lambda** *champ_base* ([16.1](#)) for inheritance: Conductivity (W.m-1.K-1).
- **cp** *champ_base* ([16.1](#)) for inheritance: Specific heat (J.kg-1.K-1).

22.10 **Fluide_sodium_liquide**

Description: Class for `Fluide_sodium_liquide`

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

See also: `fluide_reel_base` ([22.8](#))

Usage:

fluide_sodium_liquide *str*

Read *str* {

```
[ P_ref float]
```



```

[ T_ref float]
[ indice champ_base]
[ kappa champ_base]
[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]
[ porosites porosites]
[ rho champ_base]
[ lambda champ_base]
[ cp champ_base]
}
where

```

- **P_ref** float: Use to set the pressure value in the closure law. If not specified, the value of the pressure unknown will be used
- **T_ref** float: Use to set the temperature value in the closure law. If not specified, the value of the temperature unknown will be used
- **indice** champ_base (16.1) for inheritance: Refractivity of fluid.
- **kappa** champ_base (16.1) for inheritance: Absorptivity of fluid (m-1).
- **gravite** champ_base (16.1) for inheritance: Gravity field (optional).
- **porosites_champ** champ_base (16.1) for inheritance: 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 $\Psi(\text{elem1})$, $\Psi(\text{elem2})$: $\Psi(\text{face}) = 2 / (1/\Psi(\text{elem1}) + 1/\Psi(\text{elem2}))$. This keyword is optional.
- **diametre_hyd_champ** champ_base (16.1) for inheritance: Hydraulic diameter field (optional).
- **porosites** porosites (28) for inheritance: Porosities.
- **rho** champ_base (16.1) for inheritance: Density (kg.m-3).
- **lambda** champ_base (16.1) for inheritance: Conductivity (W.m-1.K-1).
- **cp** champ_base (16.1) for inheritance: Specific heat (J.kg-1.K-1).

22.11 **Fluide_stiffened_gas**

Description: Class for Stiffened Gas

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

See also: **fluide_reel_base** (22.8)

Usage:

fluide_stiffened_gas str

Read str {

```

[ gamma float]
[ pinf float]
[ mu float]
[ lambda float]
[ Cv float]
[ q float]
[ q_prim float]
[ indice champ_base]
[ kappa champ_base]
[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]

```

```

[ porosites porosites]
[ rho champ_base]
[ lambda champ_base]
[ cp champ_base]
}
where

```

- **gamma** *float*: Heat capacity ratio (Cp/Cv)
- **pinf** *float*: Stiffened gas pressure constant (if set to zero, the state law becomes identical to that of perfect gases)
- **mu** *float*: Dynamic viscosity
- **lambda** *float*: Thermal conductivity
- **Cv** *float*: Thermal capacity at constant volume
- **q** *float*: Reference energy
- **q_prim** *float*: Model constant
- **indice** *champ_base* (16.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (16.1) for inheritance: Absorptivity of fluid (m-1).
- **gravite** *champ_base* (16.1) for inheritance: Gravity field (optional).
- **porosites_champ** *champ_base* (16.1) for inheritance: The porosity is given at each element and the porosity at each face, Psi(face), is calculated by the average of the porosities of the two neighbour elements Psi(elem1), Psi(elem2) : Psi(face)=2/(1/Psi(elem1)+1/Psi(elem2)). This keyword is optional.
- **diametre_hyd_champ** *champ_base* (16.1) for inheritance: Hydraulic diameter field (optional).
- **porosites** *porosites* (28) for inheritance: Porosities.
- **rho** *champ_base* (16.1) for inheritance: Density (kg.m-3).
- **lambda** *champ_base* (16.1) for inheritance: Conductivity (W.m-1.K-1).
- **cp** *champ_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).

22.12 **Fluide_weakly_compressible**

Description: Weakly-compressible flow with a low mach number assumption; this means that the thermodynamic pressure (used in state law) can vary in space.

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

See also: **fluide_dilatable_base** (22.3)

Usage:

fluide_weakly_compressible *str*

Read *str* {

```

[ loi_etat loi_etat_base]
[ sutherland bloc_sutherland]
[ traitement_pth str into ['constant']]
[ lambda champ_base]
[ mu champ_base]
[ pression_thermo float]
[ pression_xyz champ_base]
[ use_total_pressure int]
[ use_hydrostatic_pressure int]
[ use_grad_pression_eos int]
[ time_activate_ptot float]
[ indice champ_base]
[ kappa champ_base]

```

```

[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]
[ porosites porosites]
[ rho champ_base]
[ cp champ_base]
}
where

```

- **loi_etat** *loi_etat_base* (19): The state law that will be associated to the Weakly-compressible fluid.
- **sutherland** *bloc_sutherland* (22.7): Sutherland law for viscosity and for conductivity.
- **traitement_pth** *str into ['constant']*: Particular treatment for the thermodynamic pressure Pth ; there is currently one possibility:
1) the keyword 'constant' makes it possible to have a constant Pth but not uniform in space ; it's the good choice when the flow is open (e.g. with pressure boundary conditions).
- **lambda** *champ_base* (16.1): Conductivity (W.m-1.K-1).
- **mu** *champ_base* (16.1): Dynamic viscosity (kg.m-1.s-1).
- **pression_thermo** *float*: Initial thermo-dynamic pressure used in the associated state law.
- **pression_xyz** *champ_base* (16.1): Initial thermo-dynamic pressure used in the associated state law. It should be defined with as a Champ_Fonc_xyz.
- **use_total_pressure** *int*: Flag (0 or 1) used to activate and use the total pressure in the associated state law. The default value of this Flag is 0.
- **use_hydrostatic_pressure** *int*: Flag (0 or 1) used to activate and use the hydro-static pressure in the associated state law. The default value of this Flag is 0.
- **use_grad_pression_eos** *int*: Flag (0 or 1) used to specify whether or not the gradient of the thermo-dynamic pressure will be taken into account in the source term of the temperature equation (case of a non-uniform pressure). The default value of this Flag is 1 which means that the gradient is used in the source.
- **time_activate_ptot** *float*: Time (in seconds) at which the total pressure will be used in the associated state law.
- **indice** *champ_base* (16.1) for inheritance: Refractivity of fluid.
- **kappa** *champ_base* (16.1) for inheritance: Absorptivity of fluid (m-1).
- **gravite** *champ_base* (16.1) for inheritance: Gravity field (optional).
- **porosites_champ** *champ_base* (16.1) for inheritance: The porosity is given at each element and the porosity at each face, Psi(face), is calculated by the average of the porosities of the two neighbour elements Psi(elem1), Psi(elem2) : $\text{Psi}(\text{face}) = 2 / (1/\text{Psi}(\text{elem1}) + 1/\text{Psi}(\text{elem2}))$. This keyword is optional.
- **diametre_hyd_champ** *champ_base* (16.1) for inheritance: Hydraulic diameter field (optional).
- **porosites** *porosites* (28) for inheritance: Porosities.
- **rho** *champ_base* (16.1) for inheritance: Density (kg.m-3).
- **cp** *champ_base* (16.1) for inheritance: Specific heat (J.kg-1.K-1).

22.13 Solide

Description: Solid with cp and/or rho non-uniform.

See also: milieu_base (22)

Usage:

solide *str*

Read *str* {

```

[ rho champ_base]

```

```

[ cp champ_base]
[ lambda champ_base]
[ user_field champ_base]
[ gravite champ_base]
[ porosites_champ champ_base]
[ diametre_hyd_champ champ_base]
[ porosites porosites]
}
where

```

- **rho** *champ_base* (16.1): Density (kg.m-3).
- **cp** *champ_base* (16.1): Specific heat (J.kg-1.K-1).
- **lambda** *champ_base* (16.1): Conductivity (W.m-1.K-1).
- **user_field** *champ_base* (16.1): user defined field.
- **gravite** *champ_base* (16.1) for inheritance: Gravity field (optional).
- **porosites_champ** *champ_base* (16.1) for inheritance: 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 $\Psi(\text{elem1})$, $\Psi(\text{elem2})$: $\Psi(\text{face}) = 2 / (1/\Psi(\text{elem1}) + 1/\Psi(\text{elem2}))$. This keyword is optional.
- **diametre_hyd_champ** *champ_base* (16.1) for inheritance: Hydraulic diameter field (optional).
- **porosites** *porosites* (28) for inheritance: Porosities.

23 modele_turbulence_scal_base

Description: Basic class for turbulence model for energy equation.

See also: **objet_u** (40) **schmidt** (23.4) **null** (23.2) **prandtl** (23.3)

Usage:

modele_turbulence_scal_base *str*

Read *str* {

```

[ dt_impr_nusselt float]
[ dt_impr_nusselt_mean_only dt_impr_nusselt_mean_only]
[ turbulence_paro turbulence_paro_scalaire_base]

```

}

where

- **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».
- **dt_impr_nusselt_mean_only** *dt_impr_nusselt_mean_only* (23.1): This keyword is used to print the mean values of Nusselt (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_nusselt_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, then you have to specify their names.
- **turbulence_paro** *turbulence_paro_scalaire_base* (37): Keyword to set the wall law.

23.1 Dt_impr_nusselt_mean_only

Description: not_set

See also: objet_lecture (39)

Usage:

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

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

23.2 Null

Description: Null scalar turbulence model (turbulent diffusivity = 0) which can be used with a turbulent problem.

See also: modele_turbulence_scal_base (23)

Usage:

```
null str  
Read str {  
  
    [ dt_impr_nusselt float]  
    [ dt_impr_nusselt_mean_only dt_impr_nusselt_mean_only]  
  
}  
where
```

- **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».
- **dt_impr_nusselt_mean_only** dt_impr_nusselt_mean_only (23.1) for inheritance: This keyword is used to print the mean values of Nusselt (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_nusselt_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, then you have to specify their names.

23.3 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 *str*

Read *str* {

```
[ prdt str]  
[ prandt_turbulent_fonction_nu_t_alpha str]  
[ dt_impr_nusselt float]  
[ dt_impr_nusselt_mean_only dt_impr_nusselt_mean_only]  
[ turbulence_paro turbulence_paro_scalaire_base]
```

}

where

- **prdt** *str*: Keyword to modify the constant (Prdt) of Prandtl model : $\text{Alphat} = \text{Nut} / \text{Prdt}$ Default value is 0.9
- **prandt_turbulent_fonction_nu_t_alpha** *str*: Optional keyword to specify turbulent diffusivity (by default, $\alpha_t = \nu_t / \text{Prt}$) with another formulae, for example: $\alpha_t = \nu_t^2 / (0.7 * \alpha + 0.85 * \nu_t)$ with the string $\nu_t * \nu_t / (0.7 * \alpha + 0.85 * \nu_t)$ where α is the thermal diffusivity.
- **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».
- **dt_impr_nusselt_mean_only** *dt_impr_nusselt_mean_only* (23.1) for inheritance: This keyword is used to print the mean values of Nusselt (obtained with the wall laws) on each boundary, into a file named datafile_ProblemName_nusselt_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, then you have to specify their names.
- **turbulence_paro** *turbulence_paro_scalaire_base* (37) for inheritance: Keyword to set the wall law.

23.4 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, Schmidt 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 str
Read str {
    [ scturb float]
    [ dt_impr_nusselt float]
    [ dt_impr_nusselt_mean_only dt_impr_nusselt_mean_only]
    [ turbulence_paroi turbulence_paroi_scalaire_base]
}
where

```

- **scturb** *float*: Keyword to modify the constant (Sct) of Schmidt model : $Dt=Nut/Sct$ Default value is 0.7.
- **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».
- **dt_impr_nusselt_mean_only** *dt_impr_nusselt_mean_only* (23.1) for inheritance: This keyword is used to print the mean values of Nusselt (obtained with the wall laws) on each boundary, into a file named `datafile_ProblemName_nusselt_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, then you have to specify their names.
- **turbulence_paro**i *turbulence_paro*i_scalaire_base (37) for inheritance: Keyword to set the wall law.

24 moyenne_imposee_deriv

Description: not_set

See also: `objet_u` (40) `profil` (24.5) `connexion_exacte` (24.2) `connexion_approchee` (24.1) `interpolation` (24.3) `logarithmique` (24.4)

Usage:

24.1 Connexion_approchee

Description: To read the imposed field from a file where positions and values are given (it is not necessary that the coordinates of points match the coordinates of the boundary faces, indeed, the nearest point of each face of the boundary will be used).

See also: `moyenne_imposee_deriv` (24)

Usage:

```

connexion_approchee fichier file1
where

```

- **fichier** *str* into [*'fichier'*]

- **file1** *str*: filename. 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)]
```

24.2 Connexion_exacte

Description: To read the imposed field from two files.

See also: [moyenne_imposee_deriv \(24\)](#)

Usage:

connexion_exacte fichier file1 [file2]

where

- **fichier** *str* into [*'fichier'*]
- **file1** *str*: first file, contains the points coordinates (which should be the same as the coordinates of the boundary faces). The format of this file is:

```
N
1 x(1) y(1) [z(1)]
2 x(2) y(2) [z(2)]
...
N x(N) y(N) [z(N)]
```

- **file2** *str*: second file, contains the mean values. The format of this file is:

```
N
1 valx(1) valy(1) [valz(1)]
2 valx(2) valy(2) [valz(2)]
...
N valx(N) valy(N) [valz(N)]
```

24.3 Interpolation

Synonymous: **champ_post_interpolation**

Description: To create an imposed field built by interpolation of values read from a file. The imposed field is applied on the direction given by the keyword `direction_anisotrope` (the field is zero for the other directions).

See also: [moyenne_imposee_deriv \(24\)](#)

Usage:

interpolation fichier file1

where

- **fichier** *str* into [*'fichier'*]: 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.

- **file1** *str*: name of geom_face_perio

24.4 Logarithmique

Description: To specify the imposed field (in this case, velocity) by an analytical logarithmic law of the wall:

$$g(x,y,z) = u_tau * (\log(0.5*diametre*u_tau/visco_cin)/Kappa + 5.1)$$

with $g(x,y,z)=u(x,y,z)$ if direction is set to 1, $g=v(x,y,z)$ if direction is set to 2 and $g=w(x,y,z)$ if it is set to 3

See also: [moyenne_imposee_deriv \(24\)](#)

Usage:

logarithmique **diametre** **val** **u_tau** **val_u_tau** **visco_cin** **val_visco_cin** **direction** **val_direction**
where

- **diametre** *str* into ['diametre']
- **val** *float*: diameter
- **u_tau** *str* into ['u_tau']
- **val_u_tau** *float*: value of u_tau
- **visco_cin** *str* into ['visco_cin']
- **val_visco_cin** *float*: value of visco_cin
- **direction** *str* into ['direction']
- **val_direction** *int*: direction

24.5 Profil

Description: To specify analytic profile for the imposed g field.

See also: [moyenne_imposee_deriv \(24\)](#)

Usage:

profil **profile**
where

- **profile** *n word1 word2 ... wordn*: specifies the analytic profile: 2|3 valx(x,y,z,t) valy(x,y,z,t) [valz(x,y,z,t)]

25 nom

Description: Class to name the TRUST objects.

See also: [objet_u \(40\)](#) [nom_anonyme \(25.1\)](#)

Usage:

nom [**mot**]
where

- **mot** *str*: Chain of characters.

25.1 Nom_anonyme

Description: not_set

See also: nom ([25](#))

Usage:

[**mot**]

where

- **mot** *str*: Chain of characters.

26 partitionneur_deriv

Description: not_set

See also: objet_u ([40](#)) metis ([26.3](#)) fichier_med ([26.1](#)) sous_dom ([26.5](#)) partition ([26.4](#)) union ([26.8](#)) tranche ([26.7](#)) sous_zones ([26.6](#)) fichier_decoupage ([26.2](#))

Usage:

partitionneur_deriv *str*

Read *str* {

 [**nb_parts** *int*]

}

where

- **nb_parts** *int*: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.1 Fichier_med

Description: Partitioning a domain using a MED file containing an integer field providing for each element the processor number on which the element should be located.

See also: partitionneur_deriv ([26](#))

Usage:

fichier_med *str*

Read *str* {

file *str*

 [**field** *str*]

 [**nb_parts** *int*]

}

where

- **file** *str*: file name of the MED file to load
- **field** *str*: field name of the integer (or double) field to load
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.2 Fichier_decoupage

Description: This algorithm reads an array of integer values on the disc, one value for each mesh element. Each value is interpreted as the target part number $n \geq 0$ for this element. The number of parts created is the highest value in the array plus one. Empty parts can be created if some values are not present in the array.

The file format is ASCII, and contains space, tab or carriage-return separated integer values. The first value is the number `nb_elem` of elements in the domain, followed by `nb_elem` integer values (positive or zero).

This algorithm has been designed to work together with the `'ecrire_decoupage'` option. You can generate a partition with any other algorithm, write it to disc, modify it, and read it again to generate the `.Zone` files. Contrary to other partitioning algorithms, no correction is applied by default to the partition (eg. element 0 on processor 0 and corrections for periodic boundaries). If `'corriger_partition'` is specified, these corrections are applied.

See also: `partitionneur_deriv` (26)

Usage:

fichier_decoupage *str*

Read *str* {

fichier *str*

 [**corriger_partition**]

 [**nb_parts** *int*]

}

where

- **fichier** *str*: File name
- **corriger_partition**
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.3 Metis

Description: Metis is an external partitioning library. It is a general algorithm that will generate a partition of the domain.

See also: `partitionneur_deriv` (26)

Usage:

metis *str*

Read *str* {

 [**kmetis**]

 [**use_weights**]

 [**nb_parts** *int*]

}

where

- **kmetis** : The default values are `pmetis`, default parameters are automatically chosen by Metis. `'kmetis'` is faster than `pmetis` option but the last option produces better partitioning quality. In both cases, the partitioning quality may be slightly improved by increasing the `nb_essais` option (by default `N=1`). It will compute `N` partitions and will keep the best one (smallest edge cut number). But this option is CPU expensive, taking `N=10` will multiply the CPU cost of partitioning by 10. Experiments show that only marginal improvements can be obtained with non default parameters.

- **use_weights** : If use_weights is specified, weighting of the element-element links in the graph is used to force metis to keep opposite periodic elements on the same processor. This option can slightly improve the partitionning quality but it consumes more memory and takes more time. It is not mandatory since a correction algorithm is always applied afterwards to ensure a correct partitionning for periodic boundaries.
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.4 Partition

Synonymous: **decouper**

Description: This algorithm re-use the partition of the domain named `DOMAINE_NAME`. It is useful to partition for example a post processing domain. The partition should match with the calculation domain.

See also: `partitionneur_deriv` (26)

Usage:

```
partition str
Read str {
    domaine str
    [ nb_parts int ]
}
where
```

- **domaine** *str*: domain name
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.5 Sous_dom

Description: Given a global partition of a global domain, 'sous-domaine' allows to produce a conform partition of a sub-domain generated from the bigger one using the keyword `create_domain_from_sub_domain`. The sub-domain will be partitionned in a conform fashion with the global domain.

See also: `partitionneur_deriv` (26)

Usage:

```
sous_dom str
Read str {
    fichier str
    [ fichier_ssz str ]
    [ name_ssz str ]
    [ nb_parts int ]
}
where
```

- **fichier** *str*: fichier
- **fichier_ssz** *str*: fichier sous zone
- **name_ssz** *str*: nom sous zone
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.6 Sous_zones

Description: This algorithm will create one part for each specified subdomaine/domain. All elements contained in the first subdomaine/domain are put in the first part, all remaining elements contained in the second subdomaine/domain in the second part, etc...

If all elements of the current domain are contained in the specified subdomains/domain, then N parts are created, otherwise, a supplemental part is created with the remaining elements.

If no subdomaine is specified, all subdomains defined in the domain are used to split the mesh.

See also: `partitionneur_deriv` (26)

Usage:

sous_zones *str*

Read *str* {

 [**sous_zones** *n word1 word2 ... wordn*]

 [**domaines** *n word1 word2 ... wordn*]

 [**nb_parts** *int*]

}

where

- **sous_zones** *n word1 word2 ... wordn*: N SUBZONE_NAME_1 SUBZONE_NAME_2 ...
- **domaines** *n word1 word2 ... wordn*: N DOMAIN_NAME_1 DOMAIN_NAME_2 ...
- **nb_parts** *int* for inheritance: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

26.7 Tranche

Description: This algorithm will create a geometrical partitionning by slicing the mesh in the two or three axis directions, based on the geometric center of each mesh element. *nz* must be given if dimension=3. Each slice contains the same number of elements (slices don't have the same geometrical width, and for VDF meshes, slice boundaries are generally not flat except if the number of mesh elements in each direction is an exact multiple of the number of slices). First, *nx* slices in the X direction are created, then each slice is split in *ny* slices in the Y direction, and finally, each part is split in *nz* slices in the Z direction. The resulting number of parts is *nx*ny*nz*. If one particular direction has been declared periodic, the default slicing (0, 1, 2, ..., *n-1*) is replaced by (0, 1, 2, ..., *n-1*, 0), each of the two '0' slices having twice less elements than the other slices.

See also: `partitionneur_deriv` (26)

Usage:

tranche *str*

Read *str* {

 [**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).

26.8 Union

Description: Let several local domains be generated from a bigger one using the keyword `create_domain-from_sub_domain`, and let their partitions be generated in the usual way. Provided the list of partition files for each small domain, the keyword 'union' will partition the global domain in a conform fashion with the smaller domains.

See also: `partitionneur_deriv` (26)

Usage:

union **liste** [**nb_parts**]

where

- **liste** *bloc_lecture* (3.60): List of the partition files with the following syntaxe: {sous_domaine1 decoupage1 ... sous_domaineim decoupageim } where sous_domaine1 ... sous_zomeim are small domains names and decoupage1 ... decoupageim are partition files.
- **nb_parts** *int*: The number of non empty parts that must be generated (generally equal to the number of processors in the parallel run).

27 pb_champ_evaluateur

Description: specifies problem name, the field name belonging to the problem and number of field components.

See also: `objet_u` (40)

Usage:

pb **champ** **ncomp**

where

- **pb** *str*: name of the problem where the source fields will be searched.
- **champ** *str*: name of the field
- **ncomp** *int*: number of components

28 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: `objet_u` (40)

Usage:

porosites **aco** **sous_zone1**|**sous_zone** **bloc** [**sous_zone2**] [**bloc2**] **acof**

where

- **aco** *str* into ['{']: Opening curly bracket.
- **sous_zone1**|**sous_zone** *str*: Name of the sub-area to which porosity are allocated.

- **bloc** *bloc_lecture_poro* (28.1): Surface and volume porosity values.
- **sous_zone2** *str*: Name of the 2nd sub-area to which porosity are allocated.
- **bloc2** *bloc_lecture_poro* (28.1): Surface and volume porosity values.
- **acof** *str* into `[]`: Closing curly bracket.

28.1 Bloc_lecture_poro

Description: Surface and volume porosity values.

See also: `objet_lecture` (39)

Usage:

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

where

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

29 precondition_base

Description: Basic class for preconditioning.

See also: `objet_u` (40) `ilu` (29.1) `ssor_bloc` (29.4) `precondsolv` (29.2) `ssor` (29.3)

Usage:

29.1 Ilu

Description: This preconditionner can be only used with the generic GEN solver.

See also: `precond_base` (29)

Usage:

```
ilu str
Read str {
    [ type int]
    [ filling int]
}
```

where

- **type** *int*: values can be 0|1|2|3 for null|left|right|left-and-right preconditionning (default value = 2)
- **filling** *int*: default value = 1.

29.2 Precondsolv

Description: not_set

See also: `precond_base` (29)

Usage:

precondsolv *solveur*

where

- **solveur** *solveur_sys_base* (11.16): Solver type.

29.3 Ssor

Description: Symmetric successive over-relaxation algorithm.

See also: `precond_base` (29)

Usage:

ssor *str*

Read *str* {

 [**omega** *float*]

}

where

- **omega** *float*: Over-relaxation facteur (between 1 and 2, default value 1.6).

29.4 Ssor_bloc

Description: not_set

See also: `precond_base` (29)

Usage:

ssor_bloc *str*

Read *str* {

 [**precond0** *precond_base*]

 [**precond1** *precond_base*]

 [**preconda** *precond_base*]

 [**alpha_0** *float*]

 [**alpha_1** *float*]

 [**alpha_a** *float*]

}

where

- **precond0** *precond_base* (29)
- **precond1** *precond_base* (29)
- **preconda** *precond_base* (29)
- **alpha_0** *float*
- **alpha_1** *float*
- **alpha_a** *float*

30 preconditionneur_petsc_deriv

Description: Preconditioners available with petsc solvers

See also: [objet_u \(40\)](#) [diag \(30.6\)](#) [c-amg \(30.5\)](#) [sa-amg \(30.11\)](#) [BLOCK_JACOBI_ICC \(30.1\)](#) [boomer-amg \(30.4\)](#) [null \(30.9\)](#) [lu \(30.8\)](#) [jacobi \(30.7\)](#) [EISENTAT \(30.2\)](#) [ssor \(30.13\)](#) [block_jacobi_ilu \(30.3\)](#) [spai \(30.12\)](#) [pilut \(30.10\)](#)

Usage:

30.1 Block_jacobi_icc

Description: Incomplete Cholesky factorization for symmetric matrix with the PETSc implementation.

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

Usage:

BLOCK_JACOBI_ICC *str*

Read *str* {

 [**level** *int*]

 [**ordering** *str* into [*'natural'*, *'rcm'*]]

}

where

- **level** *int*: factorization level (default value, 1). In parallel, the factorization is done by block (one per processor by default).
- **ordering** *str* into [*'natural'*, *'rcm'*]: The ordering of the local matrix is natural by default, but rcm ordering, which reduces the bandwidth of the local matrix, may interestingly improves the quality of the decomposition and reduces the number of iterations.

30.2 Eisentat

Description: SSOR version with Eisenstat trick which reduces the number of computations and thus CPU cost...

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

Usage:

EISENTAT *str*

Read *str* {

 [**omega** *float*]

}

where

- **omega** *float*: relaxation factor

30.3 Block_jacobi_ilu

Description: preconditionner

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

Usage:

block_jacobi_ilu *str*

Read *str* {

 [**level** *int*]

}

where

- **level** *int*

30.4 Boomeramg

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

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

Usage:

30.5 C-amg

Description: preconditionner

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

Usage:

30.6 Diag

Description: Diagonal (Jacobi) preconditioner.

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

Usage:

30.7 Jacobi

Description: preconditionner

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

Usage:

30.8 Lu

Description: preconditionner

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

Usage:

30.9 Null

Description: No preconditioner used

See also: `preconditionneur_petsc_deriv` (30)

Usage:

30.10 Pilut

Description: Dual Threshold Incomplete LU factorization.

See also: `preconditionneur_petsc_deriv` (30)

Usage:

pilut *str*

Read *str* {

 [**level** *int*]

 [**epsilon** *float*]

}

where

- **level** *int*: factorization level
- **epsilon** *float*: drop tolerance

30.11 Sa-amg

Description: preconditionner

See also: `preconditionneur_petsc_deriv` (30)

Usage:

30.12 Spai

Description: Spai Approximate Inverse algorithm from Parasails Hypr library.

See also: `preconditionneur_petsc_deriv` (30)

Usage:

spai *str*

Read *str* {

 [**level** *int*]

 [**epsilon** *float*]

}

where

- **level** *int*: first parameter
- **epsilon** *float*: second parameter

30.13 Ssor

Description: Symmetric Successive Over Relaxation algorithm.

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

Usage:

ssor *str*

Read *str* {

 [**omega** *float*]

}

where

- **omega** *float*: relaxation factor (default value, 1.5)

31 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 \(40\)](#) [Sch_CN_iteratif \(31.2\)](#) [schema_implicite_base \(31.20\)](#) [runge_kutta_ordre_2 \(31.5\)](#) [runge_kutta_ordre_3 \(31.7\)](#) [runge_kutta_ordre_4_d3p \(31.9\)](#) [runge_kutta_rationnel_ordre_2 \(31.12\)](#) [schema_predictor_corrector \(31.21\)](#) [runge_kutta_ordre_2_classique \(31.6\)](#) [runge_kutta_ordre_3_classique \(31.8\)](#) [runge_kutta_ordre_4_classique \(31.10\)](#) [runge_kutta_ordre_4_classique_3_8 \(31.11\)](#) [scheme_euler_explicit \(31.3\)](#) [leap_frog \(31.4\)](#) [schema_adams_bashforth_order_2 \(31.13\)](#) [schema_adams_bashforth_order_3 \(31.14\)](#)

Usage:

schema_temps_base *str*

Read *str* {

 [**tinit** *float*]

 [**tmax** *float*]

 [**tcpumax** *float*]

 [**dt_min** *float*]

 [**dt_max** *str*]

 [**dt_sauv** *float*]

 [**nb_sauv_max** *int*]

 [**dt_impr** *float*]

 [**facsec** *str*]

 [**seuil_statio** *float*]

 [**residuals** *residuals*]

 [**diffusion_implicite** *int*]

 [**seuil_diffusion_implicite** *float*]

 [**impr_diffusion_implicite** *int*]

 [**impr_extremums** *int*]

 [**no_error_if_not_converged_diffusion_implicite** *int*]

 [**no_conv_subiteration_diffusion_implicite** *int*]

 [**dt_start** *dt_start*]

 [**nb_pas_dt_max** *int*]

 [**niter_max_diffusion_implicite** *int*]

 [**precision_impr** *int*]

 [**periode_sauvegarde_securite_en_heures** *float*]

 [**no_check_disk_space**]

```

[ disable_progress ]
[ disable_dt_ev ]
[ gnuplot_header int]
}
where

```

- **tinit** *float*: Value of initial calculation time (0 by default).
- **tmax** *float*: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float*: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float*: Minimum calculation time step (1e-16s by default).
- **dt_max** *str*: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float*: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **nb_sauv_max** *int*: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of nb_sauv_max timesteps in the file), the next checkpoints will overwrite the first ones
- **dt_impr** *float*: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str*: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float*: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108): To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int*: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float*: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int*: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int*: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int*
- **no_conv_subiteration_diffusion_implicit** *int*
- **dt_start** *dt_start* (11.8): dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.

- **nb_pas_dt_max** *int*: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int*: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int*: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float*: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** : To disable the check of the available amount of disk space during the calculation.
- **disable_progress** : To disable the writing of the .progress file.
- **disable_dt_ev** : To disable the writing of the .dt_ev file.
- **gnuplot_header** *int*: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.1 Sch_cn_ex_iteratif

Description: This keyword also describes a Crank-Nicholson method of second order accuracy but here, for scalars, because of instabilities encountered when $dt > dt_{CFL}$, the Crank Nicholson scheme is not applied to scalar quantities. Scalars are treated according to Euler-Explicite scheme at the end of the CN treatment for velocity flow fields (by doing p Euler explicite under-iterations at $dt \leq dt_{CFL}$). Parameters are the same (but default values may change) compare to the Sch_CN_iterative scheme plus a relaxation keyword: niter_min (2 by default), niter_max (6 by default), niter_avg (3 by default), facsec_max (20 by default), seuil (0.05 by default)

See also: Sch_CN_iteratif ([31.2](#))

Usage:

Sch_CN_EX_iteratif *str*

Read *str* {

```
[ omega float]
[ seuil float]
[ niter_min int]
[ niter_max int]
[ niter_avg int]
[ facsec_max float]
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ nb_sauv_max int]
[ dt_impr float]
[ facsec str]
[ seuil_statio float]
[ residuals residuals]
[ diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int]
[ impr_extremums int]
[ no_error_if_not_converged_diffusion_implicit int]
[ no_conv_subiteration_diffusion_implicit int]
```

```

[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures float]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
[ gnuplot_header int]

```

}

where

- **omega** *float*: relaxation factor (0.1 by default)
- **seuil** *float* for inheritance: criteria for ending iterative process ($\text{Max}(\|u(p) - u(p-1)\|/\text{Max}\|u(p)\|) < \text{seuil}$) (0.001 by default)
- **niter_min** *int* for inheritance: minimal number of p-iterations to satisfy convergence criteria (2 by default)
- **niter_max** *int* for inheritance: number of maximum p-iterations allowed to satisfy convergence criteria (6 by default)
- **niter_avg** *int* for inheritance: threshold of p-iterations (3 by default). If the number of p-iterations is greater than niter_avg, facsec is reduced, if lesser than niter_avg, facsec is increased (but limited by the facsec_max value).
- **facsec_max** *float* for inheritance: maximum ratio allowed between dynamical time step returned by iterative process and stability time returned by CFL condition (2 by default).
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcputmax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of nb_sauv_max timesteps in the file), the next checkpoints will overwrite the first ones
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema-Adams-Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).

- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_convection$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_max$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: $dt_start\ dt_min$: the first iteration is based on dt_min .
 $dt_start\ dt_calc$: the time step at first iteration is calculated in agreement with CFL condition.
 $dt_start\ dt_fixe$ value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.2 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 (31) Sch_CN_EX_iteratif (31.1)

Usage:

Sch_CN_iteratif *str*

Read *str* {

```
[ seuil float]  
[ niter_min int]  
[ niter_max int]  
[ niter_avg int]  
[ facsec_max float]  
[ tinit float]  
[ tmax float]  
[ tcpumax float]  
[ dt_min float]  
[ dt_max str]  
[ dt_sauv float]  
[ nb_sauv_max int]  
[ dt_impr float]  
[ facsec str]  
[ seuil_statio float]  
[ residuals residuals]  
[ diffusion_implicite int]  
[ seuil_diffusion_implicite float]  
[ impr_diffusion_implicite int]  
[ impr_extremums int]  
[ no_error_if_not_converged_diffusion_implicite int]  
[ no_conv_subiteration_diffusion_implicite int]  
[ dt_start dt_start]  
[ nb_pas_dt_max int]  
[ niter_max_diffusion_implicite int]  
[ precision_impr int]  
[ periode_sauvegarde_securite_en_heures float]  
[ no_check_disk_space ]  
[ disable_progress ]  
[ disable_dt_ev ]  
[ gnuplot_header int]
```

}

where

- **seuil** *float*: criteria for ending iterative process ($\text{Max}(\|u(p) - u(p-1)\| / \text{Max} \|u(p)\|) < \text{seuil}$) (0.001 by default)
- **niter_min** *int*: minimal number of p-iterations to satisfy convergence criteria (2 by default)
- **niter_max** *int*: number of maximum p-iterations allowed to satisfy convergence criteria (6 by default)
- **niter_avg** *int*: threshold of p-iterations (3 by default). If the number of p-iterations is greater than **niter_avg**, **facsec** is reduced, if lesser than **niter_avg**, **facsec** is increased (but limited by the **facsec_max** value).
- **facsec_max** *float*: maximum ratio allowed between dynamical time step returned by iterative process and stability time returned by CFL condition (2 by default).
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).

- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of nb_sauv_max timesteps in the file), the next checkpoints will overwrite the first ones
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt = facsec * dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt = facsec * dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.3 Scheme_euler_explicit

Synonymous: **schema_euler_explicite**

Description: This is the Euler explicit scheme.

See also: **schema_temps_base** (31)

Usage:

scheme_euler_explicit *str*

Read *str* {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ nb_sauv_max int]
[ dt_impr float]
[ facsec str]
[ seuil_statio float]
[ residuals residuals]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ impr_extremums int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures float]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
[ gnuplot_header int]
```

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).

- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of nb_sauv_max timesteps in the file), the next checkpoints will overwrite the first ones
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dGi/dt of all the unknown transported values Gi have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.

- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.4 Leap_frog

Description: This is the leap-frog scheme.

See also: `schema_temps_base` (31)

Usage:

leap_frog *str*

Read *str* {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ nb_sauv_max int]
[ dt_impr float]
[ facsec str]
[ seuil_statio float]
[ residuals residuals]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ impr_extremums int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures float]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
[ gnuplot_header int]
```

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every `dt_sauv`, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that `dt_sauv` is in terms of physical time (not cpu time).

- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of nb_sauv_max timesteps in the file), the next checkpoints will overwrite the first ones
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: $dt_{start} dt_{min}$: the first iteration is based on dt_{min} .
 $dt_{start} dt_{calc}$: the time step at first iteration is calculated in agreement with CFL condition.
 $dt_{start} dt_{fixe}$ value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_{calc} .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.5 Runge_kutta_ordre_2

Description: This is a low-storage Runge-Kutta scheme of second order that uses 2 integration points. The method is presented by Williamson (case 1) in <https://www.sciencedirect.com/science/article/pii/0021999180900339>

See also: `schema_temps_base` (31)

Usage:

runge_kutta_ordre_2 *str*

```
Read str {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max str]  
    [ dt_sauv float]  
    [ nb_sauv_max int]  
    [ dt_impr float]  
    [ facsec str]  
    [ seuil_statio float]  
    [ residuals residuals]  
    [ diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int]  
    [ impr_extremums int]  
    [ no_error_if_not_converged_diffusion_implicite int]  
    [ no_conv_subiteration_diffusion_implicite int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicite int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures float]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
    [ gnuplot_header int]  
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every `dt_sauv`, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the `.sauv` files, you must specify 0. Note that `dt_sauv` is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of `nb_sauv_max` timesteps in the file), the next checkpoints will overwrite the first ones

- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min .
 dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
 dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.6 Runge_kutta_ordre_2_classique

Description: This is a classical Runge-Kutta scheme of second order that uses 2 integration points.

See also: `schema_temps_base` (31)

Usage:

runge_kutta_ordre_2_classique *str*

```
Read str {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max str]  
    [ dt_sauv float]  
    [ nb_sauv_max int]  
    [ dt_impr float]  
    [ facsec str]  
    [ seuil_statio float]  
    [ residuals residuals]  
    [ diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int]  
    [ impr_extremums int]  
    [ no_error_if_not_converged_diffusion_implicite int]  
    [ no_conv_subiteration_diffusion_implicite int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicite int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures float]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
    [ gnuplot_header int]  
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the `.sauv` files, you must specify 0. Note that **dt_sauv** is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of **nb_sauv_max** timesteps in the file), the next checkpoints will overwrite the first ones

- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min .
 dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
 dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.7 Runge_kutta_ordre_3

Description: This is a low-storage Runge-Kutta scheme of third order that uses 3 integration points. The method is presented by Williamson (case 7) in <https://www.sciencedirect.com/science/article/pii/0021999180900339>

See also: `schema_temps_base` (31)

Usage:

runge_kutta_ordre_3 *str*

```
Read str {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max str]  
    [ dt_sauv float]  
    [ nb_sauv_max int]  
    [ dt_impr float]  
    [ facsec str]  
    [ seuil_statio float]  
    [ residuals residuals]  
    [ diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int]  
    [ impr_extremums int]  
    [ no_error_if_not_converged_diffusion_implicite int]  
    [ no_conv_subiteration_diffusion_implicite int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicite int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures float]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
    [ gnuplot_header int]  
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every `dt_sauv`, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the `.sauv` files, you must specify 0. Note that `dt_sauv` is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of `nb_sauv_max` timesteps in the file), the next checkpoints will overwrite the first ones

- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: $dt_{start} dt_{min}$: the first iteration is based on dt_{min} .
 $dt_{start} dt_{calc}$: the time step at first iteration is calculated in agreement with CFL condition.
 $dt_{start} dt_{fixe}$ value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_{calc} .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.8 Runge_kutta_ordre_3_classique

Description: This is a classical Runge-Kutta scheme of third order that uses 3 integration points.

See also: `schema_temps_base` (31)

Usage:

runge_kutta_ordre_3_classique *str*

```
Read str {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max str]  
    [ dt_sauv float]  
    [ nb_sauv_max int]  
    [ dt_impr float]  
    [ facsec str]  
    [ seuil_statio float]  
    [ residuals residuals]  
    [ diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int]  
    [ impr_extremums int]  
    [ no_error_if_not_converged_diffusion_implicite int]  
    [ no_conv_subiteration_diffusion_implicite int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicite int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures float]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
    [ gnuplot_header int]  
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the `.sauv` files, you must specify 0. Note that **dt_sauv** is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of **nb_sauv_max** timesteps in the file), the next checkpoints will overwrite the first ones

- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min .
 dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
 dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.9 Runge_kutta_ordre_4_d3p

Synonymous: **runge_kutta_ordre_4**

Description: This is a low-storage Runge-Kutta scheme of fourth order that uses 3 integration points. The method is presented by Williamson (case 17) in <https://www.sciencedirect.com/science/article/pii/0021999180900339>

See also: `schema_temps_base` (31)

Usage:

runge_kutta_ordre_4_d3p *str*

Read *str* {

```
[ tinit float]  
[ tmax float]  
[ tcpumax float]  
[ dt_min float]  
[ dt_max str]  
[ dt_sauv float]  
[ nb_sauv_max int]  
[ dt_impr float]  
[ facsec str]  
[ seuil_statio float]  
[ residuals residuals]  
[ diffusion_implicite int]  
[ seuil_diffusion_implicite float]  
[ impr_diffusion_implicite int]  
[ impr_extremums int]  
[ no_error_if_not_converged_diffusion_implicite int]  
[ no_conv_subiteration_diffusion_implicite int]  
[ dt_start dt_start]  
[ nb_pas_dt_max int]  
[ niter_max_diffusion_implicite int]  
[ precision_impr int]  
[ periode_sauvegarde_securite_en_heures float]  
[ no_check_disk_space ]  
[ disable_progress ]  
[ disable_dt_ev ]  
[ gnuplot_header int]
```

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every `dt_sauv`, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the `.sauv` files, you must specify 0. Note that `dt_sauv` is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with

parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of `nb_sauv_max` timesteps in the file), the next checkpoints will overwrite the first ones

- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: `dt_start dt_min` : the first iteration is based on `dt_min`.
`dt_start dt_calc` : the time step at first iteration is calculated in agreement with CFL condition.
`dt_start dt_fixe` value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on `dt_calc`.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.10 Runge_kutta_ordre_4_classique

Description: This is a classical Runge-Kutta scheme of fourth order that uses 4 integration points.

See also: `schema_temps_base` (31)

Usage:

runge_kutta_ordre_4_classique *str*

```
Read str {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max str]  
    [ dt_sauv float]  
    [ nb_sauv_max int]  
    [ dt_impr float]  
    [ facsec str]  
    [ seuil_statio float]  
    [ residuals residuals]  
    [ diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int]  
    [ impr_extremums int]  
    [ no_error_if_not_converged_diffusion_implicite int]  
    [ no_conv_subiteration_diffusion_implicite int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicite int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures float]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
    [ gnuplot_header int]  
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the `.sauv` files, you must specify 0. Note that **dt_sauv** is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of **nb_sauv_max** timesteps in the file), the next checkpoints will overwrite the first ones

- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min .
 dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
 dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.11 Runge_kutta_ordre_4_classique_3_8

Description: This is a classical Runge-Kutta scheme of fourth order that uses 4 integration points and the 3/8 rule.

See also: `schema_temps_base` (31)

Usage:

runge_kutta_ordre_4_classique_3_8 *str*

```
Read str {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max str]  
    [ dt_sauv float]  
    [ nb_sauv_max int]  
    [ dt_impr float]  
    [ facsec str]  
    [ seuil_statio float]  
    [ residuals residuals]  
    [ diffusion_implicite int]  
    [ seuil_diffusion_implicite float]  
    [ impr_diffusion_implicite int]  
    [ impr_extremums int]  
    [ no_error_if_not_converged_diffusion_implicite int]  
    [ no_conv_subiteration_diffusion_implicite int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicite int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures float]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
    [ gnuplot_header int]  
}
```

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every `dt_sauv`, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the `.sauv` files, you must specify 0. Note that `dt_sauv` is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of `nb_sauv_max` timesteps in the file), the next checkpoints will overwrite the first ones

- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min .
 dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
 dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.12 Runge_kutta_rationnel_ordre_2

Description: This is the Runge-Kutta rational scheme of second order. The method is described in the note: Wambeck - Rational Runge-Kutta methods for solving systems of ordinary differential equations, at the link: <https://link.springer.com/article/10.1007/BF02252381>. Although rational methods require more computational work than linear ones, they can have some other properties, such as a stable behaviour with explicitness, which make them preferable. The CFD application of this RRK2 scheme is described in the note: https://link.springer.com/content/pdf/10.1007%2F3-540-13917-6_112.pdf.

See also: `schema_temps_base` (31)

Usage:

runge_kutta_rationnel_ordre_2 *str*

```
Read str {  
    [ tinit float]  
    [ tmax float]  
    [ tcpumax float]  
    [ dt_min float]  
    [ dt_max str]  
    [ dt_sauv float]  
    [ nb_sauv_max int]  
    [ dt_impr float]  
    [ facsec str]  
    [ seuil_statio float]  
    [ residuals residuals]  
    [ diffusion_implicit int]  
    [ seuil_diffusion_implicit float]  
    [ impr_diffusion_implicit int]  
    [ impr_extremums int]  
    [ no_error_if_not_converged_diffusion_implicit int]  
    [ no_conv_subiteration_diffusion_implicit int]  
    [ dt_start dt_start]  
    [ nb_pas_dt_max int]  
    [ niter_max_diffusion_implicit int]  
    [ precision_impr int]  
    [ periode_sauvegarde_securite_en_heures float]  
    [ no_check_disk_space ]  
    [ disable_progress ]  
    [ disable_dt_ev ]  
    [ gnuplot_header int]  
}  
where
```

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every `dt_sauv`, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the `.sauv` files, you must specify 0. Note that `dt_sauv` is in terms of physical time (not cpu time).

- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of nb_sauv_max timesteps in the file), the next checkpoints will overwrite the first ones
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: $dt_{start} dt_{min}$: the first iteration is based on dt_{min} .
 $dt_{start} dt_{calc}$: the time step at first iteration is calculated in agreement with CFL condition.
 $dt_{start} dt_{fixe}$ value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_{calc} .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.13 Schema_adams_bashforth_order_2

Description: not_set

See also: schema_temps_base (31)

Usage:

schema_adams_bashforth_order_2 *str*

Read *str* {

```
[ tinit float]  
[ tmax float]  
[ tcpumax float]  
[ dt_min float]  
[ dt_max str]  
[ dt_sauv float]  
[ nb_sauv_max int]  
[ dt_impr float]  
[ facsec str]  
[ seuil_statio float]  
[ residuals residuals]  
[ diffusion_implicite int]  
[ seuil_diffusion_implicite float]  
[ impr_diffusion_implicite int]  
[ impr_extremums int]  
[ no_error_if_not_converged_diffusion_implicite int]  
[ no_conv_subiteration_diffusion_implicite int]  
[ dt_start dt_start]  
[ nb_pas_dt_max int]  
[ niter_max_diffusion_implicite int]  
[ precision_impr int]  
[ periode_sauvegarde_securite_en_heures float]  
[ no_check_disk_space ]  
[ disable_progress ]  
[ disable_dt_ev ]  
[ gnuplot_header int]
```

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that **dt_sauv** is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of **nb_sauv_max** timesteps in the file), the next checkpoints will overwrite the first ones

- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min .
 dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
 dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.14 Schema_adams_bashforth_order_3

Description: not_set

See also: schema_temps_base (31)

Usage:

schema_adams_bashforth_order_3 *str*

Read *str* {

```
[ tinit float]  
[ tmax float]  
[ tcpumax float]  
[ dt_min float]  
[ dt_max str]  
[ dt_sauv float]  
[ nb_sauv_max int]  
[ dt_impr float]  
[ facsec str]  
[ seuil_statio float]  
[ residuals residuals]  
[ diffusion_implicite int]  
[ seuil_diffusion_implicite float]  
[ impr_diffusion_implicite int]  
[ impr_extremums int]  
[ no_error_if_not_converged_diffusion_implicite int]  
[ no_conv_subiteration_diffusion_implicite int]  
[ dt_start dt_start]  
[ nb_pas_dt_max int]  
[ niter_max_diffusion_implicite int]  
[ precision_impr int]  
[ periode_sauvegarde_securite_en_heures float]  
[ no_check_disk_space ]  
[ disable_progress ]  
[ disable_dt_ev ]  
[ gnuplot_header int]
```

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that **dt_sauv** is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of **nb_sauv_max** timesteps in the file), the next checkpoints will overwrite the first ones

- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min .
 dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
 dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.15 Schema_adams_moulton_order_2

Description: not_set

See also: schema_implicite_base (31.20)

Usage:

schema_adams_moulton_order_2 *str*

Read *str* {

```
[ facsec_max float]  
[ max_iter_implicite int]  
solveur solveur_implicite_base  
[ tinit float]  
[ tmax float]  
[ tcpumax float]  
[ dt_min float]  
[ dt_max str]  
[ dt_sauv float]  
[ nb_sauv_max int]  
[ dt_impr float]  
[ facsec str]  
[ seuil_statio float]  
[ residuals residuals]  
[ diffusion_implicite int]  
[ seuil_diffusion_implicite float]  
[ impr_diffusion_implicite int]  
[ impr_extremums int]  
[ no_error_if_not_converged_diffusion_implicite int]  
[ no_conv_subiteration_diffusion_implicite int]  
[ dt_start dt_start]  
[ nb_pas_dt_max int]  
[ niter_max_diffusion_implicite int]  
[ precision_impr int]  
[ periode_sauvegarde_securite_en_heures float]  
[ no_check_disk_space ]  
[ disable_progress ]  
[ disable_dt_ev ]  
[ gnuplot_header int]
```

}

where

- **facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.

Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.

Advice:

The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:

- Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
- Thermal hydraulic with forced convection and strong coupling between velocity and temperature

(Boussinesq value beta high), facsec between 90-100

-Thermohydraulic with natural convection, facsec around 300

-Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.

- **max_iter_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (32) 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, and ICE (for PB_multiphase). But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.

Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then PISO, and at least Simpler. Because the two first give a fastest convergence (several times) than PISO and the Simpler has not been validated. It seems also than Implicite and PISO schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to PISO or Implicite scheme.

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every *dt_sauv*, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that *dt_sauv* is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of *nb_sauv_max* timesteps in the file), the next checkpoints will overwrite the first ones
- **dt Impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.

Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.

- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened

meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_max$.

- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min .
 dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
 dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.16 Schema_adams_moulton_order_3

Description: not_set

See also: schema_implicite_base (31.20)

Usage:

schema_adams_moulton_order_3 *str*

Read *str* {

```
[ facsec_max float]
[ max_iter_implicite int]
solveur solveur_implicite_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ nb_sauv_max int]
[ dt_impr float]
[ facsec str]
```

```

[ seuil_statio float]
[ residuals residuals]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ impr_extremums int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures float]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
[ gnuplot_header int]
}

```

where

- **facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.

Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.

Advice:

The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:

- Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
- Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
- Thermohydraulic with natural convection, facsec around 300
- Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.

- **max_iter_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (32) 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, and ICE (for PB_multiphase). But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.

Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).

- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that **dt_sauv** is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of **nb_sauv_max** timesteps in the file), the next checkpoints will overwrite the first ones
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the **facsec** to 0.5.
Warning: Some schemes needs a **facsec** lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large **facsec** value. Start with a **facsec** value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: **dt_start dt_min** : the first iteration is based on **dt_min**.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on **dt_calc**.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).

- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.17 Schema_backward_differentiation_order_2

Description: not_set

See also: schema_implicite_base (31.20)

Usage:

schema_backward_differentiation_order_2 *str*

```
Read str {
    [ facsec_max float]
    [ max_iter_implicite int]
    solveur solveur_implicite_base
    [ tinit float]
    [ tmax float]
    [ tcpumax float]
    [ dt_min float]
    [ dt_max str]
    [ dt_sauv float]
    [ nb_sauv_max int]
    [ dt_impr float]
    [ facsec str]
    [ seuil_statio float]
    [ residuals residuals]
    [ diffusion_implicite int]
    [ seuil_diffusion_implicite float]
    [ impr_diffusion_implicite int]
    [ impr_extremums int]
    [ no_error_if_not_converged_diffusion_implicite int]
    [ no_conv_subiteration_diffusion_implicite int]
    [ dt_start dt_start]
    [ nb_pas_dt_max int]
    [ niter_max_diffusion_implicite int]
    [ precision_impr int]
    [ periode_sauvegarde_securite_en_heures float]
    [ no_check_disk_space ]
    [ disable_progress ]
    [ disable_dt_ev ]
    [ gnuplot_header int]
}
```

where

- **facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.

Warning: Some implicit schemes do not permit high `facsec_max`, example `Schema_Adams_Moulton_order_3` needs `facsec=facsec_max=1`.

Advice:

The calculation may start with a `facsec` specified by the user and increased by the algorithm up to the `facsec_max` limit. But the user can also choose to specify a constant `facsec` (`facsec_max` will be set to `facsec` value then). Faster convergence has been seen and depends on the kind of calculation:

- Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value `beta` low), `facsec` between 20-30
- Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value `beta` high), `facsec` between 90-100
- Thermohydraulic with natural convection, `facsec` around 300
- Conduction only, `facsec` can be set to a very high value ($1e8$) as if the scheme was unconditionally stable

These values can also be used as rule of thumb for initial `facsec` with a `facsec_max` limit higher.

- **max_iter_implicite** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (32) 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, and ICE (for PB_multiphase). 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 ($1e-16$ s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time ($1e30$ s by default).
- **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. To disable the writing of the `.sauv` files, you must specify 0. Note that `dt_sauv` is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of `nb_sauv_max` timesteps in the file), the next checkpoints will overwrite the first ones
- **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.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the `facsec` to 0.5.

Warning: Some schemes needs a `facsec` lower than 1 (0.5 is a good start), for example `Schema_Adams_Bashforth_order_3`.

- **seuil_statio** *float* for inheritance: Value of the convergence threshold ($1e-12$ by default). Problems

using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.

- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large *facsec* value. Start with a *facsec* value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value ($1e-6$) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: *dt_start dt_min* : the first iteration is based on *dt_min*.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on *dt_calc*.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps ($1e9$ by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.18 Schema_backward_differentiation_order_3

Description: not_set

See also: *schema_implicit_base* (31.20)

Usage:

schema_backward_differentiation_order_3 *str*

Read *str* {

[*facsec_max float*
[*max_iter_implicit int*
solveur solveur_implicit_base

```

[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ nb_sauv_max int]
[ dt_impr float]
[ facsec str]
[ seuil_statio float]
[ residuals residuals]
[ diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int]
[ impr_extremums int]
[ no_error_if_not_converged_diffusion_implicit int]
[ no_conv_subiteration_diffusion_implicit int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures float]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
[ gnuplot_header int]
}

```

where

- **facsec_max** *float*: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by facsec keyword is changed during the calculation with the implicit scheme but it couldn't be higher than facsec_max value.
Warning: Some implicit schemes do not permit high facsec_max, example Schema_Adams_Moulton_order_3 needs facsec=facsec_max=1.
Advice:
The calculation may start with a facsec specified by the user and increased by the algorithm up to the facsec_max limit. But the user can also choose to specify a constant facsec (facsec_max will be set to facsec value then). Faster convergence has been seen and depends on the kind of calculation:
-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value beta low), facsec between 20-30
-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value beta high), facsec between 90-100
-Thermohydraulic with natural convection, facsec around 300
-Conduction only, facsec can be set to a very high value (1e8) as if the scheme was unconditionally stable
These values can also be used as rule of thumb for initial facsec with a facsec_max limit higher.
- **max_iter_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (32) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solveur* is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, PISO (Pressure Implicit with Split Operator), and Implicit (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps, and ICE (for

PB_multiphase). But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.

Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simplr. Because the two first give a fastest convergence (several times) than Piso and the Simplr has not been validated. It seems also than Implicite and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of nb_sauv_max timesteps in the file), the next checkpoints will overwrite the first ones
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.

dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).

By default, the first iteration is based on dt_calc.

- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.19 Scheme_euler_implicit

Synonymous: **schema_euler_implicit**

Description: This is the Euler implicit scheme.

See also: **schema_implicit_base** ([31.20](#))

Usage:

scheme_euler_implicit *str*

Read *str* {

```
[ facsec_max float]
[ facsec_expert facsec_expert]
[ facsec_func str]
[ resolution_monolithique bloc_lecture]
[ max_iter_implicit int]
solveur solveur_implicit_base
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ nb_sauv_max int]
[ dt_impr float]
[ facsec str]
[ seuil_statio float]
[ residuals residuals]
[ diffusion_implicit int]
[ seuil_diffusion_implicit float]
[ impr_diffusion_implicit int]
[ impr_extremums int]
[ no_error_if_not_converged_diffusion_implicit int]
[ no_conv_subiteration_diffusion_implicit int]
```

```

[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicit int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures float]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
[ gnuplot_header int]
}
where

```

- **facsec_max** *float*: For old syntax, see the complete parameters of facsec for details
- **facsec_expert** *facsec_expert* (3.56): Advanced facsec specification
- **facsec_func** *str*: Advanced facsec specification as a function
- **resolution_monolithique** *bloc_lecture* (3.60): Activate monolithic resolution for coupled problems. Solves together the equations corresponding to the application domains in the given order. All application domains of the coupled equations must be given to determine the order of resolution. If the monolithic solving is not wanted for a specific application domain, an underscore can be added as prefix. For example, `resolution_monolithique { dom1 { dom2 dom3 } _dom4 }` will solve in a single matrix the equations having dom1 as application domain, then the equations having dom2 or dom3 as application domain in a single matrix, then the equations having dom4 as application domain in a sequential way (not in a single matrix).
- **max_iter_implicit** *int* for inheritance: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicit_base* (32) for inheritance: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. `solveur` is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicit (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps, and ICE (for PB_multiphase). But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.
Advice: Since the 1.6.0 version, we recommend to use first the Implicit or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicit and Piso schemes give better results than the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicit scheme.
- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every `dt_sauv`, fields are saved in the `.sauv` file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the `.sauv` files, you must specify 0. Note that `dt_sauv` is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of `nb_sauv_max` timesteps in the file), the next checkpoints will overwrite the first ones

- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min .
 dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
 dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc .
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.
- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.20 Schema_implicite_base

Description: Basic class for implicite time scheme.

See also: [schema_temps_base \(31\)](#) [schema_backward_differentiation_order_3 \(31.18\)](#) [schema_backward_differentiation_order_2 \(31.17\)](#) [scheme_euler_implicit \(31.19\)](#) [schema_adams_moulton_order_3 \(31.16\)](#) [schema_adams_moulton_order_2 \(31.15\)](#)

Usage:

schema_implicite_base *str*

Read *str* {

```
[ max_iter_implicite int]  
solveur solveur_implicite_base  
[ tinit float]  
[ tmax float]  
[ tcpumax float]  
[ dt_min float]  
[ dt_max str]  
[ dt_sauv float]  
[ nb_sauv_max int]  
[ dt_impr float]  
[ facsec str]  
[ seuil_statio float]  
[ residuals residuals]  
[ diffusion_implicite int]  
[ seuil_diffusion_implicite float]  
[ impr_diffusion_implicite int]  
[ impr_extremums int]  
[ no_error_if_not_converged_diffusion_implicite int]  
[ no_conv_subiteration_diffusion_implicite int]  
[ dt_start dt_start]  
[ nb_pas_dt_max int]  
[ niter_max_diffusion_implicite int]  
[ precision_impr int]  
[ periode_sauvegarde_securite_en_heures float]  
[ no_check_disk_space ]  
[ disable_progress ]  
[ disable_dt_ev ]  
[ gnuplot_header int]
```

}

where

- **max_iter_implicite** *int*: Maximum number of iterations allowed for the solver (by default 200).
- **solveur** *solveur_implicite_base* (32): This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme. *solver* is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are Simple (SIMPLE type algorithm), Simpler (SIMPLER type algorithm) for incompressible systems, Piso (Pressure Implicit with Split Operator), and Implicite (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps, and ICE (for PB_multiphase). But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.

Advice: Since the 1.6.0 version, we recommend to use first the Implicite or Simple, then Piso, and at least Simpler. Because the two first give a fastest convergence (several times) than Piso and the Simpler has not been validated. It seems also than Implicite and Piso schemes give better results than

the Simple scheme when the flow is not fully stationary. Thus, if the solution obtained with Simple is not stationary, it is recommended to switch to Piso or Implicite scheme.

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every dt_sauv, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that dt_sauv is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of nb_sauv_max timesteps in the file), the next checkpoints will overwrite the first ones
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicite** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large facsec value. Start with a facsec value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicite** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicite** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicite** *int* for inheritance
- **no_conv_subiteration_diffusion_implicite** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: dt_start dt_min : the first iteration is based on dt_min.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on dt_calc.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicite** *int* for inheritance: This keyword changes the default value (num-

ber of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.

- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

31.21 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 (31)

Usage:

schema_predictor_corrector *str*

Read *str* {

```
[ tinit float]
[ tmax float]
[ tcpumax float]
[ dt_min float]
[ dt_max str]
[ dt_sauv float]
[ nb_sauv_max int]
[ dt_impr float]
[ facsec str]
[ seuil_statio float]
[ residuals residuals]
[ diffusion_implicite int]
[ seuil_diffusion_implicite float]
[ impr_diffusion_implicite int]
[ impr_extremums int]
[ no_error_if_not_converged_diffusion_implicite int]
[ no_conv_subiteration_diffusion_implicite int]
[ dt_start dt_start]
[ nb_pas_dt_max int]
[ niter_max_diffusion_implicite int]
[ precision_impr int]
[ periode_sauvegarde_securite_en_heures float]
[ no_check_disk_space ]
[ disable_progress ]
[ disable_dt_ev ]
[ gnuplot_header int]
```

}

where

- **tinit** *float* for inheritance: Value of initial calculation time (0 by default).
- **tmax** *float* for inheritance: Time during which the calculation will be stopped (1e30s by default).
- **tcpumax** *float* for inheritance: CPU time limit (must be specified in hours) for which the calculation is stopped (1e30s by default).
- **dt_min** *float* for inheritance: Minimum calculation time step (1e-16s by default).
- **dt_max** *str* for inheritance: Maximum calculation time step as function of time (1e30s by default).
- **dt_sauv** *float* for inheritance: Save time step value (1e30s by default). Every **dt_sauv**, fields are saved in the .sauv file. The file contains all the information saved over time. If this instruction is not entered, results are saved only upon calculation completion. To disable the writing of the .sauv files, you must specify 0. Note that **dt_sauv** is in terms of physical time (not cpu time).
- **nb_sauv_max** *int* for inheritance: Maximum number of timesteps that will be stored in backup file (10 by default). This value is only useful when doing a complete backup of the calculation with parallel PDI (as it needs to allocate the proper amount of dataspace in advance). If this number is reached (ie we already stored the data of **nb_sauv_max** timesteps in the file), the next checkpoints will overwrite the first ones
- **dt_impr** *float* for inheritance: Scheme parameter printing time step in time (1e30s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the .out file.
- **facsec** *str* for inheritance: Value assigned to the safety factor for the time step (1. by default). It can also be a function of time. The time step calculated is multiplied by the safety factor. The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the **facsec** to 0.5.
Warning: Some schemes needs a **facsec** lower than 1 (0.5 is a good start), for example Schema_Adams_Bashforth_order_3.
- **seuil_statio** *float* for inheritance: Value of the convergence threshold (1e-12 by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.
- **residuals** *residuals* (3.108) for inheritance: To specify how the residuals will be computed (default max norm, possible to choose L2-norm instead).
- **diffusion_implicit** *int* for inheritance: Keyword to make the diffusive term in the Navier-Stokes equations implicit (in this case, it should be set to 1). The stability time step is then only based on the convection time step ($dt=facsec*dt_{convection}$). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). Caution: It is however recommended that the user avoids exceeding the convection time step by selecting a too large **facsec** value. Start with a **facsec** value of 1 and then increase it gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial velocity, in the first time step, the convection time is infinite and therefore $dt=facsec*dt_{max}$.
- **seuil_diffusion_implicit** *float* for inheritance: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for implicit diffusion.
- **impr_diffusion_implicit** *int* for inheritance: Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.
- **impr_extremums** *int* for inheritance: Print unknowns extremas
- **no_error_if_not_converged_diffusion_implicit** *int* for inheritance
- **no_conv_subiteration_diffusion_implicit** *int* for inheritance
- **dt_start** *dt_start* (11.8) for inheritance: **dt_start dt_min** : the first iteration is based on **dt_min**.
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when resuming calculation with Crank Nicholson temporal scheme to ensure continuity).
By default, the first iteration is based on **dt_calc**.
- **nb_pas_dt_max** *int* for inheritance: Maximum number of calculation time steps (1e9 by default).
- **niter_max_diffusion_implicit** *int* for inheritance: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for implicit diffusion.

- **precision_impr** *int* for inheritance: Optional keyword to define the digit number for flux values printed into .out files (by default 3).
- **periode_sauvegarde_securite_en_heures** *float* for inheritance: To change the default period (23 hours) between the save of the fields in .sauv file.
- **no_check_disk_space** for inheritance: To disable the check of the available amount of disk space during the calculation.
- **disable_progress** for inheritance: To disable the writing of the .progress file.
- **disable_dt_ev** for inheritance: To disable the writing of the .dt_ev file.
- **gnuplot_header** *int* for inheritance: Optional keyword to modify the header of the .out files. Allows to use the column title instead of columns number.

32 solveur_implicite_base

Description: Class for solver in the situation where the time scheme is the implicit scheme. Solver allows equation diffusion and convection operators to be set as implicit terms.

See also: objet_u (40) simplifier (32.6) solveur_lineaire_std (32.7)

Usage:

32.1 Ice

Description: Implicit Continuous-fluid Eulerian solver which is useful for a multiphase problem. Robust pressure reduction resolution.

See also: sets (32.4)

Usage:

ice *str*

Read *str* {

```
[ pression_degeneree int]
[ pressure_reduction|reduction_preSSION int]
[ criteres_convergence bloc_criteres_convergence]
[ iter_min int]
[ iter_max int]
[ seuil_convergence_implicite float]
[ nb_corrections_max int]
[ facsec_diffusion_for_sets 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

- **pression_degeneree** *int*: Set to 1 if the pressure field is degenerate (ex. : incompressible fluid with no imposed-pressure BCs). Default: autodetected

- **pressure_reduction|reduction_pression** *int*: Set to 1 if the user wants a resolution with a pressure reduction. Otherwise, the value is to be set to 0 so that the complete matrix is considered. The default value of this value is 1.
- **criteres_convergence** *bloc_criteres_convergence* (3.60.1) for inheritance: Set the convergence thresholds for each unknown (i.e: alpha, temperature, velocity and pressure). The default values are respectively 0.01, 0.1, 0.01 and 100
- **iter_min** *int* for inheritance: Number of minimum iterations (default value 1)
- **iter_max** *int* for inheritance: Number of maximum iterations (default value 10)
- **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).
- **facsec_diffusion_for_sets** *float* for inheritance: facsec to impose on the diffusion time step in sets while the total time step stays smaller than the convection time step.
- **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* (11.16) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

32.2 Implicite

Description: similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.

See also: piso (32.3)

Usage:

implicite *str*

Read *str* {

```
[ seuil_convergence_implicite float]
[ nb_corrections_max int]
[ seuil_convergence_solveur float]
[ seuil_generation_solveur float]
[ seuil_verification_solveur float]
[ seuil_test_preliminaire_solveur float]
[ solveur solveur_sys_base]
[ no_qdm ]
```

```

[ nb_it_max int]
[ controle_residu ]
}

```

where

- **seuil_convergence_implicite** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections than nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier-Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error $\|Ax-B\|$ is lesser than vrel after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than vrel.
- **solveur** *solveur_sys_base* (11.16) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

32.3 Piso

Description: Piso (Pressure Implicit with Split Operator) - method to solve N_S.

See also: simpler (32.6) implicite (32.2) simple (32.5)

Usage:

piso *str*

Read *str* {

```

[ seuil_convergence_implicite float]
[ nb_corrections_max int]
[ seuil_convergence_solveur float]
[ seuil_generation_solveur float]
[ seuil_verification_solveur float]
[ seuil_test_preliminaire_solveur float]
[ solveur solveur_sys_base]
[ no_qdm ]
[ nb_it_max int]
[ controle_residu ]

```

}

where

- **seuil_convergence_implicite** *float*: Convergence criteria.

- **nb_corrections_max** *int*: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections than nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error $\|Ax-B\|$ is lesser than vrel after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than vrel.
- **solveur** *solveur_sys_base* (11.16) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

32.4 Sets

Description: Stability-Enhancing Two-Step solver which is useful for a multiphase problem. Ref : J. H. MAHAFFY, A stability-enhancing two-step method for fluid flow calculations, Journal of Computational Physics, 46, 3, 329 (1982).

See also: [simpler \(32.6\)](#) [ice \(32.1\)](#)

Usage:

sets *str*

Read *str* {

```
[ criteres_convergence bloc_criteres_convergence]
[ iter_min int]
[ iter_max int]
[ seuil_convergence_implicite float]
[ nb_corrections_max int]
[ facsec_diffusion_for_sets 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

- **criteres_convergence** *bloc_criteres_convergence* (3.60.1): Set the convergence thresholds for each unknown (i.e: alpha, temperature, velocity and pressure). The default values are respectively 0.01, 0.1, 0.01 and 100
- **iter_min** *int*: Number of minimum iterations (default value 1)
- **iter_max** *int*: Number of maximum iterations (default value 10)
- **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).
- **facsec_diffusion_for_sets** *float*: facsec to impose on the diffusion time step in sets while the total time step stays smaller than the convection time step.
- **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* (11.16) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the `residu` suddenly increases.

32.5 Simple

Description: SIMPLE type algorithm

See also: `piso` (32.3) `solveur_u_p` (32.8)

Usage:

simple *str*

Read *str* {

```
[ 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* (11.16) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

32.6 Simplifier

Description: Simplifier method for incompressible systems.

See also: solveur_implicite_base (32) sets (32.4) piso (32.3)

Usage:

simpler *str*

Read *str* {

```
    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 Simplifier 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* (11.16): Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** : Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int*: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** : Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the *residu* suddenly increases.

32.7 Solveur_lineaire_std

Description: *not_set*

See also: *solveur_implicite_base* (32)

Usage:

solveur_lineaire_std *str*

Read *str* {

 [**solveur** *solveur_sys_base*]

}

where

- **solveur** *solveur_sys_base* (11.16)

32.8 Solveur_u_p

Description: similar to *simple*.

See also: *simple* (32.5)

Usage:

solveur_u_p *str*

Read *str* {

 [**relax_pression** *float*]

 [**seuil_convergence_implicite** *float*]

 [**nb_corrections_max** *int*]

 [**seuil_convergence_solveur** *float*]

 [**seuil_generation_solveur** *float*]

 [**seuil_verification_solveur** *float*]

 [**seuil_test_preliminaire_solveur** *float*]

 [**solveur** *solveur_sys_base*]

```

[ no_qdm ]
[ nb_it_max int]
[ controle_residu ]
}
where

```

- **relax_pression** *float* for inheritance: Value between 0 and 1 (by default 1), this keyword is used only by the SIMPLE algorithm for relaxing the increment of pressure.
- **seuil_convergence_implicit** *float* for inheritance: Convergence criteria.
- **nb_corrections_max** *int* for inheritance: Maximum number of corrections performed by the PISO algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections than nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections_max is set to 21).
- **seuil_convergence_solveur** *float* for inheritance: value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier-Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).
- **seuil_generation_solveur** *float* for inheritance: Option to create a GMRES solver and use vrel as the convergence threshold (implicit linear system $Ax=B$ will be solved if residual error $\|Ax-B\|$ is lesser than vrel).
- **seuil_verification_solveur** *float* for inheritance: Option to check if residual error $\|Ax-B\|$ is lesser than vrel after the implicit linear system $Ax=B$ has been solved.
- **seuil_test_preliminaire_solveur** *float* for inheritance: Option to decide if the implicit linear system $Ax=B$ should be solved by checking if the residual error $\|Ax-B\|$ is bigger than vrel.
- **solveur** *solveur_sys_base* (11.16) for inheritance: Method (different from the default one, Gmres with diagonal preconditioning) to solve the linear system.
- **no_qdm** for inheritance: Keyword to not solve qdm equation (and turbulence models of these equation).
- **nb_it_max** *int* for inheritance: Keyword to set the maximum iterations number for the Gmres.
- **controle_residu** for inheritance: Keyword of Boolean type (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

33 solveur_petsc_deriv

Description: Additional information is available in the PETSC documentation: <https://petsc.org/release/manual/>

See also: objet_u (40) lu (33.14) Cholesky_superlu (33.4) Cholesky_pastix (33.3) Cholesky_umfpack (33.5) Cholesky_out_of_core (33.2) cholesky (33.8) cholesky_mumps_blr (33.9) cli (33.10) cli_quiet (33.11) IBICGSTAB (33.6) BICGSTAB (33.1) gmres (33.13) gcp (33.12) PIPECG (33.7)

Usage:

solveur_petsc_deriv *str*

Read *str* {

```

[ seuil float]
[ quiet ]
[ impr ]
[ rtol float]
[ atol float]
[ save_matrix_mtx_format ]

```

```

}
where

```

- **seuil** *float*: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** : is a keyword which is used to not displaying any outputs of the solver.
- **impr** : used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float*
- **atol** *float*
- **save_matrix_mtx_format**

33.1 Bicgstab

Description: Stabilized Bi-Conjugate Gradient

See also: [solveur_petsc_deriv \(33\)](#)

Usage:

BICGSTAB *str*

Read *str* {

```
[ precond preconditionneur_petsc_deriv]
[ seuil float]
[ quiet ]
[ impr ]
[ rtol float]
[ atol float]
[ save_matrix_mtx_format ]
```

}

where

- **precond** *preconditionneur_petsc_deriv* ([30](#))
- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float* for inheritance
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

33.2 Cholesky_out_of_core

Description: Same as the previous one but with a written LU decomposition of disk (save RAM memory but add an extra CPU cost during $Ax=B$ solve).

See also: [solveur_petsc_deriv \(33\)](#)

Usage:

Cholesky_out_of_core *str*

Read *str* {

```
[ seuil float]
[ quiet ]
[ impr ]
```

```

    [ rtol float]
    [ atol float]
    [ save_matrix_mtx_format ]
}
where

```

- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float* for inheritance
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

33.3 Cholesky_pastix

Description: Parallelized Cholesky from PASTIX library.

See also: `solveur_petsc_deriv` ([33](#))

Usage:

Cholesky_pastix *str*

```

Read str {
    [ seuil float]
    [ quiet ]
    [ impr ]
    [ rtol float]
    [ atol float]
    [ save_matrix_mtx_format ]
}
where

```

- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float* for inheritance
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

33.4 Cholesky_superlu

Description: Parallelized Cholesky from SUPERLU_DIST library (less CPU and RAM, efficient than the previous one)

See also: `solveur_petsc_deriv` ([33](#))

Usage:

Cholesky_superlu *str*

```

Read str {

```

```

    [ seuil float]
    [ quiet ]
    [ impr ]
    [ rtol float]
    [ atol float]
    [ save_matrix_mtx_format ]
}
where

```

- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float* for inheritance
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

33.5 Cholesky_umfpack

Description: Sequential Cholesky from UMFPACK library (seems fast).

See also: [solveur_petsc_deriv \(33\)](#)

Usage:

Cholesky_umfpack *str*

```

Read str {
    [ seuil float]
    [ quiet ]
    [ impr ]
    [ rtol float]
    [ atol float]
    [ save_matrix_mtx_format ]
}
where

```

- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float* for inheritance
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

33.6 Ibicgstab

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

See also: [solveur_petsc_deriv \(33\)](#)

Usage:

IBICGSTAB *str*

Read *str* {

[**precond** *preconditionneur_petsc_deriv*]
[**seuil** *float*]
[**quiet**]
[**impr**]
[**rtol** *float*]
[**atol** *float*]
[**save_matrix_mtx_format**]

}

where

- **precond** *preconditionneur_petsc_deriv* (30)
- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float* for inheritance
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

33.7 Pipecg

Description: 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)... no example in TRUST

See also: *solveur_petsc_deriv* (33)

Usage:

PIPECG *str*

Read *str* {

[**seuil** *float*]
[**quiet**]
[**impr**]
[**rtol** *float*]
[**atol** *float*]
[**save_matrix_mtx_format**]

}

where

- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float* for inheritance
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

33.8 Cholesky

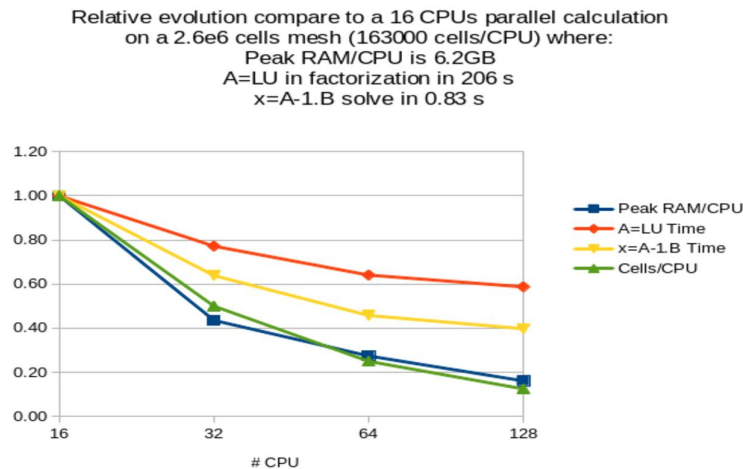
Description: Parallelized version of Cholesky from MUMPS library. This solver accepts an option to select a different ordering than the automatic selected one by MUMPS (and printed by using the `impr` option). The possible choices are Metis, Scotch, PT-Scotch or Parmetis. The two last options can only be used during a parallel calculation, whereas the two first are available for sequential or parallel calculations. It seems that the CPU cost of $A=LU$ factorization but also of the backward/forward elimination steps may sometimes be reduced by selecting a different ordering (Scotch seems often the best for b/f elimination) than the default one.

Notice that this solver requires a huge amount of memory compared to iterative methods. To know how much 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 zeroth 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) :



See also: `solveur_petsc_deriv` (33)

Usage:

cholesky *str*

Read *str* {

```
[ save_matrix|save_matrice ]
[ save_matrix_petsc_format ]
[ reduce_ram ]
[ cli_quiet solveur_petsc_option_cli]
[ cli solveur_petsc_option_cli]
[ seuil float]
[ quiet ]
[ impr ]
[ rtol float]
[ atol float]
[ save_matrix_mtx_format ]
```



```
}
```

where

- **save_matrix|save_matrice**
- **save_matrix_petsc_format**
- **reduce_ram**
- **cliQuiet** *solveur_petsc_option_cli* (3.60.2)
- **cli** *solveur_petsc_option_cli* (3.60.2)
- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float* for inheritance
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

33.9 Cholesky_mumps_blr

Description: BLR for (Block Low-Rank)

See also: [solveur_petsc_deriv](#) (33)

Usage:

cholesky_mumps_blr *str*

Read *str* {

```
[ reduce_ram ]
[ dropping_parameter float]
[ cli solveur_petsc_option_cli]
[ seuil float]
[ quiet ]
[ impr ]
[ rtol float]
[ atol float]
[ save_matrix_mtx_format ]
```

```
}
```

where

- **reduce_ram**
- **dropping_parameter** *float*
- **cli** *solveur_petsc_option_cli* (3.60.2)
- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float* for inheritance
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

33.10 Cli

Description: 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`

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

HYPRE preconditioner options:

`-pc_hypre_type` pilut (choose one of) pilut parasails boomeramg

HYPRE ParaSails Options

`-pc_hypre_parasails_nlevels` 1: Number of number of levels (None)

`-pc_hypre_parasails_thresh` 0.1: Threshold (None)

`-pc_hypre_parasails_filter` 0.1: filter (None)

`-pc_hypre_parasails_loadbal` 0: Load balance (None)

`-pc_hypre_parasails_logging`: FALSE Print info to screen (None)

`-pc_hypre_parasails_reuse`: FALSE Reuse nonzero pattern in preconditioner (None)

`-pc_hypre_parasails_sym` nonsymmetric (choose one of) nonsymmetric SPD nonsymmetric,SPD

Krylov Method (KSP) Options

`-ksp_type` Krylov method:(one of) cg cgne stcg gltr richardson chebychev gmres tcqmr bcgs bcgsl cgs tfqmr cr lsqr preonly qcg bicg fgmres minres symmlq lgmres lcd (KSPSetType)

`-ksp_max_it` 10000: Maximum number of iterations (KSPSetTolerances)

`-ksp_rtol` 0: Relative decrease in residual norm (KSPSetTolerances)

`-ksp_atol` 1e-12: Absolute value of residual norm (KSPSetTolerances)

`-ksp_divtol` 10000: Residual norm increase cause divergence (KSPSetTolerances)

`-ksp_converged_use_initial_residual_norm`: Use initial residual residual norm for computing relative convergence

`-ksp_monitor_singular_value` stdout: Monitor singular values (KSPMonitorSet)

`-ksp_monitor_short` stdout: Monitor preconditioned residual norm with fewer digits (KSPMonitorSet)

`-ksp_monitor_draw`: Monitor graphically preconditioned residual norm (KSPMonitorSet)

`-ksp_monitor_draw_true_residual`: Monitor graphically true residual norm (KSPMonitorSet)

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

Solveur_pression Petsc CLI `{-ksp_type richardson -pc_type hypre -pc_hypre_type boomeramg -ksp_atol 1.e-7 }`

See also: `solveur_petsc_deriv` (33)

Usage:

cli cli_bloc

where

- **cli_bloc** *bloc_lecture* (3.60): bloc

33.11 Cli_quiet

Description: solver

See also: `solveur_petsc_deriv` (33)

Usage:

cli_quiet cli_quiet_bloc

where

- **cli_quiet_bloc** *bloc_lecture* (3.60): bloc

33.12 Gcp

Description: Preconditioned Conjugate Gradient

See also: *solveur_petsc_deriv* (33)

Usage:

gcp *str*

Read *str* {

```
[ precond preconditionneur_petsc_deriv]
[ precond_nul ]
[ rtol float]
[ reuse_preconditioner_nb_it_max int]
[ cli solveur_petsc_option_cli]
[ reorder_matrix int]
[ read_matrix ]
[ save_matrix|save_matrice ]
[ petsc_decide int]
[ pcshell str]
[ aij ]
[ seuil float]
[ quiet ]
[ impr ]
[ atol float]
[ save_matrix_mtx_format ]
```

}

where

- **precond** *preconditionneur_petsc_deriv* (30): preconditioner
- **precond_nul** : No preconditioner used, equivalent to precondition null { }
- **rtol** *float*
- **reuse_preconditioner_nb_it_max** *int*
- **cli** *solveur_petsc_option_cli* (3.60.2)
- **reorder_matrix** *int*
- **read_matrix** : *save_matrix|read_matrix* are the keywords to save/read into a file the constant matrix A of the linear system $Ax=B$ solved (eg: matrix from the pressure linear system for an incompressible flow). It is useful when you want to minimize the MPI communications on massive parallel calculation. Indeed, in VEF discretization, the overlapping width (generally 2, specified with the *largeur_joint* option in the partition keyword partition) can be reduced to 1, once the matrix has been properly assembled and saved. The cost of the MPI communications in TRUST itself (not in PETSc) will be reduced with length messages divided by 2. So the strategy is:
 - I) Partition your VEF mesh with a *largeur_joint* value of 2
 - II) Run your parallel calculation on 0 time step, to build and save the matrix with the *save_matrix* option. A file named *Matrix_NBROWS_rows_NCPUS_cpus.petsc* will be saved to the disk (where NBROWS is the number of rows of the matrix and NCPUS the number of CPUs used).
 - III) Partition your VEF mesh with a *largeur_joint* value of 1
 - IV) Run your parallel calculation completely now and substitute the *save_matrix* option by the *read_matrix* option. Some interesting gains have been noticed when the cost of linear system solve with PETSc is small compared to all the other operations.
- **save_matrix|save_matrice** : see *read_matrix*

- **petsc_decide** *int*
- **pcshell** *str*
- **aij**
- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

33.13 Gmres

Description: Generalized Minimal Residual

See also: [solveur_petsc_deriv \(33\)](#)

Usage:

gmres *str*

Read *str* {

```
[ precond preconditionneur_petsc_deriv]
[ reuse_preconditioner_nb_it_max int]
[ save_matrix_petsc_format ]
[ nb_it_max int]
[ seuil float]
[ quiet ]
[ impr ]
[ rtol float]
[ atol float]
[ save_matrix_mtx_format ]
```

}

where

- **precond** *preconditionneur_petsc_deriv* [\(30\)](#)
- **reuse_preconditioner_nb_it_max** *int*
- **save_matrix_petsc_format**
- **nb_it_max** *int*: 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.
- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float* for inheritance
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

33.14 Lu

Description: Several solvers through PETSc API are available.

TIPS:

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

-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.

See also: `solveur_petsc_deriv` (33)

Usage:

lu *str*

Read *str* {

[**seuil** *float*]
[**quiet**]
[**impr**]
[**rtol** *float*]
[**atol** *float*]
[**save_matrix_mtx_format**]

}

where

- **seuil** *float* for inheritance: corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard $\|Ax-B\|$ is less than **seuil**.
- **quiet** for inheritance: is a keyword which is used to not displaying any outputs of the solver.
- **impr** for inheritance: used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).
- **rtol** *float* for inheritance
- **atol** *float* for inheritance
- **save_matrix_mtx_format** for inheritance

34 source_base

Description: Basic class of source terms introduced in the equation.

See also: `objet_u` (40) `darcy` (34.13) `puissance_thermique` (34.25) `forchheimer` (34.16) `dirac` (34.14) `source_constituant` (34.27) `vitesse_relative_base` (34.38) `flux_interfacial` (34.15) `frottement_interfacial` (34.17) `Portance_interfaciale` (34.6) `travail_pression` (34.36) `Dispersion_bulles` (34.5) `coriolis` (34.12) `perte_charge_singuliere` (34.24) `canal_perio` (34.11) `perte_charge_reguliere` (34.22) `source_qdm` (34.32) `acceleration` (34.8) `DP_Impose` (34.3) `boussinesq_temperature` (34.10) `boussinesq_concentration` (34.9) `terme_puissance_thermique_echange_impose` (34.35) `Correction_Tomiyama` (34.2) `Correction_Antal` (34.1) `radioactive_decay` (34.26) `source_qdm_lambdaup` (34.33) `source_th_tdivu` (34.34) `perte_charge_isotrope` (34.21) `perte_charge_directionnelle` (34.20) `perte_charge_anisotrope` (34.18) `perte_charge_circulaire` (34.19) `source_generique` (34.28) `Source_dep_inco_bases` (34.7)

Usage:

34.1 Correction_antal

Description: Antal correction source term for multiphase problem

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

Usage:

34.2 Correction_tomiyama

Description: Tomiyama correction source term for multiphase problem

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

Usage:

34.3 Dp_impose

Description: Source term to impose a pressure difference according to the formula : $DP = dp + dDP/dQ * (Q - Q0)$

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

Usage:

DP_Impose aco dp_type surface bloc_surface acof
where

- **aco** *str* into [' ']: Opening curly bracket.
- **dp_type** *type_perte_charge_deriv (34.4)*: mass flow rate (kg/s).
- **surface** *str* into ['surface']
- **bloc_surface** *bloc_lecture (3.60)*: Three syntaxes are possible for the surface definition block:
For VDF and VEF: { X|Y|Z = location subzone_name }
Only for VEF: { Surface surface_name }.
For polymac { Surface surface_name Orientation champ_uniforme }.
- **acof** *str* into [' ']: Closing curly bracket.

34.4 Type_perte_charge_deriv

Description: not_set

See also: [objet_lecture \(39\)](#) [dp \(34.4.1\)](#) [dp_regul \(34.4.2\)](#)

Usage:

34.4.1 Dp

Description: DP field should have 3 components defining dp, dDP/dQ, Q0

See also: [type_perte_charge_deriv \(34.4\)](#)

Usage:

dp dp_field

where

- **dp_field** *champ_base* (16.1): the parameters of the previous formula ($DP = dp + dDP/dQ * (Q - Q0)$): *uniform_field* 3 *dp* *dDP/dQ* *Q0* where *Q0* is a mass flow rate (kg/s).

34.4.2 Dp_regul

Description: Keyword used to regulate the DP value in order to match a target flow rate. Syntax : **dp_regul** { *DP0* *d* *deb* *d* *eps* *e* }

See also: *type_perte_charge_deriv* (34.4)

Usage:

dp_regul {

DP0 *float*

deb *str*

eps *str*

}

where

- **DP0** *float*: initial value of DP
- **deb** *str*: target flow rate in kg/s
- **eps** *str*: strength of the regulation (low values might be slow to find the target flow rate, high values might oscillate around the target value)

34.5 Dispersion_bulles

Description: Base class for source terms of bubble dispersion in momentum equation.

See also: *source_base* (34)

Usage:

Dispersion_bulles *str*

Read *str* {

 [**beta** *float*]

}

where

- **beta** *float*: Mutliplying factor for the output of the bubble dispersion source term.

34.6 Portance_interfaciale

Description: Base class for source term of lift force in momentum equation.

See also: *source_base* (34)

Usage:

Portance_interfaciale *str*

Read *str* {

```
[ beta float]
```

```
}
```

where

- **beta** *float*: Multiplying factor for the bubble lift force source term.

34.7 Source_dep_inco_bases

Description: Basic class of source terms depending of inknown.

See also: [source_base \(34\)](#) [source_pdf_base \(34.31\)](#)

Usage:

34.8 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 \(34\)](#)

Usage:

acceleration *str*

Read *str* {

```
[ vitesse champ_base]
```

```
[ acceleration champ_base]
```

```
[ omega champ_base]
```

```
[ domegadt champ_base]
```

```
[ centre_rotation champ_base]
```

```
[ option str into ['terme_complet', 'coriolis_seul', 'entrainement_seul']]
```

```
}
```

where

- **vitesse** *champ_base* (16.1): Keyword for the velocity of the referential R' into the R referential ($d\vec{OO'}/dt$ term [m.s-1]). The velocity is mandatory when you want to print the total cinetic energy into the non-mobile Galilean referential R (see `Ec_dans_repere_fixe` keyword).
- **acceleration** *champ_base* (16.1): Keyword for the acceleration of the referential R' into the R referential ($d^2\vec{OO'}/dt^2$ term [m.s-2]). *field_base* is a time dependant field (eg: `Champ_Fonc_t`).
- **omega** *champ_base* (16.1): Keyword for a rotation of the referential R' into the R referential [rad.s-1]. *field_base* is a 3D time dependant field specified for example by a `Champ_Fonc_t` keyword. The *time_field* field should have 3 components even in 2D (In 2D: 0 0 omega).
- **domegadt** *champ_base* (16.1): Keyword to define the time derivative of the previous rotation [rad.s-2]. Should be zero if the rotation is constant. The *time_field* field should have 3 components even in 2D (In 2D: 0 0 domegadt).
- **centre_rotation** *champ_base* (16.1): Keyword to specify the centre of rotation (expressed in R' coordinates) of R' into R (if the domain rotates with the R' referential, the centre of rotation is $O'=(0,0,0)$). 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.

34.9 Boussinesq_concentration

Description: Class to describe a source term that couples the movement quantity equation and constituent transport equation with the Boussinesq hypothesis.

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

Usage:

boussinesq_concentration *str*

Read *str* {

c0 *n x1 x2 ... xn*

}

where

- **c0** *n x1 x2 ... xn*: Reference concentration field type. The only field type currently available is Champ_Uniforme (Uniform field).

34.10 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 \(34\)](#)

Usage:

boussinesq_temperature *str*

Read *str* {

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 value in comparison with the mean value in the domain. It is set to 1 by default.

34.11 Canal_perio

Description: Momentum source term to maintain flow rate. The expression of the source term is:

$$S(t) = (2*(Q(0) - Q(t)) - (Q(0) - Q(t-dt)))/(coeff*dt*area)$$

Where:

coeff=damping coefficient

area=area of the periodic boundary

Q(t)=flow rate at time t

dt=time step

Three files will be created during calculation on a datafile named DataFile.data. The first file contains the flow rate evolution. The second file is useful for resuming a calculation with the flow rate of the previous stopped calculation, and the last one contains the pressure gradient evolution:

-DataFile_Channel_Flow_Rate_ProblemName_BoundaryName
 -DataFile_Channel_Flow_Rate_repr_ProblemName_BoundaryName
 -DataFile_Pressure_Gradient_ProblemName_BoundaryName

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

Usage:

canal_perio *str*

Read *str* {

 [**u_etoile** *float*]
 [**coeff** *float*]
 [**h** *float*]
bord *str*
 [**debit_impose** *float*]

}

where

- **u_etoile** *float*
- **coeff** *float*: Damping coefficient (optional, default value is 10).
- **h** *float*: Half height of the channel.
- **bord** *str*: The name of the (periodic) boundary normal to the flow direction.
- **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.

34.12 Coriolis

Description: Keyword for a Coriolis term in hydraulic equation. Warning: Only available in VDF.

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

Usage:

coriolis *str*

Read *str* {

omega *n x1 x2 ... xn*

}

where

- **omega** *n x1 x2 ... xn*: Value of omega.

34.13 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 \(34\)](#)

Usage:

darcy bloc

where

- **bloc** *bloc_lecture* (3.60): Description.

34.14 Dirac

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: *source_base* (34)

Usage:

dirac position ch
where

- **position** *n x1 x2 ... xn*
- **ch** *champ_base* (16.1): Thermal power field type. To impose a volume power on a domain sub-area, the *Champ_Uniforme_Morceaux* (*partly_uniform_field*) type must be used.
Warning : The volume thermal power is expressed in W.m-3.

34.15 Flux_interfacial

Description: Source term of mass transfer between phases connected by the saturation object defined in *saturation_xxxx*

See also: *source_base* (34)

Usage:

flux_interfacial

34.16 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* (34)

Usage:

forchheimer bloc
where

- **bloc** *bloc_lecture* (3.60): Description.

34.17 Frottement_interfacial

Description: Source term which corresponds to the phases friction at the interface

See also: *source_base* (34)

Usage:

frottement_interfacial str
Read str {
 [**a_res** float]

```

    [ dv_min float]
    [ exp_res int]
}

```

where

- **a_res** *float*: void fraction at which the gas velocity is forced to approach liquid velocity (default $\alpha_{\text{evanescence}} \times 100$)
- **dv_min** *float*: minimal relative velocity used to linearize interfacial friction at low velocities
- **exp_res** *int*: exponent that callibrates intensity of velocity convergence (default 2)

34.18 Perte_charge_anisotrope

Description: Anisotropic pressure loss.

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

Usage:

perte_charge_anisotrope *str*

Read *str* {

```

    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* ([16.9](#)): Hydraulic diameter value.
- **direction** *champ_don_base* ([16.9](#)): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

34.19 Perte_charge_circulaire

Description: New pressure loss.

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

Usage:

perte_charge_circulaire *str*

Read *str* {

```

    lambda str
    diam_hydr champ_don_base
    [ sous_zone str]
    lambda_ortho str
    diam_hydr_ortho champ_don_base
    direction champ_don_base

```

}
where

- **lambda** *str*: Function $f(\text{Re}_{\text{tot}}, \text{Re}_{\text{long}}, t, x, y, z)$ for loss coefficient in the longitudinal direction
- **diam_hydr** *champ_don_base* (16.9): Hydraulic diameter value.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.
- **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_ortho** *champ_don_base* (16.9): Transverse hydraulic diameter value.
- **direction** *champ_don_base* (16.9): Field which indicates the direction of the pressure loss.

34.20 Perte_charge_directionnelle

Description: Directional pressure loss (available in VEF and PolyMAC).

See also: [source_base](#) (34)

Usage:

perte_charge_directionnelle *str*

Read *str* {

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* (16.9): Hydraulic diameter value.
- **direction** *champ_don_base* (16.9): Field which indicates the direction of the pressure loss.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

34.21 Perte_charge_isotrope

Description: Isotropic pressure loss (available in VEF and PolyMAC).

See also: [source_base](#) (34)

Usage:

perte_charge_isotrope *str*

Read *str* {

lambda *str*
 diam_hydr *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* (16.9): Hydraulic diameter value.
- **sous_zone** *str*: Optional sub-area where pressure loss applies.

34.22 Perte_charge_reguliere

Description: Source term modelling the presence of a bundle of tubes in a flow.

See also: `source_base` (34)

Usage:

perte_charge_reguliere spec zone_name

where

- **spec** *spec_pdc_base* (34.23): 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.

34.23 Spec_pdc_base

Description: Class to read the source term modelling the presence of a bundle of tubes in a flow. $C_f = A \text{Re}^{-B}$.

See also: `objet_lecture` (39) *longitudinale* (34.23.1) *transversale* (34.23.2)

Usage:

spec_pdc_base

34.23.1 Longitudinale

Description: Class to define the pressure loss in the direction of the tube bundle.

See also: `spec_pdc_base` (34.23)

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.

34.23.2 Transversale

Description: Class to define the pressure loss in the direction perpendicular to the tube bundle.

See also: `spec_pdc_base` (34.23)

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.

- **chaîne_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.

34.24 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 (34)

Usage:

perte_charge_singuliere *str*

Read *str* {

dir *str* into [*'kx'*, *'ky'*, *'kz'*, *'K'*]
 [**coeff** *float*]
 [**regul** *bloc_lecture*]
 surface *bloc_lecture*

}

where

- **dir** *str* into [*'kx'*, *'ky'*, *'kz'*, *'K'*]: KX, KY or KZ designate directional pressure loss coefficients for respectively X, Y or Z direction. Or in the case where you chose a target flow rate with regul. Use K for isotropic pressure loss coefficient
- **coeff** *float*: Value (float) of friction coefficient (KX, KY, KZ).
- **regul** *bloc_lecture* (3.60): option to have adjustable K with flowrate target { K0 valeur_initiale_de_k deb debit_cible eps intervalle_variation_mutiplicatif }.
- **surface** *bloc_lecture* (3.60): Three syntaxes are possible for the surface definition block:
 For VDF and VEF: { X|Y|Z = location subzone_name }
 Only for VEF: { Surface surface_name }.
 For polymac { Surface surface_name Orientation champ_uniforme }

34.25 Puissance_thermique

Description: Class to define a source term corresponding to a volume power release in the energy equation.

See also: source_base (34)

Usage:

puissance_thermique **ch**

where

- **ch** *champ_base* (16.1): Thermal power field type. To impose a volume power on a domain sub-area, the Champ_Uniforme_Morceaux (partly_uniform_field) type must be used.
 Warning : The volume thermal power is expressed in W.m-3 in 3D (in W.m-2 in 2D). It is a power per volume unit (in a porous media, it is a power per fluid volume unit).

34.26 Radioactive_decay

Description: Radioactive decay source term of the form $-\lambda_i c_i$, where $0 \leq i \leq N$, N is the number of component of the constituent, c_i and λ_i are the concentration and the decay constant of the i -th component of the constituent.

See also: `source_base` (34)

Usage:

radioactive_decay **val**

where

- **val** n $x1$ $x2$... xn : n is the number of decay constants to read (int), and $val1$, $val2$... are the decay constants (double)

34.27 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` (34)

Usage:

source_constituant **ch**

where

- **ch** *champ_base* (16.1): Field type.

34.28 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` (34)

Usage:

source_generique **champ**

where

- **champ** *champ_generique_base* (9): the source field

34.29 Source_pdf

Description: Source term for Penalised Direct Forcing (PDF) method.

See also: `source_pdf_base` (34.31)

Usage:

source_pdf *str*

Read *str* {

aire *champ_base*

rotation *champ_base*


```

[ transpose_rotation ]
modele bloc_pdf_model
[ interpolation interpolation_ibm_base]
}
where

```

- **aire** *champ_base* (16.1) for inheritance: volumic field: a boolean for the cell (0 or 1) indicating if the obstacle is in the cell
- **rotation** *champ_base* (16.1) for inheritance: volumic field with 9 components representing the change of basis on cells (local to global). Used for rotating cases for example.
- **transpose_rotation** for inheritance: whether to transpose the basis change matrix.
- **modele** *bloc_pdf_model* (34.30) for inheritance: model used for the Penalized Direct Forcing
- **interpolation** *interpolation_ibm_base* (18) for inheritance: interpolation method

34.30 Bloc_pdf_model

Description: not_set

See also: objet_lecture (39)

Usage:

```

{
    eta float
    [ bilan_pdf int]
    [ temps_relaxation_coefficient_pdf float]
    [ echelle_relaxation_coefficient_pdf float]
    [ local ]
    [ vitesse_imposee_data champ_base]
    [ vitesse_imposee_fonction n word1 word2 ... wordn]
    [ variable_imposee_data champ_base]
    [ variable_imposee_fonction n word1 word2 ... wordn]
}
where

```

- **eta** *float*: penalization coefficient
- **bilan_pdf** *int*: type de bilan du terme PDF (seul/avec temps/avec convection)
- **temps_relaxation_coefficient_pdf** *float*: time relaxation on the forcing term to help
- **echelle_relaxation_coefficient_pdf** *float*: time relaxation on the forcing term to help convergence
- **local** : whether the prescribed velocity is expressed in the global or local basis
- **vitesse_imposee_data** *champ_base* (16.1): Prescribed velocity as a field
- **vitesse_imposee_fonction** *n word1 word2 ... wordn*: Prescribed velocity as a set of analytical component
- **variable_imposee_data** *champ_base* (16.1): Prescribed variable as a field
- **variable_imposee_fonction** *n word1 word2 ... wordn*: Prescribed variable as a set of analytical component

34.31 Source_pdf_base

Description: Basic class of source_PDF terms introduced in the equation.

See also: Source_dep_inco_bases (34.7) source_pdf (34.29)

Usage:

source_pdf_base *str*

Read *str* {

aire *champ_base*
rotation *champ_base*
[**transpose_rotation**]
modele *bloc_pdf_model*
[**interpolation** *interpolation_ibm_base*]

}

where

- **aire** *champ_base* (16.1): volumic field: a boolean for the cell (0 or 1) indicating if the obstacle is in the cell
- **rotation** *champ_base* (16.1): volumic field with 9 components representing the change of basis on cells (local to global). Used for rotating cases for example.
- **transpose_rotation** : whether to transpose the basis change matrix.
- **modele** *bloc_pdf_model* (34.30): model used for the Penalized Direct Forcing
- **interpolation** *interpolation_ibm_base* (18): interpolation method

34.32 Source_qdm

Description: Momentum source term in the Navier-Stokes equations.

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

Usage:

source_qdm *ch*

where

- **ch** *champ_base* (16.1): Field type.

34.33 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 \(34\)](#)

Usage:

source_qdm_lambdaup *str*

Read *str* {

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

34.34 Source_th_tdivu

Description: This term source is dedicated for any scalar (called T) transport. Coupled with upwind (amont) or muscl scheme, this term gives for final expression of convection : $\text{div}(\mathbf{U}.T) - T.\text{div}(\mathbf{U}) = \mathbf{U}.\text{grad}(T)$ This ensures, in incompressible flow when divergence free is badly resolved, to stay in a better way in the physical boundaries.

Warning: Only available in VEF discretization.

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

Usage:

source_th_tdivu

34.35 Terme_puissance_thermique_echange_impose

Description: Source term to impose thermal power according to formula : $P = \text{himp} * (T - \text{Text})$. Where T is the Trust temperature, Text is the outside temperature with which energy is exchanged via an exchange coefficient himp

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

Usage:

terme_puissance_thermique_echange_impose *str*

```
Read str {
    himp champ_base
    Text champ_base
    [ PID_controler_on_targer_power bloc_lecture]
```

```
}
```

where

- **himp** *champ_base* [\(16.1\)](#): the exchange coefficient
- **Text** *champ_base* [\(16.1\)](#): the outside temperature
- **PID_controler_on_targer_power** *bloc_lecture* [\(3.60\)](#): PID_controler_on_targer_power bloc with parameters target_power (required), Kp, Ki and Kd (at least one of them should be provided)

34.36 Travail_pression

Description: Source term which corresponds to the additional pressure work term that appears when dealing with compressible multiphase fluids

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

Usage:

travail_pression

34.37 Vitesse_derive_base

Description: Source term which corresponds to the drift-velocity between a liquid and a gas phase

See also: `vitesse_relative_base` (34.38)

Usage:

vitesse_derive_base

34.38 Vitesse_relative_base

Description: Basic class for drift-velocity source term between a liquid and a gas phase

See also: `source_base` (34) `vitesse_derive_base` (34.37)

Usage:

vitesse_relative_base

35 sous_zone

Synonymous: **sous_domaine**

Description: It is an object type describing a domain sub-set.

A `Sous_Zone` (Sub-area) type object must be associated with a `Domaine` type object. The `Read` (Lire) interpreter is used to define the items comprising the sub-area.

Caution: The `Domain` type object `nom_domaine` must have been meshed (and triangulated or tetrahedralised in VEF) prior to carrying out the `Associate` (Associer) `nom_sous_zone nom_domaine` instruction; this instruction must always be preceded by the `read` instruction.

See also: `objet_u` (40)

Usage:

sous_zone *str*

Read *str* {

```
[ restriction str]  
[ rectangle bloc_origine_cotes]  
[ segment bloc_origine_cotes]  
[ boite bloc_origine_cotes]  
[ liste n n1 n2 ... nn]  
[ fichier str]  
[ intervalle deuxentiers]  
[ polynomes bloc_lecture]  
[ couronne bloc_couronne]  
[ tube bloc_tube]  
[ fonction_sous_zone str]  
[ union str]
```

}

where

- **restriction** *str*: The elements of the sub-area `nom_sous_zone` must be included into the other sub-area named `nom_sous_zone2`. This keyword should be used first in the `Read` keyword.
- **rectangle** *bloc_origine_cotes* (35.1): The sub-area will include all the domain elements whose centre of gravity is within the Rectangle (in dimension 2).

- **segment** *bloc_origine_cotes* (35.1)
- **boite** *bloc_origine_cotes* (35.1): The sub-area will include all the domain elements whose centre of gravity is within the Box (in dimension 3).
- **liste** *n n1 n2 ... nn*: The sub-area will include n domain items, numbers No. 1 No. i No. n.
- **fichier** *str*: The sub-area is read into the file filename.
- **intervalle** *deuxentiers* (35.2): The sub-area will include domain items whose number is between n1 and n2 (where $n1 \leq n2$).
- **polynomes** *bloc_lecture* (3.60): A REPENDRE
- **couronne** *bloc_couronne* (35.3): In 2D case, to create a couronne.
- **tube** *bloc_tube* (35.4): In 3D case, to create a tube.
- **fonction_sous_zone** *str*: Keyword to build a sub-area with the the elements included into the area defined by *fonction*>0.
- **union** *str*: The elements of the sub-area *nom_sous_zone3* will be added to the sub-area *nom_sous_zone*. This keyword should be used last in the Read keyword.

35.1 Bloc_origine_cotes

Description: Class to create a rectangle (or a box).

See also: *objet_lecture* (39)

Usage:

name origin name2 cotes

where

- **name** *str into ['Origine']*: Keyword to define the origin of the rectangle (or the box).
- **origin** *x1 x2 (x3)*: Coordinates of the origin of the rectangle (or the box).
- **name2** *str into ['Cotes']*: Keyword to define the length along the axes.
- **cotes** *x1 x2 (x3)*: Length along the axes.

35.2 Deuxentiers

Description: Two integers.

See also: *objet_lecture* (39)

Usage:

int1 int2

where

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

35.3 Bloc_couronne

Description: Class to create a couronne (2D).

See also: *objet_lecture* (39)

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.

35.4 Bloc_tube

Description: Class to create a tube (3D).

See also: [objet_lecture \(39\)](#)

Usage:

name origin name2 direction name3 ri name4 re name5 h
where

- **name** *str into ['Origine']*: Keyword to define the center of the tube.
- **origin** *x1 x2 (x3)*: Center of the tube.
- **name2** *str into ['dir']*: Keyword to define the direction of the main axis.
- **direction** *str into ['X', 'Y', 'Z']*: direction of the main axis X, Y or Z
- **name3** *str into ['ri']*: Keyword to define the interior radius.
- **ri** *float*: Interior radius.
- **name4** *str into ['re']*: Keyword to define the exterior radius.
- **re** *float*: Exterior radius.
- **name5** *str into ['hauteur']*: Keyword to define the heigth of the tube.
- **h** *float*: Heigth of the tube.

36 turbulence_paro_base

Description: Basic class for wall laws for Navier-Stokes equations.

See also: [objet_u \(40\)](#) [negligeable \(36.1\)](#)

Usage:

36.1 Negligeable

Description: Keyword to suppress the calculation of a law of the wall with a turbulence model. The wall stress is directly calculated with the derivative of the velocity, in the direction perpendicular to the wall ($\tau_{\text{tan}}/\rho = \nu \, dU/dy$).

Warning: This keyword is not available for k-epsilon models. In that case you must choose a wall law.

See also: [turbulence_paro_base \(36\)](#)

Usage:

negligeable

37 turbulence_paro_scalaire_base

Description: Basic class for wall laws for energy equation.

See also: [objet_u \(40\)](#) [negligeable_scalaire \(37.1\)](#)

Usage:

37.1 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_paro_i_scalaire_base` ([37](#))

Usage:

negligeable_scalaire

38 listobj_impl

Description: `not_set`

See also: `objet_u` ([40](#)) `listobj` ([38.5](#))

Usage:

38.1 Milieu_musig

Description: MUSIG medium made of several sub mediums.

See also: `listobj` ([38.5](#))

Usage:

{ `object1` `object2` }

list of *milieu_base* ([22](#))

38.2 Milieu_composite

Description: Composite medium made of several sub mediums.

See also: `listobj` ([38.5](#))

Usage:

{ `object1` `object2` }

list of *milieu_base* ([22](#))

38.3 List_un_pb

Description: pour les groupes

See also: `listobj` ([38.5](#))

Usage:

{ `object1` , `object2` }

list of *un_pb* ([38.4](#)) separated with ,

38.4 Un_pb

Description: pour les groupes

See also: `objet_lecture` ([39](#))

Usage:

mot

where

- **mot** *str*: the string

38.5 Listobj

Description: List of objects.

See also: [listobj_impl \(38\)](#) [listchamp_generique \(9.2\)](#) [champs_a_post \(4.2.2\)](#) [definition_champs \(4.2.4\)](#) [sondes \(4.2.7\)](#) [list_stat_post \(4.2.30\)](#) [post_processings \(4.3\)](#) [liste_post_ok \(4.4\)](#) [liste_post \(4.5\)](#) [list_un_pb \(38.3\)](#) [list_list_nom \(4.24\)](#) [condlims \(5.4\)](#) [condinits \(5.5\)](#) [sources \(5.6\)](#) [Milieu_composite \(38.2\)](#) [Milieu_MUSIG \(38.1\)](#) [listeqn \(4.12\)](#) [reactions \(10.1\)](#) [list_nom_virgule \(9.3\)](#) [listsous_zone_valeur \(5.2.11\)](#) [list_info_med \(4.54\)](#) [list_bord \(3.72.4\)](#) [list_bloc_mailler \(3.72\)](#) [vect_nom \(3.130\)](#) [list_nom \(3.115\)](#) [listpoints \(4.2.11\)](#) [coarsen_operators \(3.80\)](#)

Usage:

39 objet_lecture

Description: Auxiliary class for reading.

See also: [objet_u \(40\)](#) [bloc_lecture \(3.60\)](#) [deuxmots \(5.34\)](#) [troismots \(6.2\)](#) [quatremots \(39.1\)](#) [deuxentiers \(35.2\)](#) [floatfloat \(5.36\)](#) [entierfloat \(39.2\)](#) [bloc_lecture_poro \(28.1\)](#) [postraitement_base \(4.4.2\)](#) [champ_a_post \(4.2.3\)](#) [interface_posts \(4.2.1\)](#) [definition_champ \(4.2.5\)](#) [definition_champs_fichier \(4.2.6\)](#) [sonde_base \(4.2.9\)](#) [sonde \(4.2.8\)](#) [sondes_fichier \(4.2.25\)](#) [champs_posts \(4.2.26\)](#) [bloc_fichier \(4.2.28\)](#) [champs_posts_fichier \(4.2.27\)](#) [stat_post_deriv \(4.2.31\)](#) [stats_posts \(4.2.29\)](#) [stats_posts_fichier \(4.2.37\)](#) [stats_serie_posts \(4.2.38\)](#) [stats_serie_posts_fichier \(4.2.39\)](#) [un_postraitement \(4.3.1\)](#) [nom_postraitement \(4.4.1\)](#) [type_un_post \(4.5.2\)](#) [type_postraitement_ft_lata \(4.5.3\)](#) [un_postraitement_spec \(4.5.1\)](#) [format_file_base \(4.6\)](#) [un_pb \(38.4\)](#) [troisf \(3.53\)](#) [convection_deriv \(5.2.1\)](#) [bloc_convection \(5.2\)](#) [diffusion_deriv \(5.3.1\)](#) [op_implicite \(5.3.17\)](#) [bloc_diffusion \(5.3\)](#) [condlimlu \(5.4.1\)](#) [condinit \(5.5.1\)](#) [parametre_equation_base \(5.7\)](#) [dt_impr_ustar_mean_only \(5.40.1\)](#) [modele_turbulence_hyd_deriv \(5.40\)](#) [form_a_nb_points \(5.40.3\)](#) [traitement_particulier_base \(5.35.1\)](#) [traitement_particulier \(5.35\)](#) [dt_impr_nusselt_mean_only \(23.1\)](#) [type_diffusion_turbulente_multiphase_deriv \(5.3.3\)](#) [bloc_sutherland \(22.7\)](#) [spec_pdc_base \(34.23\)](#) [type_perte_charge_deriv \(34.4\)](#) [reaction \(10.1.1\)](#) [verifiercoin_bloc \(3.133\)](#) [bloc_ef \(5.2.6\)](#) [bloc_origine_cotes \(35.1\)](#) [bloc_couronne \(35.3\)](#) [bloc_tube \(35.4\)](#) [sous_zone_valeur \(5.2.12\)](#) [bloc_diffusion_standard \(5.3.12\)](#) [info_med \(4.54.1\)](#) [bloc_lec_champ_init_canal_sinal \(16.20\)](#) [fonction_champ_reprise \(16.16\)](#) [bord_base \(3.72.5\)](#) [def_bord \(3.72.7\)](#) [mailler_base \(3.72.1\)](#) [bloc_pave \(3.72.3\)](#) [lecture_bloc_moment_base \(3.25\)](#) [un_point \(3.25.3\)](#) [remove_elem_bloc \(3.103\)](#) [bloc_decouper \(3.85\)](#) [bloc_pdf_model \(34.30\)](#) [format_lata_to_med \(3.67\)](#) [format_lata_to_CGNS \(3.65\)](#) [Coarsen_Operator_Uniform \(3.80.1\)](#)

Usage:

39.1 Quatremots

Description: Three words.

See also: [objet_lecture \(39\)](#)

Usage:

mot_1 mot_2 mot_3 mot_4

where

- **mot_1** *str*: First word.
- **mot_2** *str*: Snd word.
- **mot_3** *str*: Third word.
- **mot_4** *str*: Fourth word.

39.2 Entierfloat

Description: An integer and a real.

See also: `objet_lecture` ([39](#))

Usage:

the_int the_float

where

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

40 index

Index

/*, 216
#, 236
 , 30, 56, 59, 155, 161, 193, 311, 382
aire_interfaciale , 162
associer , 27
champ_post_interpolation , 222, 304
champ_post_statistiques_correlation , 90, 220
champ_post_statistiques_ecart_type , 89, 221
champ_post_statistiques_moyenne , 89, 224
champ_uniforme , 268
decoupebord , 31
decouper , 57, 308
decouper_multi , 58
discretiser , 33
divergence , 220
echange_externe_radiatif , 238
ecrire_fichier , 77
extraction , 221
fin , 42
frontiere_ouverte_temperature_imposee , 242
gradient , 222
interpolation_ibm_aucune , 279
interpolation_ibm_element_fluide , 279
interpolation_ibm_gradient_moyen , 281
interpolation_ibm_hybride , 280
interpolation_ibm_power_law_tbl , 281
lata_to_med , 45
lata_to_other , 45
lire , 62
lire_fichier , 62
lire_fichier_bin , 63
lire_med , 26
lml_to_lata , 46
morceau_equation , 223
operateur_eqn , 218
postraitement , 93
postraitements , 92
raffiner_simplexes , 61
rectify_mesh , 64
reduction_0d , 225
refchamp , 226
resoudre , 68
runge_kutta_ordre_4 , 335
schema_euler_explicite , 323
schema_euler_implicite , 357
sous_domaine , 396
temperature_imposee_parois , 251
tparois_vf , 226
transformation , 227
vefprep1b , 252
0 , 67
1 , 67
2 , 67
<= , 49, 50
= , 49
a , 390, 391
a_ext , 239
A_i_ext , 239
all_times , 21
amont , 157
ancien , 177–179
antisym , 156
arrete , 203–207
avec_les_cl , 191, 192, 196–201, 209–213
avec_sources , 191, 192, 196–201, 209–213
avec_sources_et_operateurs , 191, 192, 196–201, 209–213
average , 225
b , 390, 391
binaire , 33, 87, 260
C , 295
C_ext , 239
celsius , 239
centre , 157
cf , 390, 391
cgns , 44, 45, 59, 60, 78, 79, 93
chakravarthy , 157
champ_frontiere , 221, 222
chsom , 82
coarsen_i , 55
coarsen_j , 56
coarsen_k , 56
composante , 227
conservation_masse , 293, 294
constant , 293, 294, 298, 299
coriolis_seul , 384
Cotes , 397
d , 391
debit_total , 43, 44
default , 223
default_bar , 156, 164
diametre , 305
dir , 398
direction , 305
distant , 49
divrhout_moins_Tdivrhout , 177–179
divut_moins_Tdivu , 177–179
domaine , 59
double , 54, 55
dt_integr , 91

dt_post , 87, 88, 90
 edo , 293, 294
 elem , 53, 54, 80, 89, 90, 255–257, 260
 emissivite , 239
 entrainement_seul , 384
 euclidian_norm , 225
 faces , 80, 89, 90
 fichier , 303, 304
 filtrer_resu , 156, 165
 Fluctu_Temperature_ext , 239
 flux_bords , 223, 224
 Flux_Chaleur_Turb_ext , 239
 flux_surfacique_bords , 223, 224
 fonction , 261
 format_post_sup , 44, 45
 formatte , 33, 87, 260
 formule , 227
 grad_Ubar , 165
 grav , 82
 gravcl , 82
 H_ext , 239
 h_imp , 239, 247, 248
 hauteur , 398
 homogene , 49
 implicite , 166
 integrale_en_z , 43, 44
 K , 391
 k , 249
 K_Eps_ext , 239
 k_ext , 239
 K_Omega_ext , 239
 kelvin , 239
 kx , 391
 ky , 391
 kz , 391
 L1_norm , 225
 L2 , 67
 L2_norm , 225
 last_time , 21
 lata , 44, 45, 59, 60, 78, 79, 93
 lata_v2 , 44, 45, 59, 60, 78, 79, 93
 left_value , 225
 lml , 44, 45, 59, 60, 78, 79, 93
 local , 49
 max , 67, 225
 med , 44, 45, 59, 60, 78, 79, 93
 min , 225
 minmod , 157
 mixed , 54, 55
 moins_rho_moyen , 293, 294
 moyenne , 225
 moyenne_ponderee , 225
 mpi-io , 78, 79, 93, 94
 mu0 , 295
 multiple , 78, 79, 93, 94
 muscl , 157
 natural , 313
 nb_pas_dt_post , 87, 88, 90
 no , 223
 nodes , 82
 non , 56
 normalized_euclidian_norm , 225
 norme , 227
 nu , 165
 nu_transp , 165
 nut , 165
 nut_transp , 165
 omega_ext , 239
 Origine , 397, 398
 oui , 56
 pdi , 260
 periode , 82
 post_processing , 95
 postraitement , 95
 postraitement_ft_lata , 95
 produit_scalaire , 227
 rcm , 313
 re , 398
 ri , 398
 sans_rien , 191, 192, 196–201, 209–213
 scotti , 203–207
 simple , 78, 79, 93, 94
 single_hdf , 260
 single_lata , 59, 60, 78, 79, 93
 Slambda , 295
 solveur , 166
 som , 53, 54, 80, 82, 89, 90, 255–257, 260
 somme , 225
 somme_ponderee , 225
 somme_ponderee_porosite , 225
 stabilite , 223, 224
 standard , 293, 294
 sum , 225
 superbee , 157
 surface , 382
 T0 , 295
 T_ext , 239
 t_ext , 239, 247, 248
 tau_ext , 239
 temperature_unit , 239
 terme_complet , 384
 trace , 221, 222
 transportant_bar , 156
 transporte_bar , 156
 u_tau , 305
 V2_ext , 239
 valeur_a_gauche , 225
 valeur_normale , 276

vanalbada , 157
 vanleer , 157
 vecteur , 227
 visco_cin , 305
 vitesse_parois , 249
 vitesse_tangentielle , 277
 volume , 203–207
 volume_sans_lissage , 203–207
 weighted_average , 225
 weighted_sum , 225
 weighted_sum_porosity , 225
 X , 49, 50, 67, 398
 x , 390
 xyz , 260
 Y , 49, 50, 67, 398
 y , 390
 Y_ext , 239
 yes , 223
 Z , 49, 50, 67, 398
 z , 390
 , 30, 55, 59, 155, 161, 193, 310, 382
 all_options , 56
 champs , 80, 94
 champs_fichier , 80, 94
 conditions_initiales , 154, 169–192, 197, 199, 201, 210, 211, 213
 conditions_limites , 154, 169–178, 180–192, 197, 199, 201, 210, 211, 213
 definition_champs_fichier , 79, 94
 domain , 27
 domaine , 60
 exclude_groups , 27
 fichier , 60, 81, 86
 file , 27
 include_additional_face_groups , 27
 is_multi_scalar , 180, 182
 maillage_vdf , 25
 mesh , 27
 name_of_initial_domaines , 26
 name_of_new_domaines , 26
 par_sous_zone , 20
 partitionneur , 57
 postraitement , 78, 97, 99–102, 104, 105, 107, 108, 110–113, 115, 116, 118–122, 124–127, 129–133, 135–138, 140–142, 144–147, 149–151, 153
 postraitements , 78, 97, 99–102, 104, 105, 107, 108, 110–113, 115, 116, 118–122, 124–127, 129–133, 135–138, 140–142, 144–147, 149–151, 153
 pr_t , 163
 Read_file , 76
 reduction_pression , 364
 sans_dec , 24
 save_matrice , 232–235, 377, 379
 sigma , 163
 sondes , 79, 94
 sondes_fichier , 79, 94
 sondes_mobiles , 79, 94
 sondes_mobiles_fichier , 79, 94
 sous_domaine , 36, 79, 94
 statistiques , 80, 94
 statistiques_en_serie , 80, 94
 statistiques_en_serie_fichier , 80, 94
 statistiques_fichier , 80, 94
 tension_superficielle , 215, 216
 a_res , 388
 acceleration , 384
 aij , 380
 aire , 393, 394
 alias , 180, 182
 alpha , 22, 156, 158, 250
 alpha_0 , 312
 alpha_1 , 312
 alpha_a , 312
 alpha_sous_zone , 158
 montant_sous_zone , 158
 ampli_bruit , 262
 ampli_fluctuation , 276
 ampli_moyenne_imposee , 276
 ampli_moyenne_recyclee , 276
 ampli_sin , 262
 ascii , 26, 70
 atol , 372–375, 377, 380, 381
 avec_certains_bords , 38
 avec_certains_bords_pour_extraire_surface , 37
 avec_les_bords , 38
 bench_ijk_splitting_read , 25
 bench_ijk_splitting_write , 25
 beta , 250, 383, 384
 beta_co , 292, 293
 beta_th , 292, 293
 bilan_pdf , 393
 binaire , 31, 60
 binary_file , 34
 block_size_bytes , 25
 block_size_megabytes , 25
 boite , 397
 bord , 29, 52, 194, 386
 bords_a_decouper , 31
 boundaries , 34, 202, 301
 boundary_conditions , 154, 169–178, 180–192, 197, 199, 201, 210, 211, 213
 boundary_xmax , 52
 boundary_xmin , 52
 boundary_ymax , 52
 boundary_ymin , 52
 boundary_zmax , 52

boundary_zmin , 52
 btd , 160
 c0 , 385
 calc_spectre , 196
 canalx , 207
 centre_rotation , 384
 champ_front_normal_et_tangentiel_robin , 250
 champ_med , 44
 changement_de_base_p1bulle , 253
 check_divergence , 24
 checkpoint_fname , 97
 cl_pression_sommet_faible , 253
 cli , 377, 379
 cli_quiet , 377
 close_every_n , 23
 coarsen_operators , 54
 coef , 288
 coeff , 386, 391
 coefficient_diffusion , 290
 coefficients_activites , 229
 compo , 219, 224
 condition_elements , 36, 38
 condition_faces , 38
 condition_geometrique , 31
 Conduction , 77
 Conduction_ibm , 97
 conservation_Ec , 196
 constante_modele_micro_melange , 228
 constante_taux_reaction , 229
 constituant , 78, 97, 98, 100–103, 105, 107–109, 111–114, 116, 118–123, 125–127, 129–133, 135–139, 141, 142, 144–147, 149–151, 153
 contre_energie_activation , 229
 contre_reaction , 229
 controle_residu , 233, 365–371
 convection , 154, 169–179, 181–190, 192, 197, 199, 201, 210, 211, 213
 convection_diffusion_chaleur_QC , 134, 142
 convection_diffusion_chaleur_turbulent_qc , 145, 150
 convection_diffusion_chaleur_WC , 136, 143
 convection_diffusion_concentration , 98, 110, 121, 122, 137, 138
 convection_diffusion_concentration_turbulent , 100, 111, 124, 125, 140, 141
 convection_diffusion_espece_binaire_QC , 127
 Convection_Diffusion_Espece_Binaire_Turbulent-QC , 130
 convection_diffusion_espece_binaire_WC , 129
 convection_diffusion_temperature , 110, 113, 133, 137, 138, 147
 convection_diffusion_temperature_ibm , 146
 convection_diffusion_temperature_ibm_turbulent , 112
 convection_diffusion_temperature_turbulent , 111, 114, 140, 141, 149, 151
 convertalltopoly , 27
 correction_calcul_pression_initiale , 199
 correction_fraction , 285
 correction_matrice_pression , 199
 correction_matrice_projection_initiale , 198
 correction_pression_modifie , 199
 correction_variable_initiale , 169, 188, 199
 correction_visco_turb_pour_controle_pas_de_temps , 202, 203, 205–208
 correction_visco_turb_pour_controle_pas_de_temps-parametre , 202, 203, 205–208
 correction_vitesse_modifie , 199
 correction_vitesse_projection_initiale , 199
 correlations , 105, 106, 108, 133
 correspondance_elements , 279–282
 corriger_partition , 307
 couronne , 397
 Cp , 282–284, 286, 287
 cp , 35, 245, 246, 285–287, 289–300
 crank , 168
 critere_absolu , 39
 criteres_convergence , 365, 367
 cs , 163, 204
 cstdiff , 162
 Cv , 286, 287, 298
 cw , 162, 205
 deactivate_arete_mixte , 56
 deb , 383
 debit , 245, 246
 debit_impose , 386
 debut_stat , 194
 decoup , 256, 260
 default_value , 255
 definition_champs , 79, 94
 definition_champs_file , 79, 94
 delta , 244
 deprecatedkeepduplicatedprobes , 79, 94
 derivee_rotation , 289
 dh , 246
 diag , 233
 diam_hydr , 388, 389
 diam_hydr_ortho , 389
 diametre_hyd_champ , 289–300
 diffusion , 154, 169–179, 181–192, 197, 199, 201, 210, 211, 213
 diffusion_coef , 283, 284, 286
 diffusion_implicite , 317, 319, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 348, 351, 354, 356, 359, 361, 363
 dim_espace_krilov , 233

dir , 245, 246, 391
 dir_flow , 262
 dir_wall , 263
 direction , 29, 39–41, 194, 388, 389
 direction_anisotrope , 276
 disable_diphasique , 24
 disable_dt_ev , 318, 320, 323, 325, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 352, 354, 357, 359, 362, 364
 disable_equation_residual , 154, 169–180, 182–190, 192, 197, 199, 201, 210, 211, 213
 disable_progress , 318, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 352, 354, 357, 359, 362, 364
 distance_plan , 276
 dmax , 207
 dom_dist , 255
 dom_loc , 255
 domain , 51, 60, 256, 260
 domaine , 27, 29, 31, 36–41, 79, 94, 222, 223, 308
 domaine_final , 20, 39
 domaine_grossier , 31
 domaine_init , 20, 38
 domaines , 60, 309
 domegadt , 384
 DP0 , 383
 dropping_parameter , 377
 dt , 34
 dt_impr , 202, 245, 246, 301, 317, 319, 322, 324, 326, 327, 329, 331, 333, 336, 337, 339, 342, 343, 345, 348, 351, 353, 356, 358, 361, 363
 dt_impr_moy_spat , 194
 dt_impr_moy_temp , 194
 dt_impr_nusselt , 300–303
 dt_impr_nusselt_mean_only , 300–303
 dt_impr_ustar , 202, 203, 205–208
 dt_impr_ustar_mean_only , 202, 203, 205–208
 dt_max , 317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 348, 351, 353, 356, 358, 361, 363
 dt_min , 317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 348, 351, 353, 356, 358, 361, 363
 dt_post , 79, 94
 dt_projection , 191, 197, 199, 200, 209, 211, 212
 dt_sauv , 317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 348, 351, 353, 356, 358, 361, 363
 dt_start , 317, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 351, 354, 356, 359, 361, 363
 dtol_fraction , 285
 dual , 60
 dv_min , 388
 Ec , 195
 Ec_dans_repere_fixe , 195
 echelle_relaxation_coefficient_pdf , 393
 Echelle_temporelle_turbulente , 105, 107, 108
 ecrire_decoupage , 58
 ecrire_fichier_xyz_valeur , 154, 169–192, 197, 200, 201, 210, 211, 213
 ecrire_frontiere , 60
 ecrire_lata , 58
 ecrire_med , 58
 elements_fluides , 280, 282
 elements_solides , 279–281
 emissivite_pour_rayonnement_entre_deux_plaques-quasi_infinies , 247
 energie_activation , 229
 Energie_cinetique_turbulente , 105, 107, 108
 Energie_cinetique_turbulente_WIT , 105, 107, 108
 Energie_Multiphase , 105, 108
 Energie_Multiphase_h , 106
 enthalpie_reaction , 229
 epaisseur , 37, 39
 eps , 383
 epsilon , 315
 equation_frequence_resolue , 168
 equation_non_resolue , 154, 168–192, 198, 200, 201, 210, 211, 213
 equations_scalaires_passifs , 119, 122, 125, 138, 141–143, 145, 147, 151
 espece , 184, 186
 espece_en_competition_micro_melange , 228
 est_dirichlet , 279–281
 eta , 393
 evanescence , 176
 exclure_groupes , 27
 exp_res , 388
 exposant_beta , 229
 expression , 228
 expression_p_ana , 79, 93
 expression_vx_ana , 79, 93
 expression_vy_ana , 79, 93
 expression_vz_ana , 79, 93
 facon_init , 195
 facsec , 317, 319, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 348, 351, 353, 356, 359, 361, 363
 facsec_diffusion_for_sets , 365, 368
 facsec_expert , 358
 facsec_func , 358
 facsec_ini , 41
 facsec_max , 41, 319, 321, 347, 350, 352, 355, 358
 facteur , 160, 161
 facteurs , 48

fichier , 27, 79, 87, 93, 207, 258, 269, 277, 307, 308, 397
 fichier_ecriture_K_Eps , 207
 fichier_matrice , 70
 fichier_post , 30
 fichier_secmem , 69
 fichier_solution , 70
 fichier_solveur , 70
 fichier_solveur_non_recree , 234
 fichier_sortie , 44
 fichier_ssz , 308
 field , 256, 257, 260, 306
 fields , 34, 80, 94
 fields_file , 80, 94
 file , 60, 81, 86, 256, 260, 306
 file_coord_x , 52
 file_coord_y , 52
 file_coord_z , 52
 file_coords , 214
 file_per_comm_group , 23
 filling , 311
 fin_stat , 194
 flow_rate , 278
 fluid , 282–284
 fluide_incompressible , 98, 100–103, 109, 111–114, 120–123, 125, 126, 131, 133, 137–139, 141, 146–148, 151
 fluide_ostwald , 133, 146
 fluide_quasi_compressible , 127, 130, 134, 142, 145, 150
 fluide_sodium_gaz , 133
 fluide_sodium_liquide , 133
 fluide_weakly_compressible , 128, 136, 143
 flush_every_n , 23
 flux_parois , 236
 fonction , 65
 fonction_filtre , 54
 fonction_sous_zone , 397
 force , 233
 format , 60, 79, 93
 format_post , 54
 formulation_a_nb_points , 203–205, 207
 formulation_linear_pwl , 281
 frequence_recalc , 234
 function_coord_x , 52
 function_coord_y , 52
 function_coord_z , 52
 gamma , 286, 287, 298
 gas_turb , 163
 genere_fichier_solveur , 70
 ghost_size , 55
 ghost_thickness , 51
 gnuplot_header , 318, 320, 323, 325, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 352, 354, 357, 359, 362, 364
 gradient_pression_qdm_modifie , 199
 gram_schmidt , 24
 gravite , 289–300
 group_nb , 22
 groupes , 117
 h , 262, 386
 hexa_old , 39
 himp , 395
 Hlsat , 216
 Hvsat , 216
 ignore_check_fraction , 285
 ijk_splitting_ft_extension , 214
 impr , 55, 70, 230, 232–235, 279–282, 289, 372–375, 377, 380, 381
 impr_diffusion_implicite , 317, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 351, 354, 356, 359, 361, 363
 impr_extremums , 317, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 351, 354, 356, 359, 361, 363
 inclure_groupes_faces_additionnels , 27
 indice , 290–299
 info , 164
 init_Ec , 195
 initial_conditions , 154, 169–192, 197, 199, 201, 210, 211, 213
 initial_field , 264
 initial_value , 263, 264, 271
 input_field , 264
 interfaces , 79, 93
 interp_ve1 , 25
 interpolation , 393, 394
 intervalle , 397
 inverse_condition_element , 37
 is_multi_scalar , 290
 is_multi_scalar_diffusion , 180, 182
 iter_max , 365, 368
 iter_min , 365, 368
 iterations_mixed_solver , 55
 joints_non_postraites , 60
 k , 293
 kappa , 290–299
 kmetis , 307
 l_melange , 162
 lambda , 245, 246, 289–300, 388, 389, 394
 lambda_max , 395
 lambda_min , 395
 lambda_ortho , 388, 389
 larg_joint , 57
 last_time , 256, 260
 level , 313–315
 Lire_fichier , 76

list_equations , 102, 103, 113, 114, 118
liste , 65, 397
liste_cas , 35
liste_de_postraitements , 78, 97, 99–102, 104, 105, 107, 108, 110–113, 115, 116, 118–122, 124–127, 129–133, 135–138, 140–142, 144–147, 149–151, 153
liste_postraitements , 78, 97, 99–102, 104, 105, 107, 108, 110–113, 115, 116, 118–122, 124–127, 129–133, 135–138, 140–142, 144–146, 148–151, 153
loc , 256, 257, 260
local , 393
localisation , 54, 223, 228
loi_etat , 294, 299
longueur_boite , 196
longueur_maille , 203–205, 207
longueurs , 47
Lvap , 216
maillage , 27
main , 59
mass_source , 197
masse_molaire , 35, 180, 182
Masse_Multiphase , 105, 106, 108
matrice_pression_penalisee_H1 , 199
max_iter_implicit , 348, 350, 353, 355, 358, 360
mesh , 256, 260
methode , 44, 222, 223, 225, 227
methode_calcul_pression_initiale , 192, 197, 199, 201, 209, 211, 213
milieu , 78, 97, 99–103, 105, 107, 108, 110–113, 115, 116, 118–122, 124–127, 129–134, 136–138, 140–142, 144–147, 149–151, 153
milieu_composite , 105, 106, 108
Milieu_MUSIG , 105, 106, 108
min_dir_flow , 263
min_dir_wall , 263
mobile_probes , 79, 94
mobile_probes_file , 79, 94
mode_calcul_convection , 177, 179
model , 282–284
modele , 393, 394
modele_micro_melange , 228
modele_turbulence , 170, 179, 181, 186, 189, 200, 211, 212
models , 105, 106, 108
modif_div_face_dirichlet , 253
molar_mass , 286
molar_mass1 , 283, 284
molar_mass2 , 283, 284
moyenne_convergee , 224
moyenne_imposee , 277
moyenne_recyclee , 277
mu , 35, 245, 246, 286, 292–294, 298, 299
mu1 , 283, 284
mu2 , 283, 284
n , 246, 293
name_of_initial_zones , 26
name_of_new_zones , 26
name_ssz , 308
nature , 255
navier_stokes_ibm , 126, 146
navier_stokes_ibm_turbulent , 101, 112
navier_stokes_QC , 127, 134, 142
navier_stokes_standard , 98, 102, 109, 113, 120–122, 133, 137, 138, 147
navier_stokes_turbulent , 100, 103, 111, 114, 123, 125, 131, 139, 141, 148, 151
navier_stokes_turbulent_qc , 130, 145, 150
navier_stokes_WC , 128, 136, 143
nb_comp , 263, 264, 271
nb_corrections_max , 365, 366, 368, 369, 371
nb_full_mg_steps , 55
nb_histo_boxes_impr , 279–282
nb_it_max , 232, 233, 235, 365–371, 380
nb_ite_sans_accel_max , 41
nb_nodes , 51
nb_parts , 306–309
nb_parts_geom , 31
nb_parts_naif , 31
nb_parts_tot , 58
nb_pas_dt_max , 317, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 351, 354, 357, 359, 361, 363
nb_pas_dt_post , 79, 94
nb_pas_dt_post_stats_bulles , 79, 93
nb_pas_dt_post_stats_plans , 79, 93
nb_points_par_phase , 194
nb_procs , 35
nb_sauv_max , 317, 319, 322, 324, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 348, 351, 353, 356, 358, 361, 363
nb_test , 70
nb_tranche , 44
nb_tranches , 39–41
nbelem , 214
nbelem_i , 254
nbelem_j , 254
nbelem_k , 254
new_jacobian , 164
ng2 , 162
niter_avg , 319, 321
niter_max , 319, 321
niter_max_diffusion_implicit , 168, 318, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 351, 354, 357, 359, 361, 363
niter_min , 319, 321

nmax , 27
 no_alpha , 163
 no_check_disk_space , 318, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 352, 354, 357, 359, 362, 364
 no_conv_subiteration_diffusion_implicite , 317, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 351, 354, 356, 359, 361, 363
 no_error_if_not_converged_diffusion_implicite , 317, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 351, 354, 356, 359, 361, 363
 no_qdm , 365–371
 nom , 263, 264, 271
 nom_bord , 39
 nom_champ , 255
 nom_cl_derriere , 41
 nom_cl_devant , 41
 nom_domaine , 53
 nom_fichier_post , 54
 nom_fichier_solveur , 234
 nom_fichier_sortie , 31
 nom_frontiere , 222
 nom_inconnue , 180, 181
 nom_pb , 53
 nom_source , 217–224, 226–228
 nom_zones , 58
 nombre_de_noeuds , 47
 noms_champs , 53
 norm , 67
 normal_value , 270
 nproc , 214
 nu , 164, 246, 247
 nu_transp , 164
 numero , 223, 228
 numero_masse , 219
 numero_op , 219
 numero_source , 219
 nut , 164
 nut_max , 202, 203, 205–208
 nut_transp , 164
 old , 158
 omega , 262, 312, 313, 316, 319, 384, 386
 omega_relaxation_drho_dt , 294
 optimisation_sous_maillage , 223
 optimized , 232, 235
 option , 224, 384
 order , 24
 ordering , 313
 origin , 214
 origin_i , 254
 origin_j , 254
 origin_k , 254
 Origine , 47
 origine , 37
 p0 , 253
 p1 , 253
 p_imposee_aux_faces , 56
 P_ref , 216, 296, 297
 p_ref , 215, 216
 P_sat , 216
 pa , 253
 par_sous_dom , 20
 parallel_over_zone , 23
 parallele , 79, 94
 parametre_equation , 154, 169–192, 198, 200, 201, 210, 211, 213
 Partition_tool , 57
 pas_de_solution_initiale , 70
 pb_champ , 225, 226
 pb_champ_evaluateur , 276
 pb_dist , 255
 pb_loc , 255
 pb_name , 59
 pcshell , 380
 penalisation_l2_ftd , 187, 188
 perio , 214
 perio_i , 254
 perio_j , 254
 perio_k , 254
 perio_x , 51
 perio_y , 51
 perio_z , 52
 periode , 195
 periode_calc_spectre , 196
 periode_sauvegarde_securite_en_heures , 318, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 351, 354, 357, 359, 362, 364
 periodique , 58
 petsc_decide , 379
 PID_controler_on_targer_power , 395
 pinf , 298
 point1 , 37
 point2 , 37
 point3 , 37
 points_fluides , 280, 281
 points_solides , 279–282
 polynomes , 397
 porosites , 289–300
 porosites_champ , 289–300
 position , 289
 Post_processing , 78, 97, 99–102, 104, 105, 107, 108, 110–113, 115, 116, 118–122, 124–127, 129–133, 135–138, 140–142, 144–147, 149–151, 153

Post_processings , 78, 97, 99–102, 104, 105, 107, 108, 110–113, 115, 116, 118–122, 124–127, 129–133, 135–138, 140–142, 144–147, 149–151, 153
postraiter_gradient_pression_sans_masse , 192, 197, 199, 201, 209, 211, 213
Pr_t , 163
prandtl_turbulent_fonction_nu_t_alpha , 302
Prandtl , 286, 287
prandtl , 285–287
prandtl_turbulent , 163
prdt , 302
pre_smooth_steps , 55
precision_impr , 318, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 351, 354, 357, 359, 362, 363
precond , 232, 235, 372, 375, 379, 380
precond0 , 312
precond1 , 312
precond_diagonal , 232, 235
precond_nul , 232, 235, 379
preconda , 312
preconditionnement_diag , 168
pression , 294
pression_degeneree , 364
pression_thermo , 299
pression_xyz , 299
pressure_order , 24
pressure_reduction , 364
print_more_infos , 58
probes , 79, 94
probes_file , 79, 94
probleme , 36–38, 263, 264, 271
produits , 229
projection_initiale , 192, 197, 199, 201, 209, 211, 213
projection_normale_bord , 39
pulsation_w , 194
q , 298
q_prim , 298
QDM_Multiphase , 105, 106, 108
quiet , 230, 232–235, 372–375, 377, 380, 381
rapport_residus , 41
reactifs , 229
reactions , 228
read_matrix , 379
rectangle , 396
reduce_ram , 377
regul , 391
relative , 67
relax_jacobi , 55
relax_pression , 369, 371
renommer_equation , 154, 169–192, 198, 200, 201, 210, 212, 213
reorder , 58
reorder_matrix , 379
reprise , 78, 98–102, 104, 105, 107, 109–112, 114–116, 118–121, 123–126, 128–132, 134–137, 139–141, 143–146, 148–151, 153, 194
reprise_correlation , 246, 247
residuals , 317, 319, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 348, 351, 354, 356, 359, 361, 363
resolution_explicite , 168
resolution_monolithique , 358
restriction , 396
resume_last_time , 78, 98–101, 103, 104, 106, 107, 109–112, 114–116, 118–120, 122–126, 128–132, 134–137, 139–141, 143–145, 147–150, 152, 153
reuse_preconditioner_nb_it_max , 379, 380
rho , 245, 246, 289–300
rho_constant_pour_debug , 286
rho_t , 287
rho_xyz , 287
rotation , 289, 393, 394
rt , 253
rtol , 372–375, 377, 379–381
sans_passer_par_le2d , 39
sans_solveur_masse , 219
sauvegarde , 78, 98–102, 104, 105, 107, 109–113, 115, 116, 118–121, 123–127, 129–133, 135–137, 139–141, 143–146, 148–151, 153
sauvegarde_simple , 78, 98–102, 104, 105, 107, 109–112, 114–116, 118–121, 123–126, 128–132, 134–137, 139–141, 143–146, 148–151, 153
save_matrix , 232–235, 377, 379
save_matrix_mtx_format , 372–375, 377, 380, 381
save_matrix_petsc_format , 377, 380
sc , 285
scturb , 303
segment , 396
serial_statistics , 80, 94
serial_statistics_file , 80, 94
seuil , 55, 232–235, 319, 321, 371–375, 377, 380, 381
seuil_convergence_implicit , 168, 365, 366, 368, 369, 371
seuil_convergence_solveur , 168, 365–371
seuil_diffusion_implicit , 168, 317, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 351, 354, 356, 359, 361, 363
seuil_divU , 192, 197, 199, 201, 209, 211, 212
seuil_generation_solveur , 365–371
seuil_statio , 317, 319, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346,

348, 351, 353, 356, 359, 361, 363
 seuil_test_preliminaire_solveur , 365–371
 seuil_verification , 70
 seuil_verification_solveur , 365–371
 sharing_algo , 24
 sigma_turbulent , 163
 single_hdf , 26, 58
 single_precision , 23
 single_safe_file , 23
 size_dom , 214
 smooth_steps , 55
 solide , 77, 97
 solv_elem , 232
 solver_precision , 55
 solveur , 70, 168, 348, 350, 353, 355, 358, 360, 365–371
 solveur0 , 231
 solveur1 , 232
 solveur_bar , 192, 197, 199, 201, 209, 211, 213
 solveur_grossier , 55
 solveur_pression , 176, 191, 197, 199, 200, 209, 211, 212
 source , 217–224, 226–228
 source_reference , 217–224, 226–228
 sources , 154, 169–192, 197, 199, 201, 210, 211, 213, 217–224, 226–228
 sources_reference , 217–224, 226–228
 sous_zone , 36, 79, 94, 263, 264, 271, 388, 389
 sous_zones , 309
 species_number , 286
 spectre_1D , 196
 spectre_3D , 196
 splitting , 51
 standard , 164
 statistics , 80, 94
 statistics_file , 80, 94
 suffix_for_reset , 80, 94
 surface , 247, 391
 surface_tension , 215, 216
 surfacic_flux , 52
 surfacique , 311
 sutherland , 294, 299
 symx , 48
 symy , 48
 symz , 48
 t0 , 385
 t_deb , 219–221, 224
 t_debut_statistiques , 79, 93
 t_fin , 219–221, 224
 t_min , 287
 T_ref , 216, 296, 297
 t_ref , 215, 216
 T_sat , 216
 table_temps , 255
 table_temps_lue , 255
 Taux_dissipation_turbulent , 105, 107, 108
 tcpumax , 317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 348, 350, 353, 356, 358, 361, 363
 tdivu , 158
 temperature , 283, 284
 temperature_order , 24
 temperature_paroil , 236
 temps_debut_prise_en_compte_drho_dt , 294
 temps_relaxation_coefficient_pdf , 393
 test , 158
 Text , 395
 time , 256, 257, 260
 time_activate_ptot , 299
 tinf , 245, 246
 tinit , 317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 348, 350, 353, 356, 358, 361, 362
 tmax , 317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 348, 350, 353, 356, 358, 361, 363
 toutes_les_options , 56
 traitement_axi , 25
 traitement_coins , 56
 traitement_gradients , 56
 traitement_particulier , 192, 197, 199, 200, 209, 211, 212
 traitement_pth , 294, 299
 traitement_rho_gravite , 294
 tranches , 309
 transpose_rotation , 393, 394
 triangle , 37
 trois_tetra , 39
 tsup , 245, 246
 tube , 397
 turbulence_paroil , 202, 203, 205–208, 300, 302, 303
 tuyauz , 207
 type , 223, 311
 u_etoile , 386
 ubar_umprim_cible , 395
 ucent , 262
 uniform_domain_size_i , 254
 uniform_domain_size_j , 254
 uniform_domain_size_k , 254
 union , 397
 unite , 224, 228
 use_existing_domain , 256, 260
 use_grad_pression_eos , 299
 use_hydrostatic_pressure , 299
 use_links , 23
 use_osqp , 25
 use_overlapdec , 255

[use_total_pressure](#) , 299
[use_weights](#) , 307
[user_field](#) , 300
[val_Ec](#) , 195
[variable_imposee_data](#) , 393
[variable_imposee_fonction](#) , 393
[vdf_mesh](#) , 25
[velocity_order](#) , 24
[velocity_profil](#) , 278
[verif_boussinesq](#) , 385
[verification_derivee](#) , 289
[via_extraire_surface](#) , 37
[vingt_tetra](#) , 39
[vitesse](#) , 289, 384
[vitesse_imposee_data](#) , 393
[vitesse_imposee_fonction](#) , 393
[volume](#) , 246
[volumes_etendus](#) , 158
[volumes_non_etendus](#) , 158
[volumique](#) , 311
[without_dec](#) , 24
[writing_processes](#) , 25
[xinf](#) , 247
[xsup](#) , 247
[xtanh](#) , 48
[xtanh_dilatation](#) , 48
[xtanh_taille_premiere_maille](#) , 48
[yaml_fname](#) , 97
[ytanh](#) , 48
[ytanh_dilatation](#) , 48
[ytanh_taille_premiere_maille](#) , 48
[zmax](#) , 44
[zmin](#) , 44
[ztanh](#) , 48
[ztanh_dilatation](#) , 48
[ztanh_taille_premiere_maille](#) , 48

Acceleration, 384
 Ale, 155
 Amg, 229
 Amgx, 229
 Amont, 159
 Amont_old, 157
 Analyse_angle, 27
 Associate, 27
 Axi, 28

[Bicgstab](#), 372
[Bidim_axi](#), 28
[Binaire](#), 95
[Binaire_gaz_parfait_qc](#), 283
[Binaire_gaz_parfait_wc](#), 283
[Block_jacobi_icc](#), 313
[Block_jacobi_ilu](#), 313

[Boomeramg](#), 314
[Bord](#), 50
[Bord_base](#), 48
[Boundary_field_inward](#), 270
[Boussinesq_concentration](#), 384
[Boussinesq_temperature](#), 385
[Btd](#), 160

[C-amg](#), 314
[Calcul](#), 28
[Calculer_moments](#), 28
[Canal](#), 194
[Canal_perio](#), 385
[Centre](#), 160
[Centre4](#), 160
[Centre_de_gravite](#), 29
[Centre_old](#), 159
[Ch_front_input](#), 270
[Ch_front_input_uniforme](#), 271
[Champ_base](#), 254
[Champ_composite](#), 258
[Champ_don_base](#), 258
[Champ_don_lu](#), 258
[Champ_fonc_fonction](#), 258
[Champ_fonc_fonction_txyz](#), 259
[Champ_fonc_fonction_txyz_morceaux](#), 259
[Champ_fonc_interp](#), 254
[Champ_fonc_med](#), 259
[Champ_fonc_med_table_temps](#), 255
[Champ_fonc_med_tabule](#), 256
[Champ_fonc_reprise](#), 260
[Champ_fonc_t](#), 261
[Champ_fonc_tabule](#), 261
[Champ_fonc_tabule_morceaux_interp](#), 257
[Champ_fonc_txyz](#), 267
[Champ_fonc_xyz](#), 267
[Champ_front_base](#), 268
[Champ_front_bruite](#), 271
[Champ_front_calc](#), 272
[Champ_front_composite](#), 272
[Champ_front_contact_vef](#), 272
[Champ_front_debit](#), 273
[Champ_front_debit_massique](#), 273
[Champ_front_debit_qc_vdf](#), 269
[Champ_front_debit_qc_vdf_fonc_t](#), 270
[Champ_front_fonc_pois_ipsn](#), 273
[Champ_front_fonc_pois_tube](#), 274
[Champ_front_fonc_t](#), 274
[Champ_front_fonc_txyz](#), 274
[Champ_front_fonc_xyz](#), 274
[Champ_front_fonction](#), 274
[Champ_front_lu](#), 275
[Champ_front_med](#), 271
[Champ_front_musig](#), 275

Champ_front_normal_vef, 275
 Champ_front_parametrique, 269
 Champ_front_pression_from_u, 276
 Champ_front_recyclage, 276
 Champ_front_tabule, 277
 Champ_front_tabule_lu, 277
 Champ_front_tangentiel_vef, 277
 Champ_front_uniforme, 277
 Champ_front_xyz_debit, 278
 Champ_front_xyz_tabule, 269
 Champ_generique_base, 217
 Champ_init_canal_sinal, 262
 Champ_input_base, 263
 Champ_input_p0, 263
 Champ_input_p0_composite, 264
 Champ_musig, 264
 Champ_ostwald, 264
 Champ_parametrique, 257
 Champ_post_de_champs_post, 217
 Champ_post_extraction, 221
 Champ_post_morceau_equation, 223
 Champ_post_operateur_base, 218
 Champ_post_operateur_divergence, 220
 Champ_post_operateur_eqn, 218
 Champ_post_operateur_gradient, 222
 Champ_post_reduction_0d, 225
 Champ_post_refchamp, 226
 Champ_post_statistiques_base, 219
 Champ_post_tparoi_vef, 226
 Champ_post_transformation, 227
 Champ_som_lu_vdf, 265
 Champ_som_lu_vef, 265
 Champ_tabule_lu, 265
 Champ_tabule_morceaux, 257
 Champ_tabule_temps, 266
 Champ_uniforme_morceaux, 266
 Champ_uniforme_morceaux_tabule_temps, 266
 Champ_front_fonc_txyz, 17
 Chimie, 228
 Chmoy_faceperio, 194
 Cholesky, 230, 375
 Cholesky_mumps_blr, 377
 Cholesky_out_of_core, 372
 Cholesky_pastix, 373
 Cholesky_superlu, 373
 Cholesky_umfpack, 374
 Circle, 86
 Circle_3, 86
 Class_generic, 229
 Cli, 377
 Cli_quiet, 378
 Condinits, 166
 Condlim_base, 236
 Condlims, 166
 Conduction, 154
 Conduction_ibm, 168
 Connexion_approchee, 303
 Connexion_exacte, 304
 Constituant, 289
 Convection_deriv, 155
 Convection_diffusion_chaleur_qc, 177
 Convection_diffusion_chaleur_turbulent_qc, 179
 Convection_diffusion_chaleur_wc, 178
 Convection_diffusion_concentration, 180
 Convection_diffusion_concentration_turbulent, 181
 Convection_diffusion_espece_binaire_qc, 182
 Convection_diffusion_espece_binaire_turbulent_qc, 169
 Convection_diffusion_espece_binaire_wc, 183
 Convection_diffusion_espece_multi_qc, 184
 Convection_diffusion_espece_multi_turbulent_qc, 185
 Convection_diffusion_espece_multi_wc, 184
 Convection_diffusion_temperature, 186
 Convection_diffusion_temperature_ibm, 187
 Convection_diffusion_temperature_ibm_turbulent, 188
 Convection_diffusion_temperature_turbulent, 189
 Coolprop_qc, 284
 Coolprop_wc, 284
 Coriolis, 386
 Correction_antal, 382
 Correction_tomiyama, 382
 Correlation, 88–90, 219
 Corriger_frontiere_periodique, 29
 Create_domain_from_sub_domain, 20
 Cudss, 230
 Darcy, 386
 Debog, 30
 Decoupebord_pour_rayonnement, 31
 Decouper_bord_coincident, 31
 Dg, 251
 Di_12, 156
 Diag, 314
 Diffusion_deriv, 161
 Dilate, 32
 Dimension, 32
 Dirac, 387
 Dirichlet, 238
 Disable_tu, 32
 Discretisation_base, 251
 Discretiser_domaine, 32
 Discretize, 32
 Dispersion_bulles, 383
 Distance_parois, 33
 Domain, 50
 Domaine, 253
 Domaine_base, 213
 Domaine_ijk, 213

Domaineaxi1d, 253
 Dp, 382
 Dp_impose, 382
 Dp_regul, 383
 Dt_calc, 230
 Dt_fixe, 230
 Dt_min, 231
 Dt_start, 231
 Dt_post, 87, 90

 Ec, 195
 Ecart_type, 89, 221
 Ecart_type, 88, 90
 Echange_couplage_thermique, 236
 Echelle_temporelle_turbulente, 170
 Ecrire, 76
 Ecrire_champ_med, 33
 Ecrire_fichier_bin, 77
 Ecrire_fichier_formatte, 33
 Ecriturelecturespecial, 34
 Ef, 156, 251
 Ef_axi, 251
 Ef_stab, 157
 Eisentat, 313
 End, 41
 Energie_cinetique_turbulente, 173
 Energie_cinetique_turbulente_wit, 174
 Energie_multiphase, 171
 Energie_multiphase_h, 172
 Enthalpie_imposee_paroι, 251
 Entree_temperature_imposee_h, 239
 Eos_qc, 282
 Eos_wc, 282
 Epsilon, 50
 Eqn_base, 190
 Execute_parallel, 35
 Export, 35
 Extract_2d_from_3d, 35
 Extract_2daxi_from_3d, 36
 Extraire_domaine, 36
 Extraire_plan, 36
 Extraire_surface, 37
 Extrudebord, 38
 Extrudeparoi, 39
 Extruder, 39
 Extruder_en20, 40
 Extruder_en3, 40

 Fichier_decoupage, 306
 Fichier_med, 306
 Fluide_base, 290
 Fluide_dilatable_base, 291
 Fluide_incompressible, 291
 Fluide_ostwald, 292
 Fluide_quasi_compressible, 293
 Fluide_reel_base, 295
 Fluide_sodium_gaz, 295
 Fluide_sodium_liquide, 296
 Fluide_stiffened_gas, 297
 Fluide_weakly_compressible, 298
 Flux_interfacial, 387
 Forchheimer, 387
 Formatte, 96
 Frontiere_ouverte, 239
 Frontiere_ouverte_alpha_impose, 239
 Frontiere_ouverte_concentration_imposee, 239
 Frontiere_ouverte_enthalpie_imposee, 242
 Frontiere_ouverte_fraction_massique_imposee, 240
 Frontiere_ouverte_gradient_pression_impose, 240
 Frontiere_ouverte_gradient_pression_impose_vefprep1b, 240
 Frontiere_ouverte_gradient_pression_libre_vef, 240
 Frontiere_ouverte_gradient_pression_libre_vefprep1b, 241
 Frontiere_ouverte_pression_imposee, 241
 Frontiere_ouverte_pression_imposee_orlansky, 241
 Frontiere_ouverte_pression_moyenne_imposee, 241
 Frontiere_ouverte_rho_u_impose, 242
 Frontiere_ouverte_vitesse_imposee, 242
 Frontiere_ouverte_vitesse_imposee_sortie, 242
 Frottement_interfacial, 387

 Gaz_parfait_qc, 286
 Gaz_parfait_wc, 286
 Gcp, 234, 379
 Gcp_ns, 231
 Gen, 232
 Generic, 157
 Gmres, 233, 380

 Ibcgstab, 374
 Ibm_aucune, 279
 Ibm_element_fluide, 279
 Ibm_gradient_moyen, 280
 Ibm_hybride, 280
 Ibm_power_law_tbl, 281
 Ice, 364
 Ijk, 252
 Ijk_grid_geometry, 253
 Ilu, 311
 Implicite, 365
 Imprimer_flux, 42
 Imprimer_flux_sum, 43
 Init_par_partie, 267
 Integrer_champ_med, 43
 Interface_base, 214
 Interface_sigma_constant, 215
 Interfacial_area, 161

Internes, 50
 Interpolation, 222, 304
 Interpolation_ibm_base, 278
 Interpolation_ibm_power_law_tbl_u_star, 278
 Interprete, 19
 Interprete_geometrique_base, 44

 Jacobi, 314

 Kquick, 158

 L_melange, 162
 Lata_2_med, 44
 Lata_2_other, 45
 Lata_to_cgns, 44
 Leap_frog, 325
 Link_cgns_files, 21
 Lire_ideas, 45
 Lire_tgrid, 63
 List_bloc_mailler, 46
 List_bord, 48
 List_nom, 69
 List_nom_virgule, 217
 Liste_post, 94
 Liste_post_ok, 92
 Listobj, 400
 Listobj_impl, 399
 Lml_2_lata, 46
 Logarithmique, 305
 Loi_etat_base, 282
 Loi_etat_gaz_parfait_base, 284
 Loi_etat_gaz_reel_base, 285
 Loi_etat_tppi_base, 285
 Loi_fermeture_base, 288
 Loi_fermeture_test, 288
 Loi_horaire, 288
 Longitudinale, 390
 Longueur_melange, 206
 Lu, 314, 380

 Mailler, 46
 Mailler_base, 46
 Maillerparallel, 51
 Masse_multiphase, 174
 Merge_med, 21
 Metis, 307
 Milieu_base, 289
 Milieu_composite, 399
 Milieu_musig, 399
 Mkdir, 52
 Mod_turb_hyd_ss_maille, 202
 Modele_turbulence_hyd_deriv, 201
 Modele_turbulence_scal_base, 300
 Modif_bord_to_raccord, 52

 Modifydomainexid, 53
 Mor_eqn, 153
 Moyenne, 88–91, 224
 Moyenne_imposee_deriv, 303
 Moyenne_volumique, 53
 Multi_gaz_parfait_qc, 285
 Multi_gaz_parfait_wc, 285
 Multiplefiles, 21
 Muscl, 159
 Muscl3, 155
 Muscl_new, 159
 Muscl_old, 155
 My_comm_group, 21

 Navier_stokes_ibm, 198
 Navier_stokes_ibm_turbulent, 200
 Navier_stokes_qc, 191
 Navier_stokes_standard, 208
 Navier_stokes_turbulent, 210
 Navier_stokes_turbulent_qc, 212
 Navier_stokes_wc, 196
 Negligeable, 159, 165, 398
 Negligeable_scalaire, 398
 Nettoiepasnoeuds, 56
 Neumann, 243
 Neumann_homogene, 237
 Neumann_paro, 237
 Neumann_paro_adiabatique, 238
 Nom, 305
 Null, 208, 301, 315
 Numero_elem_sur_maitre, 84

 Objet_lecture, 400
 Op_conv_ef_stab_polymac_face, 22
 Op_conv_ef_stab_polymac_p0_face, 22
 Op_conv_ef_stab_polymac_p0p1nc_elem, 22
 Op_conv_ef_stab_polymac_p0p1nc_face, 22
 Optimal, 233
 Option, 165
 Option_cgns, 23
 Option_dg, 23
 Option_ijk, 24
 Option_interpolation, 24
 Option_polymac, 24
 Option_vdf, 56
 Orientefacesbord, 56
 Orienter_simplexes, 63

 P1b, 165
 P1ncp1b, 165
 Parallel_io_parameters, 25
 Parametre_diffusion_implicit, 168
 Parametre_equation_base, 167
 Parametre_implicit, 167

Paroi, 238
 Paroi_adiabatique, 243
 Paroi_contact, 243
 Paroi_contact_fictif, 244
 Paroi_decalee_robin, 244
 Paroi_defilante, 245
 Paroi_echange_contact_correlation_vdf, 245
 Paroi_echange_contact_correlation_vef, 246
 Paroi_echange_contact_vdf, 247
 Paroi_echange_externer_impose, 247
 Paroi_echange_externer_impose_h, 247
 Paroi_echange_externer_radiatif, 238
 Paroi_echange_global_impose, 248
 Paroi_echange_interne_global_impose, 236
 Paroi_echange_interne_global_parfait, 237
 Paroi_echange_interne_impose, 237
 Paroi_echange_interne_parfait, 237
 Paroi_fixe, 248
 Paroi_fixe_iso_genepi2_sans_contribution_aux_vitesses_sommets, 248
 Paroi_flux_impose, 248
 Paroi_knudsen_non_negligeable, 249
 Paroi_temperature_imposee, 249
 Partition, 57, 308
 Partition_multi, 58
 Partitionneur_deriv, 306
 Pave, 47
 Pb_avec_liste_conc, 117
 Pb_avec_passif, 118
 Pb_base, 115
 Pb_conduction, 77
 Pb_conduction_ibm, 97
 Pb_gen_base, 77
 Pb_hem, 107
 Pb_hydraulique, 119
 Pb_hydraulique_cloned_concentration, 98
 Pb_hydraulique_cloned_concentration_turbulent, 99
 Pb_hydraulique_concentration, 120
 Pb_hydraulique_concentration_scalaires_passifs, 122
 Pb_hydraulique_concentration_turbulent, 123
 Pb_hydraulique_concentration_turbulent_scalaires_passifs, 124
 Pb_hydraulique_ibm, 125
 Pb_hydraulique_ibm_turbulent, 100
 Pb_hydraulique_list_concentration, 101
 Pb_hydraulique_list_concentration_turbulent, 103
 Pb_hydraulique_melange_binaire_qc, 127
 Pb_hydraulique_melange_binaire_turbulent_qc, 129
 Pb_hydraulique_melange_binaire_wc, 128
 Pb_hydraulique_turbulent, 130
 Pb_multiphase, 104
 Pb_multiphase_h, 106
 Pb_thermohydraulique, 132
 Pb_thermohydraulique_cloned_concentration, 109
 Pb_thermohydraulique_cloned_concentration_turbulent, 110
 Pb_thermohydraulique_concentration, 136
 Pb_thermohydraulique_concentration_scalaires_passifs, 138
 Pb_thermohydraulique_concentration_turbulent, 139
 Pb_thermohydraulique_concentration_turbulent_scalaires_passifs, 140
 Pb_thermohydraulique_especes_qc, 142
 Pb_thermohydraulique_especes_turbulent_qc, 144
 Pb_thermohydraulique_especes_wc, 143
 Pb_thermohydraulique_ibm, 145
 Pb_thermohydraulique_ibm_turbulent, 111
 Pb_thermohydraulique_list_concentration, 113
 Pb_thermohydraulique_list_concentration_turbulent, 114
 Pb_thermohydraulique_qc, 134
 Pb_thermohydraulique_scalaires_passifs, 147
 Pb_thermohydraulique_turbulent, 148
 Pb_thermohydraulique_turbulent_qc, 149
 Pb_thermohydraulique_turbulent_scalaires_passifs, 150
 Pb_thermohydraulique_wc, 135
 Pbc_med, 152
 Pdi, 96
 Pdi_expert, 96
 Periodique, 249
 Perte_charge_anisotrope, 388
 Perte_charge_circulaire, 388
 Perte_charge_directionnelle, 389
 Perte_charge_isotrope, 389
 Perte_charge_reguliere, 389
 Perte_charge_singuliere, 391
 Petsc, 234
 Petsc_gpu, 234
 Pilote_icoco, 59
 Pilut, 315
 Pipecg, 375
 Piso, 366
 Plan, 85
 Point, 83
 Points, 82
 Polyedriser, 59
 Polymac, 252
 Polymac_p0, 252
 Polymac_p0p1nc, 252
 Porosites, 310
 Portance_interfaciale, 383
 Position_like, 85
 Post_processing, 92
 Post_processings, 91
 Postraitement_base, 92
 Postraiter_domaine, 59
 Prandtl, 163, 301
 Precisiongeom, 60

Precond_base, 311
 Preconditionneur_petsc_deriv, 313
 Precondsolv, 311
 Predefini, 224
 Pression, 88, 90, 91
 Problem_read_generic, 152
 Probleme_couple, 116
 Profil, 305
 Profils_thermo, 193
 Puissance_thermique, 391

 Qdm_multiphase, 175
 Quick, 159

 Raccord, 48
 Radioactive_decay, 391
 Radius, 84
 Raffiner_anisotrope, 60
 Raffiner_isotrope, 61
 Raffiner_isotrope_parallele, 25
 Read, 61
 Read_file, 62
 Read_file_binary, 63
 Read_med, 26
 Read_unsupported_ascii_file_from_icem, 63
 Redresser_hexaedres_vdf, 64
 Refine_mesh, 64
 Regroupebord, 64
 Remove_elem, 64
 Remove_invalid_internal_boundaries, 65
 Reordonner, 66
 Reorienter_tetraedres, 66
 Reorienter_triangles, 66
 Rhot_gaz_parfait_qc, 287
 Rhot_gaz_reel_qc, 287
 Robin_vef, 250
 Rotation, 67
 Runge_kutta_ordre_2, 327
 Runge_kutta_ordre_2_classique, 328
 Runge_kutta_ordre_3, 330
 Runge_kutta_ordre_3_classique, 332
 Runge_kutta_ordre_4_classique, 336
 Runge_kutta_ordre_4_classique_3_8, 338
 Runge_kutta_ordre_4_d3p, 334
 Runge_kutta_rationnel_ordre_2, 340

 Sa-amg, 315
 Saturation_base, 215
 Saturation_constant, 215
 Saturation_sodium, 216
 Scalaire_impose_parois, 250
 Scatter, 67
 Scattermed, 68
 Sch_cn_ex_iteratif, 318
 Sch_cn_iteratif, 320
 Schema_adams_bashforth_order_2, 343
 Schema_adams_bashforth_order_3, 344
 Schema_adams_moulton_order_2, 346
 Schema_adams_moulton_order_3, 349
 Schema_backward_differentiation_order_2, 352
 Schema_backward_differentiation_order_3, 354
 Schema_implicite_base, 359
 Schema_predictor_corrector, 362
 Schema_temps_base, 316
 Scheme_euler_explicit, 323
 Scheme_euler_implicit, 357
 Schmidt, 302
 Segment, 83
 Segmentfacesx, 83
 Segmentfacesy, 84
 Segmentfacesz, 84
 Segmentpoints, 83
 Sets, 367
 Sgdh, 163
 Simple, 368
 Simpler, 369
 Single_hdf, 96
 Smago, 162
 Solide, 299
 Solve, 68
 Solveur_implicite_base, 364
 Solveur_lineaire_std, 370
 Solveur_petsc_deriv, 371
 Solveur_sys_base, 235
 Solveur_u_p, 370
 Sonde_base, 82
 Sortie_libre_temperature_imposee_h, 250
 Source_base, 381
 Source_constituant, 392
 Source_dep_inco_bases, 384
 Source_generique, 392
 Source_pdf, 392
 Source_pdf_base, 393
 Source_qdm, 394
 Source_qdm_lambdaup, 394
 Source_th_tdivu, 395
 Sources, 167
 Sous_dom, 308
 Sous_maille_smago, 204
 Sous_maille_wale, 205
 Sous_zone, 396
 Sous_zones, 308
 Spai, 315
 Spec_pdc_r_base, 390
 Ssor, 312, 315
 Ssor_bloc, 312
 Stab, 163
 Standard, 164

Stat_per_proc_perf_log, [68](#)
 Stat_post_deriv, [88](#)
 Statistiques, [87](#), [90](#), [91](#)
 Statistiques_en_serie, [91](#)
 Supg, [160](#)
 Supprime_bord, [68](#)
 Symetrie, [250](#)
 System, [69](#)

 T_deb, [88](#)
 T_fin, [89](#)
 Taux_dissipation_turbulent, [176](#)
 Tayl_green, [267](#)
 Temperature, [193](#)
 Terme_puissance_thermique_echange_impose, [395](#)
 Test_solveur, [69](#)
 Test_sse_kernels, [27](#)
 Testeur, [70](#)
 Testeur_medcoupling, [70](#)
 Tetraedriser, [70](#)
 Tetraedriser_homogene, [70](#)
 Tetraedriser_homogene_compact, [71](#)
 Tetraedriser_homogene_fin, [72](#)
 Tetraedriser_par_prisme, [72](#)
 Thi, [195](#)
 Traitement_particulier_base, [193](#)
 Tranche, [309](#)
 Transformer, [73](#)
 Transversale, [390](#)
 Travail_pression, [395](#)
 Trianguler, [74](#)
 Trianguler_fin, [74](#)
 Trianguler_h, [74](#)
 Turbulence_paro_base, [398](#)
 Turbulence_paro_scalaire_base, [398](#)
 Turbulente, [161](#)
 type, [88](#), [90](#)
 Type_diffusion_turbulente_multiphase_deriv, [161](#)
 Type_perte_charge_deriv, [382](#)

 Uniform_field, [268](#)
 Union, [309](#)

 Valeur_totale_sur_volume, [268](#)
 Vdf, [252](#)
 Vect_nom, [75](#)
 Vef, [252](#)
 Verifier_qualite_raffinements, [75](#)
 Verifier_simplexes, [76](#)
 Verifiercoin, [76](#)
 Vitesse_derive_base, [395](#)
 Vitesse_relative_base, [396](#)
 Volume, [85](#)

 Wale, [162](#)

 Write_med, [20](#)

 Xyz, [96](#)
 xyz, [17](#)