



HAL
open science

Parallel Computational Acoustics Library - Reference Manual

Frédéric Magoulès, François-Xavier Roux

► **To cite this version:**

Frédéric Magoulès, François-Xavier Roux. Parallel Computational Acoustics Library - Reference Manual. [Intern report] A02-R-074 || magoules02c, 2002, 129 p. inria-00099431

HAL Id: inria-00099431

<https://inria.hal.science/inria-00099431v1>

Submitted on 26 Sep 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Parallel Computational Acoustics Library

Reference Manual*

by F. Magoulès and F.-X. Roux

June 13, 2002

Contents

1	Modules	17
1.1	block_size	17
1.2	direct_symmetric	17
1.3	direct_unsymmetric	17
1.4	facet_struct	18
1.5	factorized_matrix_struct	18
1.6	feti_data	19
1.7	feti_param	20
1.8	interf_descriptor_struct	20
1.9	interf_mesh_struct	22
1.10	load_curve_struct	22
1.11	mesh_feti_struct	23
1.12	mesh_struct	23
1.13	model	24
1.14	mpif	25
1.15	nodal_list_struct	25
1.16	outfil	26
1.17	realloc	26
1.18	region_property_struct	27
1.19	solver_param	27
1.20	sparse_matrix_struct	28

* Generated by adoC, awk documenting C, June 13, 2002

2	Functions	29
2.1	add_interf_matrix	29
2.2	add_interf_matrix_structure	29
2.3	add_sparse_sym_matrix	30
2.4	add_sparse_sym_struct	30
2.5	add_sparse_unsym_matrix	31
2.6	add_sparse_unsym_struct	31
2.7	assemble_bc	32
2.8	assemble_facet_bc	32
2.9	assemble_impedance	33
2.10	assemble_interf_matrix	33
2.11	assemble_nodal_bc	34
2.12	bcast_acoustic_property	35
2.13	bcast_admittance_property	35
2.14	bcast_facet	36
2.15	bcast_infinite_property	36
2.16	bcast_load_curve	37
2.17	bcast_model_attribute	37
2.18	bcast_model_parameters	38
2.19	bcast_nodal_list	38
2.20	bcast_solver_parameters	39
2.21	build_diag	39
2.22	build_dirich_aug	40
2.23	build_interf_matrix_structure	40
2.24	build_interface	41
2.25	build_matrix_structure	41
2.26	build_node_eq	42
2.27	build_rev_geometry	42
2.28	build_rev_nodid	43
2.29	build_sparse_sym_struct	43
2.30	build_sparse_unsym_struct	44
2.31	build_topology	44
2.32	calpd	45
2.33	cmk_number	45
2.34	condens_1	45

2.35	condens_2	46
2.36	condens_3	47
2.37	copy_block	47
2.38	dirichlet_bc	48
2.39	distribute_model	48
2.40	echo_facet	49
2.41	echo_mesh	49
2.42	echo_mesh_femview	49
2.43	echo_model	50
2.44	echo_nodal_list	50
2.45	equation_setup	51
2.46	extent_i1	51
2.47	extent_r2	52
2.48	extract_submesh	52
2.49	facet_rhs	53
2.50	factorize_numeric	53
2.51	factorize_symbolic	54
2.52	feti_abort	54
2.53	feti_finalize	55
2.54	feti_init	55
2.55	fill_sym_sparse	56
2.56	fill_unsym_sparse	56
2.57	frontal_numb	57
2.58	full_schur_b	57
2.59	full_blo	58
2.60	full_fwbw	58
2.61	full_ldlt	59
2.62	full_ldlt_b	59
2.63	full_ldu	59
2.64	full_ldu_b	60
2.65	full_schur_b_sparse	60
2.66	full_sto	61
2.67	full_unsto	61
2.68	galerkin_error	62
2.69	galerkin_meshsize	62

2.70	galerkin_tau	63
2.71	gather_nodid	63
2.72	gather_sol	64
2.73	global_eq_mask	64
2.74	global_solution	65
2.75	globaldotproduct	65
2.76	globalsum	66
2.77	globalsumc	66
2.78	globalsumi	67
2.79	half_schur_b	67
2.80	half_blo	68
2.81	half_fwbw	68
2.82	half_mm_b	69
2.83	half_schur_b_sparse	69
2.84	half_sto	70
2.85	half_unsto	70
2.86	ilu_0	70
2.87	ilu_preconditioner	71
2.88	imprim	71
2.89	imprim2	72
2.90	init_model	72
2.91	initialize	73
2.92	inivec	73
2.93	input_data	74
2.94	interf_full_sym_struct	74
2.95	interf_full_unsym_struct	75
2.96	interf_sparse_sym_struct	75
2.97	interf_sparse_unsym_struct	76
2.98	interface_assemb	76
2.99	interface_average	77
2.100	interface_equation	77
2.101	interface_exchange	78
2.102	interface_init	78
2.103	interface_jump	78
2.104	interface_jump2	79

2.105	interface_mesh_equation	80
2.106	interface_mesh_geometry	80
2.107	interface_mesh_topology	81
2.108	interface_node	81
2.109	interface_signe	82
2.110	invD_times_U	82
2.111	invL_times_U	83
2.112	invL_times_sparsU	83
2.113	invUt_times_sparsU	83
2.114	jacobi_convergence	84
2.115	jacobi_optimized	85
2.116	L_times_invD	85
2.117	L_times_invU	86
2.118	LdLt_tri_block_numeric	86
2.119	LdLt_tri_block_symbolic	86
2.120	Ldlt_tri_dirich	87
2.121	LdU_tri_block_numeric	87
2.122	LdU_tri_block_symbolic	88
2.123	LdU_tri_dirich	88
2.124	mask_to_list	89
2.125	merge_nodal_list	89
2.126	mtxv	90
2.127	mtxv_par	90
2.128	mxvadd	91
2.129	nodal_list_equation	91
2.130	opti_number	92
2.131	output_results	92
2.132	point_front	93
2.133	post_process	93
2.134	print_vector	94
2.135	proband	94
2.136	probit	95
2.137	problem_form	95
2.138	problo	96
2.139	proplot	96

2.140	pt_extract	97
2.141	read_mesh	97
2.142	realloc_c1	98
2.143	realloc_c2	98
2.144	realloc_i1	99
2.145	realloc_i2	99
2.146	realloc_r1	100
2.147	realloc_r2	100
2.148	ren_sym_sparse	100
2.149	ren_unsym_sparse	101
2.150	renum_eq_mpc	101
2.151	renum_facet	102
2.152	renum_nodal_list	102
2.153	solve_rhs	103
2.154	son_3d	103
2.155	sort_list	104
2.156	sparse_fwbw	104
2.157	sparse_matrix_equalization	105
2.158	sparse_matrix_equalization2	105
2.159	sparse_prod	106
2.160	split_mesh	106
2.161	stoprun	107
2.162	sym_approx_schur_cpmt	107
2.163	sym_build_schur_cpmt	108
2.164	sym_condens	108
2.165	sym_direct_solve	109
2.166	sym_extract	110
2.167	sym_fill_tridiag_sparse	110
2.168	sym_fwbw_tri_dirichlet	111
2.169	sym_multi_level1_feti	111
2.170	sym_multi_level2_feti	112
2.171	sym_proax	113
2.172	sym_renumb_matrix_bc	114
2.173	sym_tridiag_struct	114
2.174	symfwbw_tri_dirichlet	115

2.175	symunsym_sparse_mask	115
2.176	un_condens	116
2.177	un_extract	116
2.178	unsym_approx_schur_cpmt	117
2.179	unsym_build_schur_cpmt	118
2.180	unsym_direct_solve	118
2.181	unsym_fill_tridiag_sparse	119
2.182	unsym_fwbw_tri_dirichlet	119
2.183	unsym_multi_level1_feti	120
2.184	unsym_multi_level2_feti	120
2.185	unsym_proax	121
2.186	unsym_renumb_matrix_bc	122
2.187	unsym_tridiag_struct	122
2.188	unsymfwbw_tri_dirichlet	123
2.189	unsymunsym_sparse_mask	123
2.190	update_facet	124
2.191	Update_Nodal_List	124
2.192	update_region	125
2.193	write_field	125
2.194	write_field_femview	126
2.195	write_matrix	126
2.196	write_meshfield_femvtk	127
2.197	zorthodir_solver	127
2.198	zorthodir_solver_par	128

Index

Alphabetical

add_interf_matrix, 29
add_interf_matrix_structure, 29
add_sparse_sym_matrix, 30
add_sparse_sym_struct, 30
add_sparse_unsym_matrix, 31
add_sparse_unsym_struct, 31
assemble_bc, 32
assemble_facet_bc, 32
assemble_impedance, 33
assemble_interf_matrix, 33
assemble_nodal_bc, 34
bcast_acoustic_property, 35
bcast_admittance_property, 35
bcast_facet, 36
bcast_infinite_property, 36
bcast_load_curve, 37
bcast_model_attribute, 37
bcast_model_parameters, 38
bcast_nodal_list, 38
bcast_solver_parameters, 39
block_size, 17
build_diag, 39
build_dirich_aug, 40
build_interf_matrix_structure , 40
build_interface, 41
build_matrix_structure, 41
build_node_eq, 42
build_rev_geometry, 42
build_rev_nodid, 43
build_sparse_sym_struct, 43
build_sparse_unsym_struct, 44
build_topology, 44
calpd, 45
cmk_number, 45
condens_1, 45
condens_2, 46
condens_3, 47
copy_block, 47
direct_symmetric, 17
direct_unsymmetric, 17
dirichlet_bc, 48
distribute_model, 48
echo_facet, 49
echo_mesh, 49
echo_mesh_femview, 49
echo_model, 50
echo_nodal_list, 50
equation_setup, 51
extent_i1, 51
extent_r2, 52
extract_submesh, 52
facet_rhs, 53
facet_struct, 18
factorize_numeric, 53
factorize_symbolic, 54
factorized_matrix_struct, 18
feti_abort, 54
feti_data, 19
feti_finalize, 55
feti_init, 55
feti_param, 20
fill_sym_sparse, 56
fill_unsym_sparse, 56
frontal_num, 57
full_blo, 58
full_fwbw, 58
full_ldlt, 59
full_ldlt_b, 59
full_ldu, 59
full_ldu_b, 60
full_schur_b, 57
full_schur_b_sparse, 60
full_sto, 61
full_unsto, 61
galerkin_error, 62
galerkin_meshsize, 62
galerkin_tau, 63
gather_nodid, 63
gather_sol, 64
global_eq_mask, 64
global_solution, 65
global_dotproduct, 65
global_sum, 66
global_sumc, 66
global_sumi, 67
half_blo, 68
half_fwbw, 68
half_mm_b, 69
half_schur_b, 67
half_schur_b_sparse, 69
half_sto, 70
half_unsto, 70
ilu_0, 70
ilu_preconditioner, 71
imprim, 71
imprim2, 72
init_model, 72
initialize, 73

inivec, 73
input_data, 74
interf_descriptor_struct, 20
interf_full_sym_struct, 74
interf_full_unsym_struct, 75
interf_mesh_struct, 22
interf_sparse_sym_struct, 75
interf_sparse_unsym_struct, 76
interface_assemb, 76
interface_average, 77
interface_equation, 77
interface_exchange, 78
interface_init, 78
interface_jump, 78
interface_jump2, 79
interface_mesh_equation, 80
interface_mesh_geometry, 80
interface_mesh_topology, 81
interface_node, 81
interface_signe, 82
invD_times_U, 82
invL_times_sparsU, 83
invL_times_U, 83
invUt_times_sparsU, 83
jacobi_convergence, 84
jacobi_optimized, 85
L_times_invD, 85
L_times_invU, 86
LdLt_tri_block_numeric, 86
LdLt_tri_block_symbolic, 86
Ldlt_tri_dirich, 87
LdU_tri_block_numeric, 87
LdU_tri_block_symbolic, 88
LdU_tri_dirich, 88
load_curve_struct, 22
mask_to_list, 89
merge_nodal_list, 89
mesh_feti_struct, 23
mesh_struct, 23
model, 24
mpif, 25
mtxv, 90
mtxv_par, 90
mxvadd, 91
nodal_list_equation, 91
nodal_list_struct, 25
opti_number, 92
outfil, 26
output_results, 92
point_front, 93
post_process, 93
print_vector, 94
proband, 94
probit, 95
problem_form, 95
problo, 96
problot, 96
pt_extract, 97
read_mesh, 97
realloc, 26
realloc_c1, 98
realloc_c2, 98
realloc_i1, 99
realloc_i2, 99
realloc_r1, 100
realloc_r2, 100
region_property_struct, 27
ren_sym_sparse, 100
ren_unsym_sparse, 101
renum_eq_mpc, 101
renum_facet, 102
renum_nodal_list, 102
solve_rhs, 103
solver_param, 27
son_3d, 103
sort_list, 104
sparse_fwbw, 104
sparse_matrix_equalization, 105
sparse_matrix_equalization2, 105
sparse_matrix_struct, 28
sparse_prod, 106
split_mesh, 106
stoprun, 107
sym_approx_schur_cpmt, 107
sym_build_schur_cpmt, 108
sym_condens, 108
sym_direct_solve, 109
sym_extract, 110
sym_fill_tridiag_sparse, 110
sym_fwbw_tri_dirichlet, 111
sym_multi_level1_feti, 111
sym_multi_level2_feti, 112
sym_proax, 113
sym_renumb_matrix_bc, 114
sym_tridiag_struct, 114
symfwbw_tri_dirichlet, 115
symunsym_sparse_mask, 115
un_condens, 116
un_extract, 116
unsym_approx_schur_cpmt, 117
unsym_build_schur_cpmt, 118
unsym_direct_solve, 118
unsym_fill_tridiag_sparse, 119
unsym_fwbw_tri_dirichlet, 119

unsym_multi_level1_feti, 120
 unsym_multi_level2_feti, 120
 unsym_proax, 121
 unsym_renumb_matrix_bc, 122
 unsym_tridiag_struct, 122
 unsymfwbw_tri_dirichlet, 123
 unsymunsym_sparse_mask, 123
 update_facet, 124
 Update_Nodal_List, 124
 update_region, 125
 write_field, 125
 write_field_femview, 126
 write_matrix, 126
 write_meshfield_femvtk, 127
 zorthodir_solver, 127
 zorthodir_solver_par, 128

Files

add_interf_matrix.f90
 add_interf_matrix, 29
 add_interf_matrix_struct.f90
 add_interf_matrix_structure, 29
 add_sparse_sym_matrix.f90
 add_sparse_sym_matrix, 30
 add_sparse_sym_struct.f90
 add_sparse_sym_struct, 30
 add_sparse_unsym_matrix.f90
 add_sparse_unsym_matrix, 31
 add_sparse_unsym_struct.f90
 add_sparse_unsym_struct, 31
 assemble_bc.f90
 assemble_bc, 32
 assemble_facet_bc.f90
 assemble_facet_bc, 32
 assemble_impedance.f90
 assemble_impedance, 33
 assemble_interf_matrix.f90
 assemble_interf_matrix, 33
 assemble_nodal_bc.f90
 assemble_nodal_bc, 34
 bcast_acoustic_property.f90
 bcast_acoustic_property, 35
 bcast_admittance_property.f90
 bcast_admittance_property, 35
 bcast_facet.f90
 bcast_facet, 36
 bcast_infinite_property.f90
 bcast_infinite_property, 36
 bcast_load_curve.f90
 bcast_load_curve, 37
 bcast_model_attribute.f90
 bcast_model_attribute, 37

bcast_model_parameters.f90
 bcast_model_parameters, 38
 bcast_nodal_list.f90
 bcast_nodal_list, 38
 bcast_solver_parameters.f90
 bcast_solver_parameters, 39
 block_size.f90
 block_size, 17
 build_diag.f90
 build_diag, 39
 build_dirich_aug.f90
 build_dirich_aug, 40
 build_interf_matrix_struct.f90
 build_interf_matrix_structure, 40
 build_interface.f90
 build_interface, 41
 build_matrix_struct.f90
 build_matrix_structure, 41
 build_node_eq.f90
 build_node_eq, 42
 build_rev_geometry.f90
 build_rev_geometry, 42
 build_rev_nodid.f90
 build_rev_nodid, 43
 build_sparse_sym_struct.f90
 build_sparse_sym_struct, 43
 build_sparse_unsym_struct.f90
 build_sparse_unsym_struct, 44
 build_topology.f90
 build_topology, 44
 calpd.f90
 calpd, 45
 cmk_number.f90
 cmk_number, 45
 condens.1.f90
 condens_1, 45
 condens.2.f90
 condens_2, 46
 condens.3.f90
 condens_3, 47
 copy_block.f90
 copy_block, 47
 direct_symmetric.f90
 direct_symmetric, 17
 direct_unsymmetric.f90
 direct_unsymmetric, 17
 dirichlet_bc.f90
 dirichlet_bc, 48
 distribute_model.f90
 distribute_model, 48
 echo_facet.f90
 echo_facet, 49

echo_mesh.f90
 echo_mesh, 49
 echo_mesh_femview.f90
 echo_mesh_femview, 49
 echo_model.f90
 echo_model, 50
 echo_nodal_list.f90
 echo_nodal_list, 50
 equation_setup.f90
 equation_setup, 51
 extent_i1.f90
 extent_i1, 51
 extent_r2.f90
 extent_r2, 52
 extract_submesh.f90
 extract_submesh, 52
 facet_rhs.f90
 facet_rhs, 53
 facet_struct.f90
 facet_struct, 18
 factorize_numeric.f90
 factorize_numeric, 53
 factorize_symbolic.f90
 factorize_symbolic, 54
 factorized_matrix_struct.f90
 factorized_matrix_struct, 18
 feti_abort.f90
 feti_abort, 54
 feti_data.f90
 feti_data, 19
 feti_finalize.f90
 feti_finalize, 55
 feti_init.f90
 feti_init, 55
 feti_param.f90
 feti_param, 20
 fill_sym_sparse.f90
 fill_sym_sparse, 56
 fill_unsym_sparse.f90
 fill_unsym_sparse, 56
 frontal_num_b.f90
 frontal_num_b, 57
 full_blo.f90
 full_blo, 58
 full_fwbw.f90
 full_fwbw, 58
 full_ldt.f90
 full_ldt, 59
 full_ldt_b.f90
 full_ldt_b, 59
 full_ldu.f90
 full_ldu, 59
 full_ldu_b, 60
 full_schur_b.f90
 full_schur_b, 57
 full_schur_b_sparse.f90
 full_schur_b_sparse, 60
 full_sto.f90
 full_sto, 61
 full_unsto.f90
 full_unsto, 61
 galerkin_error.f90
 galerkin_error, 62
 galerkin_meshsize.f90
 galerkin_meshsize, 62
 galerkin_tau.f90
 galerkin_tau, 63
 gather_nodid.f90
 gather_nodid, 63
 gather_sol.f90
 gather_sol, 64
 global_eq_mask.f90
 global_eq_mask, 64
 global_solution.f90
 global_solution, 65
 globaldotproduct.f90
 globaldotproduct, 65
 globalsum.f90
 globalsum, 66
 globalsumc.f90
 globalsumc, 66
 globalsumi.f90
 globalsumi, 67
 half_blo.f90
 half_blo, 68
 half_fwbw.f90
 half_fwbw, 68
 half_mm_b.f90
 half_mm_b, 69
 half_Schur_b.f90
 half_schur_b, 67
 half_schur_b_sparse.f90
 half_schur_b_sparse, 69
 half_sto.f90
 half_sto, 70
 half_unsto.f90
 half_unsto, 70
 ilu_0.f90
 ilu_0, 70
 ilu_preconditioner.f90
 ilu_preconditioner, 71
 imprim.f90
 imprim, 71

imprim2.f90
 imprim2, 72
 init_model.f90
 init_model, 72
 initialize.f90
 initialize, 73
 inivec.f90
 inivec, 73
 input_data.f90
 input_data, 74
 interf_descriptor_struct.f90
 interf_descriptor_struct, 20
 interf_full_sym_struct.f90
 interf_full_sym_struct, 74
 interf_full_unsym_struct.f90
 interf_full_unsym_struct, 75
 interf_mesh_struct.f90
 interf_mesh_struct, 22
 interf_sparse_sym_struct.f90
 interf_sparse_sym_struct, 75
 interf_sparse_unsym_struct.f90
 interf_sparse_unsym_struct, 76
 interface_assemb.f90
 interface_assemb, 76
 interface_average.f90
 interface_average, 77
 interface_equation.f90
 interface_equation, 77
 interface_exchange.f90
 interface_exchange, 78
 interface_init.f90
 interface_init, 78
 interface_jump.f90
 interface_jump, 78
 interface_jump2.f90
 interface_jump2, 79
 interface_mesh_equation.f90
 interface_mesh_equation, 80
 interface_mesh_geometry.f90
 interface_mesh_geometry, 80
 interface_mesh_topology.f90
 interface_mesh_topology, 81
 interface_node.f90
 interface_node, 81
 interface_signe.f90
 interface_signe, 82
 invD_times_U.f90
 invD_times_U, 82
 invl_times_sparsu.f90
 invL_times_sparsU, 83
 invL_times_U.f90
 invL_times_U, 83
 invut_times_sparsu.f90
 invUt_times_sparsU, 83
 jacobi_convergence.f90
 jacobi_convergence, 84
 jacobi_optimized.f90
 jacobi_optimized, 85
 l_times_invD.f90
 L_times_invD, 85
 l_times_invU.f90
 L_times_invU, 86
 ldlt_tri_block_numeric.f90
 LdLt_tri_block_numeric, 86
 ldlt_tri_block_symbolic.f90
 LdLt_tri_block_symbolic, 86
 ldlt_tri_dirich.f90
 Ldlt_tri_dirich, 87
 ldu_tri_block_numeric.f90
 LdU_tri_block_numeric, 87
 ldu_tri_block_symbolic.f90
 LdU_tri_block_symbolic, 88
 ldu_tri_dirich.f90
 LdU_tri_dirich, 88
 load_curve_struct.f90
 load_curve_struct, 22
 mask_to_list.f90
 mask_to_list, 89
 merge_nodal_list.f90
 merge_nodal_list, 89
 mesh_feti_struct.f90
 mesh_feti_struct, 23
 mesh_struct.f90
 mesh_struct, 23
 model.f90
 model, 24
 mpif.f90
 mpif, 25
 mtvx.f90
 mtvx, 90
 mtvx_par.f90
 mtvx_par, 90
 mxvadd.f90
 mxvadd, 91
 nodal_list_equation.f90
 nodal_list_equation, 91
 nodal_list_struct.f90
 nodal_list_struct, 25
 opti_number.f90
 opti_number, 92
 outfil.f90
 outfil, 26
 output_results.f90
 output_results, 92

point_front.f90
 point_front, 93
 post_process.f90
 post_process, 93
 print_vector.f90
 print_vector, 94
 proband.f90
 proband, 94
 probit.f90
 probit, 95
 problem_form.f90
 problem_form , 95
 problo.f90
 problo, 96
 problot.f90
 problot, 96
 pt_extract.f90
 pt_extract, 97
 read_mesh.f90
 read_mesh, 97
 realloc.f90
 realloc, 26
 realloc_c1.f90
 realloc_c1, 98
 realloc_c2.f90
 realloc_c2, 98
 realloc_i1.f90
 realloc_i1, 99
 realloc_i2.f90
 realloc_i2, 99
 realloc_r1.f90
 realloc_r1, 100
 realloc_r2.f90
 realloc_r2, 100
 region_property_struct.f90
 region_property_struct, 27
 ren_sym_sparse.f90
 ren_sym_sparse, 100
 ren_unsym_sparse.f90
 ren_unsym_sparse, 101
 renum_eq_mpc.f90
 renum_eq_mpc, 101
 renum_facet.f90
 renum_facet, 102
 renum_nodal_list.f90
 renum_nodal_list, 102
 solve_rhs.f90
 solve_rhs, 103
 solver_param.f90
 solver_param, 27
 son_3D.f90
 son_3d, 103
 sort_list.f90
 sort_list, 104
 sparse_fwbw.f90
 sparse_fwbw, 104
 sparse_matrix_equalization.f90
 sparse_matrix_equalization, 105
 sparse_matrix_equalization2.f90
 sparse_matrix_equalization2, 105
 sparse_matrix_struct.f90
 sparse_matrix_struct, 28
 sparse_prod.f90
 sparse_prod, 106
 split_mesh.f90
 split_mesh, 106
 stoprun.f90
 stoprun, 107
 sym_approx_schur_cpmt.f90
 sym_approx_schur_cpmt, 107
 sym_build_schur_cpmt.f90
 sym_build_schur_cpmt, 108
 sym_condens.f90
 sym_condens, 108
 sym_direct_solve.f90
 sym_direct_solve, 109
 sym_extract.f90
 sym_extract, 110
 sym_fill_tridiag_sparse.f90
 sym_fill_tridiag_sparse, 110
 sym_fwbw_tri_dirichlet.f90
 sym_fwbw_tri_dirichlet, 111
 sym_multi_level1_feti.f90
 sym_multi_level1_feti, 111
 sym_multi_level2_feti.f90
 sym_multi_level2_feti, 112
 sym_proax.f90
 sym_proax, 113
 sym_renumb_matrix_bc.f90
 sym_renumb_matrix_bc, 114
 sym_tridiag_struct.f90
 sym_tridiag_struct, 114
 symfwbw_tri_dirichlet.f90
 symfwbw_tri_dirichlet, 115
 symunsym_sparse_mask.f90
 symunsym_sparse_mask, 115
 un_condens.f90
 un_condens, 116
 un_extract.f90
 un_extract, 116
 unsym_approx_schur_cpmt.f90
 unsym_approx_schur_cpmt, 117
 unsym_build_schur_cpmt.f90
 unsym_build_schur_cpmt, 118

unsym_direct_solve.f90
 unsym_direct_solve, 118
 unsym_fill_tridiag_sparse.f90
 unsym_fill_tridiag_sparse, 119
 unsym_fwbw_tri_dirichlet.f90
 unsym_fwbw_tri_dirichlet, 119
 unsym_multi_level1_feti.f90
 unsym_multi_level1_feti, 120
 unsym_multi_level2_feti.f90
 unsym_multi_level2_feti, 120
 unsym_proax.f90
 unsym_proax, 121
 unsym_renumb_matrix_bc.f90
 unsym_renumb_matrix_bc, 122
 unsym_tridiag_struct.f90
 unsym_tridiag_struct, 122
 unsymfwbw_tri_dirichlet.f90
 unsymfwbw_tri_dirichlet, 123
 unsymunsym_sparse_mask.f90
 unsymunsym_sparse_mask, 123
 update_facet.f90
 update_facet, 124
 update_nodal_list.f90
 Update_Nodal_List, 124
 update_region.f90
 update_region, 125
 write_field.f90
 write_field, 125
 write_field_femview.f90
 write_field_femview, 126
 write_matrix.f90
 write_matrix, 126
 write_meshfield_femvtk.f90
 write_meshfield_femvtk, 127
 zorthodir_solver.f90
 zorthodir_solver, 127
 zorthodir_solver_par.f90
 zorthodir_solver_par, 128

Sorted

Functions

add_interf_matrix, 29
 add_interf_matrix_structure, 29
 add_sparse_sym_matrix, 30
 add_sparse_sym_struct, 30
 add_sparse_unsym_matrix, 31
 add_sparse_unsym_struct, 31
 assemble_bc, 32
 assemble_facet_bc, 32
 assemble_impedance, 33
 assemble_interf_matrix, 33
 assemble_nodal_bc, 34
 bcast_acoustic_property, 35
 bcast_admittance_property, 35
 bcast_facet, 36
 bcast_infinite_property, 36
 bcast_load_curve, 37
 bcast_model_attribute, 37
 bcast_model_parameters, 38
 bcast_nodal_list, 38
 bcast_solver_parameters, 39
 build_diag, 39
 build_dirich_aug, 40
 build_interf_matrix_structure, 40
 build_interface, 41
 build_matrix_structure, 41
 build_node_eq, 42
 build_rev_geometry, 42
 build_rev_nodid, 43
 build_sparse_sym_struct, 43
 build_sparse_unsym_struct, 44
 build_topology, 44
 calpd, 45
 cmk_number, 45
 condens_1, 45
 condens_2, 46
 condens_3, 47
 copy_block, 47
 dirichlet_bc, 48
 distribute_model, 48
 echo_facet, 49
 echo_mesh, 49
 echo_mesh_femview, 49
 echo_model, 50
 echo_nodal_list, 50
 equation_setup, 51
 extent_i1, 51
 extent_r2, 52
 extract_submesh, 52
 facet_rhs, 53
 factorize_numeric, 53
 factorize_symbolic, 54
 feti_abort, 54
 feti_finalize, 55
 feti_init, 55
 fill_sym_sparse, 56
 fill_unsym_sparse, 56
 frontal_num, 57
 full_blo, 58
 full_fwbw, 58
 full_dlt, 59
 full_dlt_b, 59
 full_du, 59
 full_du_b, 60

full_schur_b, 57
 full_schur_b_sparse, 60
 full_sto, 61
 full_unsto, 61
 galerkin_error, 62
 galerkin_meshsize, 62
 galerkin_tau, 63
 gather_nodid, 63
 gather_sol, 64
 global_eq_mask, 64
 global_solution, 65
 globaldotproduct, 65
 globalsum, 66
 globalsumc, 66
 globalsumi, 67
 half_blo, 68
 half_fwbw, 68
 half_mm_b, 69
 half_schur_b, 67
 half_schur_b_sparse, 69
 half_sto, 70
 half_unsto, 70
 ilu_0, 70
 ilu_preconditioner, 71
 imprim, 71
 imprim2, 72
 init_model, 72
 initialize, 73
 inivec, 73
 input_data, 74
 interf_full_sym_struct, 74
 interf_full_unsym_struct, 75
 interf_sparse_sym_struct, 75
 interf_sparse_unsym_struct, 76
 interface_assemb, 76
 interface_average, 77
 interface_equation, 77
 interface_exchange, 78
 interface_init, 78
 interface_jump, 78
 interface_jump2, 79
 interface_mesh_equation, 80
 interface_mesh_geometry, 80
 interface_mesh_topology, 81
 interface_node, 81
 interface_signe, 82
 invD_times_U, 82
 invL_times_sparsU, 83
 invL_times_U, 83
 invUt_times_sparsU, 83
 jacobi_convergence, 84
 jacobi_optimized, 85
 L_times_invD, 85
 L_times_invU, 86
 LdLt_tri_block_numeric, 86
 LdLt_tri_block_symbolic, 86
 Ldlt_tri_dirich, 87
 LdU_tri_block_numeric, 87
 LdU_tri_block_symbolic, 88
 LdU_tri_dirich, 88
 mask_to_list, 89
 merge_nodal_list, 89
 mtxv, 90
 mtxv_par, 90
 mxvadd, 91
 nodal_list_equation, 91
 opti_number, 92
 output_results, 92
 point_front, 93
 post_process, 93
 print_vector, 94
 proband, 94
 probit, 95
 problem_form, 95
 problo, 96
 problot, 96
 pt_extract, 97
 read_mesh, 97
 realloc_c1, 98
 realloc_c2, 98
 realloc_i1, 99
 realloc_i2, 99
 realloc_r1, 100
 realloc_r2, 100
 ren_sym_sparse, 100
 ren_unsym_sparse, 101
 renum_eq_mpc, 101
 renum_facet, 102
 renum_nodal_list, 102
 solve_rhs, 103
 son_3d, 103
 sort_list, 104
 sparse_fwbw, 104
 sparse_matrix_equalization, 105
 sparse_matrix_equalization2, 105
 sparse_prod, 106
 split_mesh, 106
 stoprun, 107
 sym_approx_schur_cpmt, 107
 sym_build_schur_cpmt, 108
 sym_condens, 108
 sym_direct_solve, 109
 sym_extract, 110
 sym_fill_tridiag_sparse, 110

- sym_fwbw_tri_dirichlet, 111
- sym_multi_level1_feti, 111
- sym_multi_level2_feti, 112
- sym_proax, 113
- sym_renumb_matrix_bc, 114
- sym_tridiag_struct, 114
- symfwbw_tri_dirichlet, 115
- symunsym_sparse_mask, 115
- un_condens, 116
- un_extract, 116
- unsym_approx_schur_cpmt, 117
- unsym_build_schur_cpmt, 118
- unsym_direct_solve, 118
- unsym_fill_tridiag_sparse, 119
- unsym_fwbw_tri_dirichlet, 119
- unsym_multi_level1_feti, 120
- unsym_multi_level2_feti, 120
- unsym_proax, 121
- unsym_renumb_matrix_bc, 122
- unsym_tridiag_struct, 122
- unsymfwbw_tri_dirichlet, 123
- unsymunsym_sparse_mask, 123
- update_facet, 124
- Update_Nodal_List, 124
- update_region, 125
- write_field, 125
- write_field_femview, 126
- write_matrix, 126
- write_meshfield_femvtk, 127
- zorthodir_solver, 127
- zorthodir_solver_par, 128

Modules

- block_size, 17
- direct_symmetric, 17
- direct_unsymmetric, 17
- facet_struct, 18
- factorized_matrix_struct, 18
- feti_data, 19
- feti_param, 20
- interf_descriptor_struct, 20
- interf_mesh_struct, 22
- load_curve_struct, 22
- mesh_feti_struct, 23
- mesh_struct, 23
- model, 24
- mpif, 25
- nodal_list_struct, 25
- outfil, 26
- realloc, 26
- region_property_struct, 27
- solver_param, 27
- sparse_matrix_struct, 28

1 Modules

1.1 **block_size**

NAME

block_size

SYNOPSIS

```
MODULE block_size
```

DESCRIPTION

Block size for factorization of matrices used in the direct solver.

ARGUMENTS

n1 – integer

n2 – integer

1.2 **direct_symmetric**

NAME

direct_symmetric

SYNOPSIS

```
MODULE direct_symmetric
```

DESCRIPTION

Direct method module for symmetric complex matrix.

ARGUMENTS

ldu_Robin – factorized matrix structure

ldu_Dirichlet – factorized matrix structure

MODULES

```
USE factorized_matrix_struct
```

1.3 **direct_unsymmetric**

NAME

direct_unsymmetric

SYNOPSIS

```
MODULE direct_unsymmetric
```

DESCRIPTION

Direct method module for unsymmetric complex matrix.

ARGUMENTS

ldu_Robin – factorized matrix structure
ldu_Dirichlet – factorized matrix structure

MODULES

USE factorized_matrix_struct

1.4 **facet_struct**

NAME

facet_struct

SYNOPSIS

```
MODULE facet_struct
```

DESCRIPTION

Facet structure module.

ARGUMENTS

numb_facets – integer (= number of facets)
numb_quad – integer (= number of 4-node facets)
numb_tria – integer (= number of 3-node facets)
p_geometry – integer array (= pointer of facet connectivity per nodes)
geometry – integer array (= facet connectivity per nodes)
facet_id – integer array (= id of facet)
type_facet – integer array (= type of facet)
id_mat – integer array (= material associated with facet)
id_curve – integer array (= load curve associated with facet)
value – complex array (= average surface value)

1.5 **factorized_matrix_struct**

NAME

factorized_matrix_struct

SYNOPSIS

```
MODULE factorized_matrix_struct
```

DESCRIPTION

Block tridiagonal sparse matrix structure.

ARGUMENTS

neq – integer (dimension of the matrix)
sparse_size – integer (total number of non zero entries in sparse structure)
nfront – integer (= number of fronts of the tridiagonal structure)
nfront_clamp – integer (= number of last clamped fronts i.e. clamped degrees of freedom)
low_sparse_size – integer (= total size for storage of lower diagonal blocks)
upp_sparse_size – integer (= total size for storage of upper diagonal blocks)
profile_size – integer (= total size for storage of half dense diagonal blocks)
maxdim – integer (= maximum dimension of diagonal blocks)
bc_num – integer
p_row – integer array (= pointer of first entry of each row)
p_diag – integer array (= pointer of first entry of each row in diagonal block)
p_upp – integer array (= pointer of first entry of each row in upper block)
p_ext – integer array (= pointer of first entry of each row in external block)
column_num – integer array (= column number of non zero entry in sparse structure)
p_front – integer array (= pointer of fronts)
new2old – integer array (= equation number correspondance)
p_dia_bloc – integer array (= pointer of half dense diagonal blocks)
bc_mask – integer array
coef – complex array (= sparse matrix coefficients)
dia_blo – complex array (= half dense diagonal blocks)

1.6 **feti_data**

NAME

feti_data

SYNOPSIS

MODULE `feti_data`

DESCRIPTION

Data structures for complex FETI solver.

ARGUMENTS

subdom – mesh feti structure
Dirich – nodal list structure
interfine – interface descriptor structure
mesh_interfine – interface mesh structure
transp – sparse matrix structure
feti_rhs – complex array
feti_solution – complex array

MODULES

USE mesh_feti_struct
 USE nodal_list_struct
 USE interf_descriptor_struct
 USE interf_mesh_struct
 USE sparse_matrix_struct

1.7 **feti_param**

NAME

feti_param

SYNOPSIS

MODULE `feti_param`

DESCRIPTION

FETI parameters.

ARGUMENTS

typre – integer
feti_max_numb_it – integer
feti_numb_dir – integer
numb_restart – integer
feti_eps – real
eps_stagn – real

1.8 **interf_descriptor_struct**

NAME

interf_descriptor_struct

SYNOPSIS

MODULE `interf_descriptor_struct`

DESCRIPTION

Interface structure. The inner and outer data structures are for the case of non conforming grids or multi-level interface management. In case of conforming interfaces they are identical to the interface data structure.

ARGUMENTS

subdom_numb – integer (= subdomain number)
numb_subdom – integer (= number of subdomains)
subdom_neq – integer (= number of equations in subdomain)
numb_neighb – integer (= number of neighbouring subdomains)
list_neighb – integer array (= list of neighbouring subdomains)
signe – integer array (= sign of interface)
interf_nodes – integer (= number of interface nodes in subdomain)
p_intf_nodes – integer array (= pointer of interface nodes)
list_intf_nodes – integer array (= list of interface nodes)
interf_neq – integer (= number of interface equations in subdomain)
p_intf_eq – integer array (= pointer of interface equations)
list_intf_eq – integer array (= list of interface equations)
weight – real array (= weighting factor)
weightd – real array (= weighting factor)
inn_interf_neq – integer (= number of inner interface equations)
p_list_inn – integer array (= pointer of inner interface equations)
list_inn – integer array (= list of inner interface equations)
buff_inn – complex array (= sending buffer of values for inner interface equations)
out_interf_neq – integer (= number of outer interface equations)
p_list_out – integer array (= pointer of outer interface equations)
list_out – integer array (= list of outer interface equations)
buff_out – complex array (= sending buffer of values for outer interface equations)
global_numb_elements – integer (= global number of element)
elem2dom – integer array (= global element to subdomain allocation array)
local_numb_elements – integer (= local number of element)
l2g_elem – integer array (= local to global element number correspondance)

1.9 interf_mesh_struct

NAME

interf_mesh_struct

SYNOPSIS

MODULE `interf_mesh_struct`

DESCRIPTION

Interface mesh structure. The inner and outer data structures are for the case of non conforming grids or multi-level interface management. In case of conforming interfaces they are identical to the interface data structure.

ARGUMENTS

numb_nodes – integer (= number of nodes)
numb_neighb – integer (= number of neighbouring subdomains)
list_neighb – integer array (= list of neighbouring subdomains)
numb_elements – integer (= number of elements)
p_elem – integer array (= pointer of interface elements)
type_elem – integer array
elem_region – integer array
elem_signe – integer array
p_node_id – integer array (= pointer of interface nodes)
node_id – integer array (= list of interface nodes)
p_geometry – integer array (= pointer of nodes in each element)
geometry – integer array (= list of nodes in each element)
neq – integer (= number of
equations) – pointer of equations attached to each node + list
eq_id – integer array
eq_neighb – integer array
p_node_eq – integer array
node_eq – integer array
type_dof – integer array (= pointed list of types of dof)
p_topology – integer array (= pointer of interface equations)
topology – integer array (= list of interface equations)

1.10 load_curve_struct

NAME

load_curve_struct

SYNOPSIS

MODULE `load_curve_struct`

DESCRIPTION

Load curve structure.

ARGUMENTS

id_curve – integer
numb_points – integer
var – real array
factor – real array

1.11 **mesh_feti_struct**

NAME

mesh_feti_struct

SYNOPSIS

MODULE `mesh_feti_struct`

DESCRIPTION

Mesh structure.

ARGUMENTS

numb_elements – integer (= number of elements)
numb_nodes – integer (= number of nodes)
neq – integer (= number of equations)
p_node_eq – integer array (= pointer of node to equation correspondance)
node_eq – integer array (= list of equations associated with each node)
type_dof – integer array (= type of equations associated with each node)
l2g_elem – integer array (= local to global element number correspondance)
l2g_node – integer array (= local to global node number correspondance)

1.12 **mesh_struct**

NAME

mesh_struct

SYNOPSIS

MODULE `mesh_struct`

DESCRIPTION

Mesh structure.

ARGUMENTS

numb_connect – number of connected parts
numb_regions – number of elementary regions
numb_acou_regions – number of acoustic regions
numb_admit_regions – number of admittance regions
numb_infinite_regions – number of infinite regions
numb_nodes – number of nodes
space_dim – number of coordinates per node
node_id – external node id
coordinates – coordinates of nodes
tolerance – relative precision tolerance for coordinates
min_nodid – minimum node id
max_nodid – maximum node id
rev_node_id – internal node id
numb_elements – number of elements
numb_hexa – number of 8-node solid elements
numb_tetra – number of 4-node solid elements
numb_admit_quad – number of 4-node admittance elements
numb_admit_quad – number of 3-node admittance elements
numb_infinite_quad – number of 4-node infinite elements
numb_infinite_tria – number of 3-node infinite elements
geometry – pointed list of nodes in each element
p_geometry – pointer to geometry array
rev_geometry – pointed list of elements each node belongs to
p_rev_geometry – pointer to rev_geometry array
elem_id – id of element
elem_region – internal id of region of element
elem_region_id – external id of region of element
neighb_region – internal id of neighboring region
type_elem – type of element
neq – number of equations
node_eq – pointed list of equations attached to each node
type_dof – pointed list of types of dof attached to each node
p_node_eq – pointer to node_eq and type_dof arrays
topology – pointed list of equations in each element
p_topology – pointer of element connectivity per equations
primal_field – acoustic pressure
dual_field – normal velocity

1.13 **model**

NAME

model

SYNOPSIS

```
MODULE model
```

DESCRIPTION

Data structures used for model.

ARGUMENTS

see module

MODULES

```
USE mesh_struct
USE nodal_list_struct
USE facet_struct
USE sparse_matrix_struct
USE load_curve_struct
USE region_property_struct
```

1.14 **mpif**

NAME

mpif

SYNOPSIS

```
SUBROUTINE mpif
```

DESCRIPTION

FETI communicator declaration and mpif as a module

ARGUMENTS

```
max_count – parameter integer
FETI_COM – integer
COUPLE_COM – integer INCLUDES INCLUDE 'mpif.h'
```

1.15 **nodal_list_struct**

NAME

nodal_list_struct

SYNOPSIS

```
MODULE nodal_list_struct
```

DESCRIPTION

Nodal list structure.

ARGUMENTS

neq – integer (= dimension of problem)
numb – integer (= number of entries in the list)
list_id – integer (= external id of the list)
type_list – integer (= type of the list)
node_id – integer array (= list of nodes)
type_dof – integer array (= type of dof for each entry)
value – complex array (= values)
id_curve – integer array (= curve number associated to each entry)
id_mat – integer array (= material number)
eq_id – integer array (= list of listed equations)
mask – integer array (= mask array equal zero if not a listed equation)

1.16 **outfil**

NAME

outfil

SYNOPSIS

MODULE `outfil`

DESCRIPTION

Output unit structure.

ARGUMENTS

P6 – integer (= output unit)
print_matrix – integer
print_rhs – integer
print_solution – integer
print_mesh – integer

1.17 **realloc**

NAME

realloc

SYNOPSIS

MODULE `realloc`

DESCRIPTION

Functions for memory allocation.

ARGUMENTS

SUBROUTINE realloc_i1
 SUBROUTINE extent_i1
 SUBROUTINE realloc_r1
 SUBROUTINE realloc_c1
 SUBROUTINE realloc_i2
 SUBROUTINE realloc_r2
 SUBROUTINE extent_r2
 SUBROUTINE realloc_c2

1.18 **region_property_struct**

NAME

region_property_struct

SYNOPSIS

MODULE region_property_struct

DESCRIPTION

Module of acoustic, admittance and infinite element material.

ARGUMENTS

density – real
id_curve_density – integer
celerity – complex
id_curve_celerity – integer
impedance – complex
id_curve_impedance – integer
origin – real array
axes – real array
adist – real
bdist – real
cdist – real
order – integer
tolerance – real

1.19 **solver_param**

NAME

solver_param

SYNOPSIS

MODULE solver_param

DESCRIPTION

Solver parameters module.

ARGUMENTS

type_galerkin – integer
symmetric_matrix – integer
type_solver – integer
type_precond – integer
max_numb_it – integer
numb_dir – integer
out_of_core – integer
type_opt – integer
nsd – integer
epsilon – real
numb_freq – integer
frequency – real array

1.20 **sparse_matrix_struct**

NAME

sparse_matrix_struct

SYNOPSIS

MODULE `sparse_matrix_struct`

DESCRIPTION

Sparse matrix structure.

ARGUMENTS

neq – integer (= dimension of the matrix)
sparse_size – integer (= total number of non zero entries)
p_row – integer array (= position of first entry of each row)
column_numb – integer array (= column number of each non zero entry)
coef – complex array (= non zero entries)
symmetric – integer (= symmetric matrix flag)

2 Functions

2.1 **add_interf_matrix**

NAME

add_interf_matrix

SYNOPSIS

```
SUBROUTINE add_interf_matrix
```

DESCRIPTION

Add interface sparse matrix to impedance sparse matrix.

ARGUMENTS

None

MODULES

```
USE model  
USE feti_data  
USE solver_param  
USE outfil  
USE direct_unsymmetric
```

2.2 **add_interf_matrix_structure**

NAME

add_interf_matrix_structure

SYNOPSIS

```
SUBROUTINE add_interf_matrix_structure
```

DESCRIPTION

Add interface sparse matrix data structure.

ARGUMENTS

None

MODULES

```
USE model  
USE feti_data
```

2.3 add_sparse_sym_matrix

NAME

add_sparse_sym_matrix

SYNOPSIS

SUBROUTINE `add_sparse_sym_matrix` (`a,b,b2a_eqid`)

DESCRIPTION

Add symmetric interface sparse matrix to impedance matrix.

ARGUMENTS

a – sparse matrix structure
b – sparse matrix structure
b2a_eqid – integer array

MODULES

USE `outfil`
 USE `realloc`
 USE `sparse_matrix_struct`

2.4 add_sparse_sym_struct

NAME

add_sparse_sym_struct

SYNOPSIS

SUBROUTINE `add_sparse_sym_struct` (`a,b,b2a_eqid`)

DESCRIPTION

Add symmetric interface sparse matrix structure.

ARGUMENTS

a – sparse matrix structure
b – sparse matrix structure
b2a_eqid – integer array

MODULES

USE `outfil`
 USE `realloc`
 USE `sparse_matrix_struct`

2.5 add_sparse_unsym_matrix

NAME

add_sparse_unsym_matrix

SYNOPSIS

SUBROUTINE `add_sparse_unsym_matrix` (`a,b,b2a_eqid`)

DESCRIPTION

Add unsymmetric interface sparse matrix to impedance matrix.

ARGUMENTS

a – sparse matrix structure
b – sparse matrix structure
b2a_eqid – integer array

MODULES

USE `outfil`
 USE `realloc`
 USE `sparse_matrix_struct`

2.6 add_sparse_unsym_struct

NAME

add_sparse_unsym_struct

SYNOPSIS

SUBROUTINE `add_sparse_unsym_struct` (`a,b,b2a_eqid`)

DESCRIPTION

Add unsymmetric interface sparse matrix structure.

ARGUMENTS

a – sparse matrix structure
b – sparse matrix structure
b2a_eqid – integer array

MODULES

USE `outfil`
 USE `realloc`
 USE `sparse_matrix_struct`

2.7 assemble_bc

NAME

assemble_bc

SYNOPSIS

```

SUBROUTINE assemble_bc (dom,acou_mat,dim_neumann,Neumann,      &
&                        dim_fneumann,FNeumann,rhs,frequency,dim_lc,&
&                        LC)

```

DESCRIPTION

Assemble right hand side.

ARGUMENTS

dom – mesh structure
acout_mat – region property structure array
dim_neumann – integer
Neumann – nodal list structure array
dim_fneumann – integer
FNeumann – facet structure array
rhs – complex array
frequency – real
dim_lc – integer
LC – load curve structure array

MODULES

```

USE mesh_struct
USE nodal_list_struct
USE facet_struct
USE region_property_struct
USE load_curve_struct
USE outfil

```

2.8 assemble_facet_bc

NAME

assemble_facet_bc

SYNOPSIS

```

SUBROUTINE assemble_facet_bc (dom,acou_mat,FNeumann,rhs,frequency,&
&                             dim_lc,LC)

```

DESCRIPTION

Assemble contribution of facet Neumann boundary conditions to right hand side.

ARGUMENTS

dom – mesh structure
acou_mat – region property structure array
FNeumann – facet structure
rhs – complex array
frequency – real
dim_lc – integer
LC – load curve structure array

MODULES

USE mesh_struct
 USE facet_struct
 USE region_property_struct
 USE load_curve_struct
 USE outfil

2.9 **assemble_impedance**

NAME

assemble_impedance

SYNOPSIS

```

SUBROUTINE assemble_impedance (dom,a,frequency,acou_mat,admi_mat, &
&                               inf_mat,typegalerkin)
  
```

DESCRIPTION

Assemble impedance sparse matrix on domain.

ARGUMENTS

dom – mesh structure
a – sparse matrix structure
frequency – real
acou_mat – region property structure array
admi_mat – region property structure array
inf_mat – region property structure array
typegalerkin – integer

MODULES

USE sparse_matrix_struct
 USE mesh_struct
 USE region_property_struct
 USE outfil

2.10 **assemble_intf_matrix**

NAME

assemble_intf_matrix

SYNOPSIS

```

SUBROUTINE assemble_intf_matrix (dom,mesh_intf,b,frequency,    &
&                               acou_mat,type_sol,subdom_num)

```

DESCRIPTION

Assemble interface sparse matrix

ARGUMENTS

dom – mesh structure
mesh_intf – interface mesh structure
b – sparse matrix structure
frequency – real
acou_mat – region property structure array
type_sol – integer
subdom_num – integer

MODULES

USE sparse_matrix_struct
USE interf_mesh_struct
USE mesh_struct
USE region_property_struct
USE outfil

WARNINGS

The alpha and beta parameters are the parameters used in the continuous relation $(\alpha u + \beta \partial^2 \tau u)$ which gives after discretization $(\alpha M - \beta K)$.

2.11 **assemble_nodal_bc**

NAME

assemble_nodal_bc

SYNOPSIS

```

SUBROUTINE assemble_nodal_bc (dom,acou_mat,Neumann,rhs,frequency,  &
&                             dim_lc,LC)

```

DESCRIPTION

Assemble contribution of nodal Neumann boundary conditions to right hand side.

ARGUMENTS

dom – mesh structure
acou_mat – region property structure array
Neumann – nodal list structure
rhs – complex array
frequency – real
dim_lc – integer
lc – load curve structure array

MODULES

USE mesh_struct
 USE nodal_list_struct
 USE region_property_struct
 USE load_curve_struct
 USE outfil

2.12 **bcast_acoustic_property**

NAME

bcast_acoustic_property

SYNOPSIS

SUBROUTINE `bcast_acoustic_property` (`mat`)

DESCRIPTION

Broadcast acoustic region property to all processes.

ARGUMENTS

mat – region property structure

MODULES

USE region_property_struct
 USE mpif

2.13 **bcast_admittance_property**

NAME

bcast_admittance_property

SYNOPSIS

SUBROUTINE `bcast_admittance_property` (`mat`)

DESCRIPTION

Broadcast admittance region property to all processes.

ARGUMENTS

mat – region property structure

MODULES

USE region_property_struct
USE mpif

2.14 **bcast_facet**

NAME

bcast_facet

SYNOPSIS

SUBROUTINE `bcast_facet (bc,intf)`

DESCRIPTION

Broadcast facet list to all processes and extract.

ARGUMENTS

bc – facet structure
intf – interface descriptor structure

MODULES

USE facet_struct
USE interf_descriptor_struct
USE outfil
USE mpif

2.15 **bcast_infinite_property**

NAME

bcast_infinite_property

SYNOPSIS

SUBROUTINE `bcast_infinite_property (mat)`

DESCRIPTION

Broadcast infinite element region property to all processes.

ARGUMENTS

mat – region property structure

MODULES

```
USE region_property_struct
USE mpif
```

2.16 bcast_load_curve

NAME

bcast_load_curve

SYNOPSIS

```
SUBROUTINE bcast_load_curve (lc)
```

DESCRIPTION

Broadcast load curve to all processes.

ARGUMENTS

lc – load curve structure

MODULES

```
USE load_curve_struct
USE outfl
USE mpif
```

2.17 bcast_model_attribute

NAME

bcast_model_attribute

SYNOPSIS

```
SUBROUTINE bcast_model_attribute (attribute)
```

DESCRIPTION

Broadcast model attribute to all processes.

ARGUMENTS

attribute – character string

MODULES

```
USE mpif
```

2.18 bcast_model_parameters

NAME

bcast_model_parameters

SYNOPSIS

SUBROUTINE `bcast_model_parameters (n1,n2,n3,n4,n5)`

DESCRIPTION

Broadcast model parameters to all processes.

ARGUMENTS

n1 – integer
n2 – integer
n3 – integer
n4 – integer
n5 – integer

MODULES

USE `solver_param`
 USE `outfil`
 USE `mpif`

2.19 bcast_nodal_list

NAME

bcast_nodal_list

SYNOPSIS

SUBROUTINE `bcast_nodal_list (bc,intf,opt)`

DESCRIPTION

Broadcast nodal list to all processes and extract.

ARGUMENTS

bc – nodal list structure
intf – interface descriptor structure
opt – integer

MODULES

USE `nodal_list_struct`
 USE `interf_descriptor_struct`
 USE `outfil`
 USE `mpif`

2.20 bcast_solver_parameters

NAME

bcast_solver_parameters

SYNOPSIS

SUBROUTINE `bcast_solver_parameters`

DESCRIPTION

Broadcast solver parameters to all processes.

ARGUMENTS

None

MODULES

USE `solver_param`
USE `block_size`
USE `outfil`
USE `mpif`**2.21 build_diag**

NAME

build_diag

SYNOPSIS

SUBROUTINE `build_diag` (`neq`,`dim`,`p_row`,`column_numb`,`coef`,`diag`)

DESCRIPTION

Define diagonal coefficients of sparse matrix

ARGUMENTS

neq – integer
dim – integer
p_row – integer array
column_numb – integer array
coef – complex array
diag – complex array

MODULES

USE `outfil`

2.22 build_dirich_aug

NAME

build_dirich_aug

SYNOPSIS

```

SUBROUTINE build_dirich_aug (intf_mesh,Dirichlet,Dirich_aug,
&                             listintf)

```

DESCRIPTION

Compute Schur complement matrix.

ARGUMENTS

intf_mesh – interface mesh structure
Dirichlet – factorized matrix structure
Dirich_aug – factorized matrix structure
listintf – nodal list structure

MODULES

```

USE solver_param
USE outfil
USE interf_mesh_struct
USE nodal_list_struct
USE realloc
USE factorized_matrix_struct

```

2.23 build_interf_matrix_structure

NAME

build_interf_matrix_structure

SYNOPSIS

```

SUBROUTINE build_interf_matrix_structure

```

DESCRIPTION

Build interface sparse matrix data structure.

ARGUMENTS

None

MODULES

```
USE model
USE feti_data
USE solver_param
USE outfil
```

2.24 build_interface

NAME

build_interface

SYNOPSIS

```
SUBROUTINE build_interface
```

DESCRIPTION

Build subdomain interface.

ARGUMENTS

None

MODULES

```
USE outfil
USE mpif
USE feti_data
USE model
USE feti_param
```

2.25 build_matrix_structure

NAME

build_matrix_structure

SYNOPSIS

```
SUBROUTINE build_matrix_structure
```

DESCRIPTION

Build sparse matrix data structure.

ARGUMENTS

None

MODULES

USE model
USE solver_param

2.26 build_node_eq

NAME

build_node_eq

SYNOPSIS

SUBROUTINE build_node_eq (dom,inf_mat)

DESCRIPTION

Build correspondance between nodes and equations.

ARGUMENTS

dom – mesh structure
intf_mat – region property structure array

MODULES

USE mesh_struct
USE region_property_struct
USE realloc
USE outfil

2.27 build_rev_geometry

NAME

build_rev_geometry

SYNOPSIS

SUBROUTINE build_rev_geometry (dom)

DESCRIPTION

Build pointed list of elements each node belongs to.

ARGUMENTS

dom – mesh structure

MODULES

USE outfil
USE realloc
USE mesh_struct

2.28 build_rev_nodid

NAME

build_rev_nodid

SYNOPSIS

SUBROUTINE build_rev_nodid (dom)

DESCRIPTION

Build external to internal node number correspondance and change list of nodes in each element accordingly.

ARGUMENTS

dom – mesh structure

MODULES

USE outfil
USE mesh_struct**2.29 build_sparse_sym_struct**

NAME

build_sparse_sym_struct

SYNOPSIS

SUBROUTINE build_sparse_sym_struct (dom,a)

DESCRIPTION

Build sparse matrix structure.

ARGUMENTS

dom – mesh structure
a – sparse matrix structure

MODULES

USE outfil
USE realloc
USE mesh_struct
USE sparse_matrix_struct

2.30 build_sparse_unsym_struct

NAME

build_sparse_unsym_struct

SYNOPSIS

SUBROUTINE `build_sparse_unsym_struct (dom,a)`

DESCRIPTION

Build sparse matrix structure.

ARGUMENTS

dom – mesh structure
a – sparse matrix structure

MODULES

USE `outfil`
USE `realloc`
USE `mesh_struct`
USE `sparse_matrix_struct`**2.31 build_topology**

NAME

build_topology

SYNOPSIS

SUBROUTINE `build_topology (dom)`

DESCRIPTION

Build list of equations in each element, i.e. .build topology of elements using list of nodes per element and list and type of equations per node.

ARGUMENTS

dom – mesh structure

MODULES

USE `mesh_struct`
USE `outfil`
USE `realloc`

2.32 calpd

NAME

calpd

SYNOPSIS

SUBROUTINE calpd (pl,pc,pd,n,nza)

DESCRIPTION

Determine the cpd pointer which point to the adress inside a of the diagonal coefficient of the matrix. The matrix has a general structure.

ARGUMENTS

pl – integer array
pc – integer array
pd – integer array
n – integer
nza – integer

2.33 cmk_number

NAME

cmk_number

SYNOPSIS

SUBROUTINE cmk_number (nodes,p_graph,graph,list)

DESCRIPTION

Renumber graph via reverse Cuthill-Mac Kee algorithm.

ARGUMENTS

nodes – integer
p_graph – integer array
graph – integer array
list – integer array

MODULES

USE outfil

2.34 condens_1

NAME

condens_1

SYNOPSIS

```
SUBROUTINE condens_1 (lda,dim1,dim2,a,d,e,f,ie0,if0,nop)
```

DESCRIPTION

Compute the Schur complement matrix $[D^{-1}U^{-t}F]^t$ with LDU the Gauss-Jordan factorisation of the matrix A .

ARGUMENTS

lda – integer
dim1 – integer
dim2 – integer
a – complex array
d – complex array
e – complex array
f – complex array
ie0 – integer array
if0 – integer array
nop – real

MODULES

USE block_size

2.35 **condens_2**

NAME

condens_2

SYNOPSIS

```
SUBROUTINE condens_2 (lda,dim1,dim2,a,e,ie0,nop)
```

DESCRIPTION

Compute $[L^{-1}E]$ matrix where LDU is the Gauss-Jordan factorisation of the matrix A .

ARGUMENTS

lda – integer
dim1 – integer
dim2 – integer
a – complex array
e – complex array
ie0 – integer array
nop – real

MODULES

USE block_size

2.36 **condens_3**

NAME

condens_3

SYNOPSIS

SUBROUTINE `condens_3` (`lda,dim1,dim2,e,f,c,ie0,if0,nop`)

DESCRIPTION

Compute the Schur complement matrix $C = C - [FU^{-1}d^{-1}][L^{-1}E]$ where $[L^{-1}E]$ and $[FU^{-1}d^{-1}]$ have already been computed.

ARGUMENTS

lda – integer
dim1 – integer
dim2 – integer
e – complex array
f – complex array
c – complex array
ie0 – integer array
if0 – integer array
nop – real

MODULES

USE `block_size`**2.37** **copy_block**

NAME

copy_block

SYNOPSIS

SUBROUTINE `copy_block` (`lda,dim1,dim2,a,b`)

DESCRIPTION

Copy one block of a matrix.

ARGUMENTS

lda – integer
dim1 – integer
dim2 – integer
a – complex array
b – complex array

MODULES

USE outfil

2.38 **dirichlet_bc**

NAME

dirichlet_bc

SYNOPSIS

SUBROUTINE dirichlet_bc (frequency,dim_lc,lc,bc)

DESCRIPTION

Build Interpolate Dirichlet boundary condition.

ARGUMENTS

frequency – real
dim_lc – integer
lc – load curve structure array
bc – nodal list structure

MODULES

USE load_curve_struct
 USE nodal_list_struct
 USE outfil

2.39 **distribute_model**

NAME

distribute_model

SYNOPSIS

SUBROUTINE distribute_model

DESCRIPTION

Distribute domain decomposed model.

ARGUMENTS

None

MODULES

USE model
 USE feti_data
 USE outfil

2.40 **echo_facet**

NAME

echo_facet

SYNOPSIS

SUBROUTINE `echo_facet` (`P6`,`bc`)

DESCRIPTION

Print facets data structure.

ARGUMENTS

P6 – integer (= output unit)**bc** – facet structure

MODULES

USE `facet_struct`**2.41** **echo_mesh**

NAME

echo_mesh

SYNOPSIS

SUBROUTINE `echo_mesh` (`P6`,`dom`,`subdom_numb`)

DESCRIPTION

Print mesh data structure.

ARGUMENTS

P6 – integer (output unit)**dom** – mesh structure**subdom_numb** – integer

MODULES

USE `mesh_struct`**2.42** **echo_mesh_femview**

NAME

echo_mesh_femview

SYNOPSIS

```
SUBROUTINE echo_mesh_femview (dom,subdom_num,numb_subdom,elem2dom,&
& title,directory)
```

DESCRIPTION

Write mesh geometrical data in a FEMVIEW file.

ARGUMENTS

dom – mesh structure
subdom_num – integer
numb_subdom – integer
elem2dom – integer array
title – character string
directory – character string

MODULES

USE mesh_struct

2.43 **echo_model**

NAME

echo_model

SYNOPSIS

```
SUBROUTINE echo_model (subdom_num,numb_subdom)
```

DESCRIPTION

Print all model data structures.

ARGUMENTS

subdom_num – integer
numb_subdom – integer

MODULES

USE model
USE outfil

2.44 **echo_nodal_list**

NAME

echo_nodal_list

SYNOPSIS

```
SUBROUTINE echo_nodal_list (P6,bc)
```

DESCRIPTION

Print nodal list structure.

ARGUMENTS

P6 – integer (= output unit)
bc – nodal list structure

MODULES

```
USE nodal_list_struct
```

2.45 **equation_setup**

NAME

equation_setup

SYNOPSIS

```
SUBROUTINE equation_setup
```

DESCRIPTION

Setup system of equations associated with model.

ARGUMENTS

None

MODULES

```
USE model
USE solver_param
USE outfil
```

2.46 **extent_i1**

NAME

extent_i1

SYNOPSIS

```
SUBROUTINE extent_i1 (tab,inc_dim1,ierr)
```

DESCRIPTION

Extension of multi-dimensional integer pointer.

ARGUMENTS

tab – integer array
inc_dim1 – integer
ierr – integer

2.47 **extent_r2**

NAME

extent_r2

SYNOPSIS

SUBROUTINE `extent_r2` (`tab,dim1,inc_dim2,ierr`)

DESCRIPTION

Extension of multi-dimensional real pointer.

ARGUMENTS

tab – real array
dim1 – integer
inc_dim2 – integer
ierr – integer

2.48 **extract_submesh**

NAME

extract_submesh

SYNOPSIS

SUBROUTINE `extract_submesh` (`dom,intf,target_subdom`)

DESCRIPTION

Extract subdomain submesh and send it to associated process. Build topology of elements using list of nodes per element and list and type of equations per node.

ARGUMENTS

dom – mesh structure
intf – interface descriptor structure
target_subdom – integer

MODULES

```

USE mesh_struct
USE interf_descriptor_struct
USE outfil
USE mpif
USE realloc

```

2.49 facet_rhs

NAME

facet_rhs

SYNOPSIS

```

SUBROUTINE facet_rhs (dom,bc)

```

DESCRIPTION

Compute specific values for non homogeneous Robin conditions. $P1$ discretization is used at the nodes of facets.

ARGUMENTS

dom – mesh structure
bc – facet structure

MODULES

```

USE mesh_struct
USE facet_struct
USE outfil

```

2.50 factorize_numeric

NAME

factorize_numeric

SYNOPSIS

```

SUBROUTINE factorize_numeric

```

DESCRIPTION

Numerical factorization of local matrices.

ARGUMENTS

None

MODULES

USE outfl
USE solver_param

2.51 factorize_symbolic

NAME

factorize_symbolic

SYNOPSIS

SUBROUTINE factorize_symbolic

DESCRIPTION

Symbolic factorization of local matrices.

ARGUMENTS

None

MODULES

USE outfl
USE solver_param

2.52 feti_abort

NAME

feti_abort

SYNOPSIS

SUBROUTINE feti_abort

DESCRIPTION

Abort FETI.

ARGUMENTS

None

MODULES

USE mpif
USE outfl

2.53 feti_finalize

NAME

feti_finalize

SYNOPSIS

SUBROUTINE `feti_finalize`

DESCRIPTION

Close FETI communication space and output file.

ARGUMENTS

None

MODULES

USE `mpif`
USE `outfil`**2.54 feti_init**

NAME

feti_init

SYNOPSIS

SUBROUTINE `feti_init` (`subdom_num`,`num_subdom`)

DESCRIPTION

Create FETI communication space, get subdomain number and number of subdomains and open output file.

ARGUMENTS

subdom_num – integer
num_subdom – integer

MODULES

USE `model`
USE `outfil`
USE `mpif`
USE `feti_data`
USE `feti_param`

2.55 **fill_sym_sparse**

NAME

fill_sym_sparse

SYNOPSIS

```

SUBROUTINE fill_sym_sparse (neq,sparse_size,rowp,col,inicoef,      &
&                          new2old,p_row,col_num,coef)

```

DESCRIPTION

Filling of renumbered lower triangular part of the symmetric sparse structure stored row by row from the upper triangular symmetric one.

ARGUMENTS

rowp – pointer of rows of the upper triangular part of the initial matrix
col – list of column indices of rows of the initial matrix
inicoef – list of entries of the initial matrix
new2old – new to old correspondance
p_row – pointer of rows of the lower triangular part of the symmetric matrix
col_num – list of column indices of rows of the renumbered matrix
coef – list of entries of the renumbered matrix

MODULES

USE outfil

2.56 **fill_unsym_sparse**

NAME

fill_unsym_sparse

SYNOPSIS

```

SUBROUTINE fill_unsym_sparse(neq,sparse_size,rowp,col,inicoef,    &
&                          new2old,p_row,col_num,coef)

```

DESCRIPTION

Filling of renumbered unsymmetric sparse structure stored by row

ARGUMENTS

rowp – pointer of rows of the initial matrix
col – list of column indices of rows of the initial matrix
inicoef – list of entries of the initial matrix
new2old – new to old correspondance
p_row – pointer of rows of the renumbered matrix
col_num – list of column indices of rows of the renumbered matrix
coef – list of entries of the renumbered matrix

MODULES

USE outfil

2.57 **frontal_num**

NAME

frontal_num

SYNOPSIS

```
SUBROUTINE frontal_num (nodes,p_graph,graph,i0,list,profile,indic,&
& connect)
```

DESCRIPTION

Frontal renumbering of graph.

ARGUMENTS

nodes – integer
p_graph – integer array
graph – integer array
i0 – integer
list – integer array
profile – real
indic – integer array
connect – integer array

2.58 **full_schur_b**

NAME

full_schur_b

SYNOPSIS

```
SUBROUTINE full_schur_b (lda,nrow,ncol,u,l,s,n2)
```

DESCRIPTION

Compute Schur complement: $S = A - LU$ using a block algorithm.

ARGUMENTS

lda – integer (= dimension of matrix A)
nrow – integer
ncol – integer
u(nrow,ncol) – complex array (= the upper band)
l(ncol,nrow) – complex array (= the lower band)
s(ncol,ncol) – complex array (= the Schur complement)
n2 – integer

2.59 **full_blo**

NAME

full_blo

SYNOPSIS

SUBROUTINE `full_blo` (`lda,n,a,full_a`)

DESCRIPTION

Store a dense diagonal block.

ARGUMENTS

lda – integer
n – integer
a – complex array
full_a – complex array

2.60 **full_fwbw**

NAME

full_fwbw

SYNOPSIS

SUBROUTINE `full_fwbw` (`n,a,y,x`)

DESCRIPTION

Compute dense matrix forward-backward substitution for a dense unsymmetric complex matrix.

ARGUMENTS

n – integer
a – complex array
y – complex array
x – complex array

2.61 full_ldlt

NAME

full_ldlt

SYNOPSIS

SUBROUTINE `full_ldlt` (`lda,n,a,d`)

DESCRIPTION

Compute symmetric Gauss Jordan factorization of dense matrix. The lower triangular part only is computed.

ARGUMENTS

lda – integer
n – integer
a – complex array
d – complex array

MODULES

USE `block_size`**2.62 full_ldlt_b**

NAME

full_ldlt_b

SYNOPSIS

SUBROUTINE `full_ldlt_b` (`lda,n,a,d`)

DESCRIPTION

Compute symmetric Gauss Jordan factorization of symmetric dense matrix.

ARGUMENTS

lda – integer
n – integer
a – complex array
d – complex array

2.63 full_ldu

NAME

full_ldu

SYNOPSIS

```
SUBROUTINE full_ldu (lda,n,a,d)
```

DESCRIPTION

Compute Gauss-Jordan factorization of dense matrix.

ARGUMENTS

lda – integer
n – integer
a – complex array
d – complex array

MODULES

```
USE block_size
```

2.64 **full_ldu_b**

NAME

full_ldu_b

SYNOPSIS

```
SUBROUTINE full_ldu_b (lda,n,a,d)
```

DESCRIPTION

Compute Gauss-Jordan factorization of dense matrix.

ARGUMENTS

lda – integer
n – integer
a – complex array
d – complex array

MODULES

```
USE block_size
```

2.65 **full_schur_b_sparse**

NAME

full_schur_b_sparse

SYNOPSIS

```
SUBROUTINE full_schur_b_sparse (nrow,ncol,iu0,u,jl0,l,s,lda,n1,n2)
```

DESCRIPTION

Compute Schur complement: $S = A - LU$ using a block algorithm.

ARGUMENTS

nrow – integer
ncol – integer
iu0 – integer
u – complex array
jl0 – integer array
l – complex array
s – complex array
lda – integer
n1 – integer
n2 – integer

2.66 **full_sto**

NAME

full_sto

SYNOPSIS

SUBROUTINE full_sto (lda,n,mat,nblo)

DESCRIPTION

Store a full block.

ARGUMENTS

lda – integer
n – integer
mat – complex array
nblo – integer

2.67 **full_unsto**

NAME

full_unsto

SYNOPSIS

SUBROUTINE full_unsto (n,mat,nblo)

DESCRIPTION

Un-store a dense block.

ARGUMENTS

n – integer
mat – complex array
nblo – integer

2.68 **galerkin_error**

NAME

galerkin_error

SYNOPSIS

SUBROUTINE galerkin_error (subdom_num, frequency)

DESCRIPTION

Compute the finite element error analysis.

ARGUMENTS

subdom_num – integer
frequency – real

MODULES

USE model
 USE outfil
 USE realloc
 USE feti_data
 USE feti_param
 USE mpif

2.69 **galerkin_meshsize**

NAME

galerkin_meshsize

SYNOPSIS

SUBROUTINE galerkin_meshsize (dom, x, h)

DESCRIPTION

Compute the Galerkin meshsize-value.

ARGUMENTS

dom – mesh structure
x – real (= hexaedrom nodes coordinates)
h – real (= mesh size)

MODULES

```
USE mesh_struct
USE outfil
```

2.70 galerkin_tau

NAME

galerkin_tau

SYNOPSIS

```
SUBROUTINE galerkin_tau (k,h,tau)
```

DESCRIPTION

Compute Galerkin tau-value.

ARGUMENTS

k – real (= wavenumber of the Helmholtz equation)
h – real (= mesh size)
tau – real (= tauk2-GLS or tauk4-GGLS coefficient)

MODULES

```
USE outfil
```

2.71 gather_nodid

NAME

gather_nodid

SYNOPSIS

```
SUBROUTINE gather_nodid (numb_nodes,l2g_node,node_id,           &
&                        glob_numb_nodes,glob_node_id)
```

DESCRIPTION

Gather subdomain node ids.

ARGUMENTS

numb_nodes – integer
l2g_node – integer array
node_id – integer array
glob_numb_nodes – integer
glob_node_id – integer array

2.72 gather_sol

NAME

gather_sol

SYNOPSIS

```

SUBROUTINE gather_sol (numb_nodes,l2g_node,p_node_eq,type_dof,    &
&                      node_eq,min_dof,max_dof,glob_numb_nodes,dof,&
&                      neq,v,glob_neq,glob_v)

```

DESCRIPTION

Gather subdomain solutions.

ARGUMENTS

```

numb_nodes – integer
l2g_node – integer array
p_node_eq – integer array
type_dof – integer array
node_eq – integer array
min_dof – integer
max_dof – integer
glob_numb_nodes – integer
dof – integer array
neq – integer
v – complex array
glob_neq – integer
glob_v – complex array

```

2.73 global_eq_mask

NAME

global_eq_mask

SYNOPSIS

```

SUBROUTINE global_eq_mask(numb_nodes,l2g_node,p_node_eq,type_dof, &
&                          min_dof,max_dof,glob_numb_nodes,dof)

```

DESCRIPTION

Build mask of equations in subdomain.

ARGUMENTS

numb_nodes – integer
l2g_node – integer array
p_node_eq – integer array
type_dof – integer array
min_dof – integer
max_dof – integer
glob_numb_nodes – integer
dof – integer array

2.74 **global_solution**

NAME

global_solution

SYNOPSIS

SUBROUTINE `global_solution` (`intf,dom,glob_dom,v,glob_v`)

DESCRIPTION

Gather global mesh and solution field.

ARGUMENTS

intf – interface descriptor structure
dom – mesh structure
glob_dom – mesh structure
v – complex array
glob_v – complex array

MODULES

USE `mesh_struct`
 USE `interf_descriptor_struct`
 USE `outfil`
 USE `mpif`
 USE `realloc`

2.75 **globaldotproduct**

NAME

globaldotproduct

SYNOPSIS

SUBROUTINE `globaldotproduct` (`uv,u,v,intf`)

DESCRIPTION

Global dot-product.

ARGUMENTS

uv – complex
u – complex array
v – complex array
intf – interface descriptor structure

MODULES

USE mpif
 USE interf_descriptor_struct
 USE outfil

2.76 **globalsum**

NAME

globalsum

SYNOPSIS

SUBROUTINE `globalsum (buf,n)`

DESCRIPTION

Sum the vector `buf(n)` across the network.

ARGUMENTS

buff – real array
n – integer

MODULES

USE mpif

2.77 **globalsumc**

NAME

globalsumc

SYNOPSIS

SUBROUTINE `globalsumc (buf,n)`

DESCRIPTION

Sum the vector `buf(n)` across the network.

ARGUMENTS

buf – complex array
n – integer

MODULES

USE mpif
 USE outfil

2.78 **globalsumi**

NAME

globalsumi

SYNOPSIS

SUBROUTINE `globalsumi (buf,n)`

DESCRIPTION

Sum the vector `buf(n)` across the network.

ARGUMENTS

buf – integer array
n – integer

MODULES

USE mpif

2.79 **half_schur_b**

NAME

half_schur_b

SYNOPSIS

SUBROUTINE `half_schur_b (lda,nrow,ncol,u,l,s,n2)`

DESCRIPTION

Compute upper triangular part of Schur complement matrix $S = A - LU$ using a block algorithm.

ARGUMENTS

lda – integer
nrow – integer
ncol – integer
u – complex array
l – complex array
s – complex array
n2 – integer

2.80 **half_blo**

NAME

half_blo

SYNOPSIS

SUBROUTINE `half_blo` (`lda,n,a,half_a`)

DESCRIPTION

Store the lower triangular part of matrix A row by row.

ARGUMENTS

lda – integer
n – integer
a – complex array
half_a – complex array

2.81 **half_fwbw**

NAME

half_fwbw

SYNOPSIS

SUBROUTINE `half_fwbw` (`n,a,y,x`)

DESCRIPTION

Compute dense matrix forward-backward substitution in the case of only the upper triangular part of symmetric matrix A is stored column by column.

ARGUMENTS

n – integer
a – complex array
y – complex array
x – complex array

2.82 half_mm_b

NAME

half_mm_b

SYNOPSIS

```
SUBROUTINE half_mm_b (ncol,nrow,l,ldl,u,ldu,s,lds)
```

DESCRIPTION

Compute upper triangular part of Schur complement matrix $S = A - LU$.

ARGUMENTS

ncol – integer
nrow – integer
l(ncol,nrow) – complex array (= the lower band)
ldl – integer
u(nrow,ncol) – complex array (= the upper band)
ldu – integer
s(ncol,ncol) – complex array (= the Schur complement)
lds – integer (= the dimension of matrix A)

2.83 half_schur_b_sparse

NAME

half_schur_b_sparse

SYNOPSIS

```
SUBROUTINE half_schur_b_sparse (nrow,ncol,i0,u,l,s,lda,n1,n2)
```

DESCRIPTION

Compute upper triangular part of Schur complement matrix $S = A - LU$ using a block algorithm.

ARGUMENTS

nrow – integer
ncol – integer
i0(ncol) – integer array (= the position of the last non zero entry in columns of u and rows of l)
u(nrow,ncol) – complex array (= the upper band)
l(ncol,nrow) – complex array (= the lower band)
s(lda,ncol) – complex array (= the Schur complement)
lda – integer
n1 – integer
n2 – integer

2.84 **half_sto**

NAME

half_sto

SYNOPSIS

SUBROUTINE `half_sto` (`lda,n,mat,nblo`)

DESCRIPTION

Store the lower triangular part of matrix *A* row by row.

ARGUMENTS

lda – integer
n – integer
mat – complex array
nblo – integer

2.85 **half_unsto**

NAME

half_unsto

SYNOPSIS

SUBROUTINE `half_unsto` (`n, half_mat, nblo`)

DESCRIPTION

Un-store the upper triangular part of matrix *A* column by column.

ARGUMENTS

n – integer
half_mat – complex array
nblo – integer

2.86 **ilu_0**

NAME

ilu_0

SYNOPSIS

SUBROUTINE `ilu_0` (`neq, sparse_size, p_row, column_numb, coef, lu, p_diag`)

DESCRIPTION

Preconditioner of a general matrix with `ilu(0)`.

ARGUMENTS

neq – integer
sparse_size – integer
p_row – integer array
column_num – integer array
coef – complex array
lu – sparse matrix structure
p_diag – integer array

MODULES

USE `sparse_matrix_struct`
 USE `outfil`

2.87 **ilu_preconditioner**

NAME

ilu_preconditioner

SYNOPSIS

SUBROUTINE `ilu_preconditioner` (`coef,pl,pc,pd,n,nza,ierr`)

DESCRIPTION

`ilu(0)` preconditioner.

ARGUMENTS

coef – complex array
pl – integer array
pc – integer array
pd – integer array
n – integer
nza – integer
ierr – integer

2.88 **imprim**

NAME

imprim

SYNOPSIS

SUBROUTINE `imprim` (`frequency,subdom_num,numb_subdom`)

DESCRIPTION

Print the matrix of domain.

ARGUMENTS

frequency – real
subdom_num – integer
numb_subdom – integer

MODULES

USE model
 USE outfil

2.89 **imprim2**

NAME

imprim2

SYNOPSIS

SUBROUTINE `imprim2` (`frequency`,`subdom_num`,`numb_subdom`)

DESCRIPTION

Print the matrix of interface.

ARGUMENTS

frequency – real
subdom_num – integer
numb_subdom – integer

MODULES

USE model
 USE feti_data
 USE outfil

2.90 **init_model**

NAME

init_model

SYNOPSIS

SUBROUTINE `init_model` (`subdom_num`)

DESCRIPTION

Initialize domain data.

ARGUMENTS

subdom_num – integer

MODULES

USE model

2.91 initialize

NAME

initialize

SYNOPSIS

SUBROUTINE initialize

DESCRIPTION

Initialize all model nodal data structures.

ARGUMENTS

None

MODULES

USE model
USE solver_param
USE outfl

2.92 inivec

NAME

inivec

SYNOPSIS

SUBROUTINE inivec (lx,x)

DESCRIPTION

Pseudo random initialization.

ARGUMENTS

lx – integer
x – complex array

2.93 **input_data**

NAME

input_data

SYNOPSIS

SUBROUTINE `input_data`

DESCRIPTION

Input domain data.

ARGUMENTS

None

MODULES

USE `model`
USE `solver_param`
USE `block_size`
USE `outfil`**2.94** **interf_full_sym_struct**

NAME

interf_full_sym_struct

SYNOPSIS

SUBROUTINE `interf_full_sym_struct` (`intf_mesh,a`)

DESCRIPTION

Build symmetric interface full matrix structure.

ARGUMENTS

intf_mesh – interface mesh structure
a – sparse matrix structure

MODULES

USE `outfil`
USE `realloc`
USE `interf_mesh_struct`
USE `sparse_matrix_struct`

2.95 interf_full_unsym_struct

NAME

interf_full_unsym_struct

SYNOPSIS

SUBROUTINE `interf_full_unsym_struct` (`intf_mesh`,`a`)

DESCRIPTION

Build symmetric interface full matrix structure.

ARGUMENTS

intf_mesh – interface mesh structure
a – sparse matrix structure

MODULES

USE `outfil`
USE `realloc`
USE `interf_mesh_struct`
USE `sparse_matrix_struct`**2.96 interf_sparse_sym_struct**

NAME

interf_sparse_sym_struct

SYNOPSIS

SUBROUTINE `interf_sparse_sym_struct` (`intf_mesh`,`a`)

DESCRIPTION

Build symmetric interface sparse matrix structure.

ARGUMENTS

intf_mesh – interface mesh structure
a – sparse matrix structure

MODULES

USE `outfil`
USE `realloc`
USE `interf_mesh_struct`
USE `sparse_matrix_struct`

2.97 interf_sparse_unsym_struct

NAME

interf_sparse_unsym_struct

SYNOPSIS

SUBROUTINE `interf_sparse_unsym_struct (intf_mesh,a)`

DESCRIPTION

Build unsymmetric interface sparse matrix structure.

ARGUMENTS

intf_mesh – interface mesh structure
a – sparse matrix structure

MODULES

USE `outfil`
USE `realloc`
USE `interf_mesh_struct`
USE `sparse_matrix_struct`**2.98 interface_assemb**

NAME

interface_assemb

SYNOPSIS

SUBROUTINE `interface_assemb (intf,unassemb,assemb)`

DESCRIPTION

Assemble a field along the interface.

ARGUMENTS

intf – interface descriptor structure
unassemb – complex array
assemb – complex array

MODULES

USE `interf_descriptor_struct`
USE `outfil`

2.99 interface_average

NAME

interface_average

SYNOPSIS

SUBROUTINE `interface_average` (`intf`,`unasemb`,`asemb`)

DESCRIPTION

Average a field along the interface

ARGUMENTS

intf – interface descriptor structure
unasemb – complex array (= unassembled field as input)
asemb – complex array (= averaged field along the interface as output)

MODULES

USE `interf_descriptor_struct`**2.100 interface_equation**

NAME

interface_equation

SYNOPSIS

SUBROUTINE `interface_equation` (`dom`,`intf`)

DESCRIPTION

Detect interface equations.

ARGUMENTS

dom – mesh structure
intf – interface descriptor structure

MODULES

USE `outfl`
 USE `mesh_struct`
 USE `interf_descriptor_struct`
 USE `mpif`
 USE `realloc`

2.101 interface_exchange

NAME

interface_exchange

SYNOPSIS

SUBROUTINE `interface_exchange` (`intf`)

DESCRIPTION

Exchange interface data with neighbouring subdomains.

ARGUMENTS

intf – interface descriptor structure

MODULES

USE `mpif`
USE `interf_descriptor_struct`
USE `outfil`**2.102 interface_init**

NAME

interface_init

SYNOPSIS

SUBROUTINE `interface_init` (`intf`)

DESCRIPTION

Initialize interface structure.

ARGUMENTS

intf – interface descriptor structure

MODULES

USE `interf_descriptor_struct`
USE `realloc`
USE `outfil`**2.103 interface_jump**

NAME

interface_jump

SYNOPSIS

```
SUBROUTINE interface_jump (intf,v,w)
```

DESCRIPTION

Compute the jump of a field along the interface.

ARGUMENTS

intf – interface descriptor structure
v – complex array (= local field as input)
w – complex array (= jump of v as output)

MODULES

USE interf_descriptor_struct

2.104 **interface_jump2**

NAME

interface_jump2

SYNOPSIS

```
SUBROUTINE interface_jump2 (intf,k,v,lambda,g)
```

DESCRIPTION

Compute interface gradient.

ARGUMENTS

intf – interface descriptor structure
k – sparse matrix structure
v – complex array
lambda – complex array
g – complex array

MODULES

USE interf_descriptor_struct
 USE sparse_matrix_struct
 USE outfil

SEE ALSO

interface_jump.f90

2.105 interface_mesh_equation

NAME

interface_mesh_equation

SYNOPSIS

SUBROUTINE `interface_mesh_equation` (`dom,intf,mesh_intf`)

DESCRIPTION

Build interface equation per node.

ARGUMENTS

dom – mesh structure
intf – interface descriptor structure
mesh_intf – interface mesh structure

MODULES

USE `outfil`
 USE `interf_descriptor_struct`
 USE `interf_mesh_struct`
 USE `facet_struct`
 USE `mesh_struct`
 USE `mpif`
 USE `realloc`

2.106 interface_mesh_geometry

NAME

interface_mesh_geometry

SYNOPSIS

SUBROUTINE `interface_mesh_geometry` (`dom,intf,mesh_intf`)

DESCRIPTION

Detect interface facets.

ARGUMENTS

dom – mesh structure
intf – interface descriptor structure
mesh_intf – interface mesh structure

MODULES

```
USE outfil
USE interf_descriptor_struct
USE facet_struct
USE mesh_struct
USE interf_mesh_struct
USE mpif
USE realloc
```

2.107 interface_mesh_topology

NAME

interface_mesh_topology

SYNOPSIS

```
SUBROUTINE interface_mesh_topology (mesh_intf)
```

DESCRIPTION

Build list of equations in each element.

ARGUMENTS

mesh_intf – interface mesh structure

MODULES

```
USE interf_mesh_struct
USE outfil
USE realloc
```

2.108 interface_node

NAME

interface_node

SYNOPSIS

```
SUBROUTINE interface_node (intf)
```

DESCRIPTION

Detect interface nodes.

ARGUMENTS

intf – interface descriptor structure

MODULES

```

USE outfil
USE interf_descriptor_struct
USE mpif
USE realloc

```

2.109 interface_signe

NAME

interface_signe

SYNOPSIS

```

SUBROUTINE interface_signe (intf)

```

DESCRIPTION

Sign sub-domains and interfaces.

ARGUMENTS

intf – interface descriptor structure

MODULES

```

USE interf_descriptor_struct
USE outfil
USE mpif

```

2.110 invD_times_U

NAME

invD_times_U

SYNOPSIS

```

SUBROUTINE invD_times_U (lda,nrow,d,ncol,u)

```

DESCRIPTION

Compute $[D]^{-1}U$.

ARGUMENTS

lda – integer
nrow – integer
d – complex array
ncol – integer
u – complex array

2.111 invL_times_U

NAME

invL_times_U

SYNOPSIS

SUBROUTINE `invL_times_U (lda,nrow,ldlt,ncol,u)`

DESCRIPTION

Compute $[L]^{-1}U$.

ARGUMENTS

lda – integer
nrow – integer
ldlt – complex array
ncol – integer
u – complex array

2.112 invL_times_sparsU

NAME

invL_times_sparsU

SYNOPSIS

SUBROUTINE `invL_times_sparsU (n,ldlt,lda,ncol_u,i0,u,n1,n2)`

DESCRIPTION

Compute $[L]^{-1}U$

ARGUMENTS

n – integer
ldlt – complex array
lda – integer
ncol_u – integer
i0 – integer array
u – complex array
n1 – integer
n2 – integer

2.113 invUt_times_sparsU

NAME

invUt_times_sparsU

SYNOPSIS

```
SUBROUTINE invUt_times_sparsU (n,ldu,lda,ncol_u,i0,u,n1,n2)
```

DESCRIPTION

Compute $[U]^{-t}U$.

ARGUMENTS

n – integer
ldu – complex array
lda – integer
ncol_u – integer
i0 – integer array
u – complex array
n1 – integer
n2 – integer

2.114 **jacobi_convergence**

NAME

jacobi_convergence

SYNOPSIS

```
SUBROUTINE jacobi_convergence (omega,omega_moins,omega_plus,k_max,&
& p1,q1,p2,q2,rho_max)
```

DESCRIPTION

Compute maximum value of convergence rate of jacobi algorithm.

ARGUMENTS

omega – real
omega_moins – real
omega_plus – real
k_max – real
p1 – real
q1 – real
p2 – real
q2 – real
rho_max – real

MODULES

USE outfil

2.115 jacobi_optimized

NAME

jacobi_optimized

SYNOPSIS

```

SUBROUTINE jacobi_optimized (type_optimized,omega,h,L,      &
&                          alpha_left,alpha_right,      &
&                          beta_left,beta_right)

```

DESCRIPTION

Compute optimized coefficient of jacobi algorithm.

ARGUMENTS

type_optimized – integer
omega – real
h – real
L – real
alpha_left – complex
alpha_right – complex
beta_left – complex
beta_right – complex

MODULES

```

USE mesh_struct
USE outfil

```

2.116 L_times_invD

NAME

L_times_invD

SYNOPSIS

```

SUBROUTINE L_times_invD (lda,nrow,d,ncol,u)

```

DESCRIPTION

Compute $L[D]^{-1}$.

ARGUMENTS

lda – integer
nrow – integer
d – complex array
ncol – integer
u – complex array

2.117 L_times_invU

NAME

L_times_invU

SYNOPSIS

SUBROUTINE L_times_invU (lda,nrow,ldu,ncol,l)

DESCRIPTION

Compute $L[U]^{-1}$.

ARGUMENTS

lda – integer (= the dimension of matrix A)
nrow – integer
ldu – complex array (= the ldu factorization of diagonal block)
ncol – integer
l – complex array (= the lower band)

2.118 LdLt_tri_block_numeric

NAME

LdLt_tri_block_numeric

SYNOPSIS

SUBROUTINE LdLt_tri_block_numeric

DESCRIPTION

Numerical factorization using tridiagonal block LdLt algorithm.

ARGUMENTS

None

MODULES

USE solver_param
 USE feti_data
 USE model
 USE outfil
 USE direct_symmetric

2.119 LdLt_tri_block_symbolic

NAME

LdLt_tri_block_symbolic

SYNOPSIS

```
SUBROUTINE LdLt_tri_block_symbolic
```

DESCRIPTION

Symbolic factorization for block tridiagonal LDL^t algorithm.

ARGUMENTS

None

MODULES

```
USE model
USE outfil
USE solver_param
USE feti_data
USE direct_symmetric
```

2.120 Ldlt_tri_dirich

NAME

Ldlt_tri_dirich

SYNOPSIS

```
SUBROUTINE Ldlt_tri_dirich (nop,ldu)
```

DESCRIPTION

Block tridiagonal factorization of a sparse symmetric matrix with reverse Cuthill Mac Kee numbering. Case of Dirichlet conditions for the last block.

ARGUMENTS

nop – real
ldu – factorized matrix structure

MODULES

```
USE solver_param
USE outfil
USE factorized_matrix_struct
```

2.121 LdU_tri_block_numeric

NAME

LdU_tri_block_numeric

SYNOPSIS

```
SUBROUTINE LdU_tri_block_numeric
```

DESCRIPTION

Numerical factorization using tridiagonal block *LDU* algorithm.

ARGUMENTS

None

MODULES

```
USE solver_param
USE feti_data
USE model
USE outfil
USE direct_unsymmetric
```

2.122 LdU_tri_block_symbolic

NAME

LdU_tri_block_symbolic

SYNOPSIS

```
SUBROUTINE LdU_tri_block_symbolic
```

DESCRIPTION

Symbolic factorization for block tridiagonal LdU algorithm.

ARGUMENTS

None

MODULES

```
USE model
USE outfil
USE solver_param
USE feti_data
USE direct_unsymmetric
```

2.123 LdU_tri_dirich

NAME

LdU_tri_dirich

SYNOPSIS

```
SUBROUTINE LdU_tri_dirich (nop,ldu)
```

DESCRIPTION

Block tridiagonal factorization of a sparse symmetric matrix with reverse Cuthill Mac Kee numbering.

ARGUMENTS

nop – real
ldu – factorized matrix structure

MODULES

USE solver_param
 USE outfl
 USE factorized_matrix_struct

2.124 **mask_to_list**

NAME

mask_to_list

SYNOPSIS

```
SUBROUTINE mask_to_list (dim,mask,ntrue,list,P6)
```

DESCRIPTION

Build concatenated list from a mask array.

ARGUMENTS

dim – integer
mask – integer array
ntrue – integer
list – integer array
P6 – integer

2.125 **merge_nodal_list**

NAME

merge_nodal_list

SYNOPSIS

```
SUBROUTINE merge_nodal_list (neq,dim_bc,bc,tot)
```

DESCRIPTION

Build concatenated boundary condition.

ARGUMENTS

neq – integer
dim_bc – integer
bc – nodal list structure array
tot – nodal list structure

MODULES

USE ...

2.126 **mtxv**

NAME

mtxv

SYNOPSIS

SUBROUTINE *mtxv* (*n,m,a,x,y*)

DESCRIPTION

Conjugate matrix-vector product: $y = A^*x$.

ARGUMENTS

n – integer
m – integer
a – complex array
x – complex array
y – complex array

2.127 **mtxv_par**

NAME

mtxv_par

SYNOPSIS

SUBROUTINE *mtxv_par* (*n,m,a,x,y,intf*)

DESCRIPTION

Parallel conjugate matrix-vector product: $y = A^*x$.

ARGUMENTS

n – integer
m – integer
a – complex array
x – complex array
y – complex array
intf – interface descriptor structure

MODULES

USE interf_descriptor_struct
 USE outfil

2.128 **mxvadd**

NAME

mxvadd

SYNOPSIS

SUBROUTINE `mxvadd (n,m,a,x,y)`

DESCRIPTION

Add matrix-vector product: $y = y + Ax$.

ARGUMENTS

n – integer
m – integer
a – complex array
x – complex array
y – complex array

2.129 **nodal_list_equation**

NAME

nodal_list_equation

SYNOPSIS

SUBROUTINE `nodal_list_equation (dom,bc)`

DESCRIPTION

Build list and mask of boundary equations.

ARGUMENTS

dom – mesh structure
bc – nodal list structure

MODULES

```
USE mesh_struct
USE nodal_list_struct
USE realloc
USE outfil
```

2.130 **opti_number**

NAME

opti_number

SYNOPSIS

```
SUBROUTINE opti_number (nodes,p_graph,graph,list_opt)
```

DESCRIPTION

Renumber subgraph via Cuthill-Mac Kee algorithm.

ARGUMENTS

nodes – integer
p_graph – integer array
graph – integer array
list_opt – integer array

MODULES

```
USE outfil
```

2.131 **output_results**

NAME

output_results

SYNOPSIS

```
SUBROUTINE output_results (frequency,freq_num)
```

DESCRIPTION

Print results.

ARGUMENTS

frequency – real
freq_num – integer

MODULES

```

USE outfil
USE model
USE feti_data
USE mpif

```

2.132 point_front

NAME

point_front

SYNOPSIS

```

SUBROUTINE point_front (neq,p_row,sparse_size,col_numb,new2old,    &
&                        nfront,p_front,nclamp,nfront_clamp)

```

DESCRIPTION

Build pointer of fronts of block tridiagonal sparse matrix.

ARGUMENTS

```

    neq - integer
    p_row - integer array (= pointer of rows of unsymmetric matrix)
    sparse_size - integer
    col_numb - integer array (= list of column indices of rows of unsym-
        metric matrix)
    new2old - integer array (= new to old correspondance)
    nfront - integer
    p_front - integer array (= pointer of fronts)
    nclamp - integer
    nfront_clamp - integer

```

MODULES

```

USE outfil

```

2.133 post_process

NAME

post_process

SYNOPSIS

```

SUBROUTINE post_process

```

DESCRIPTION

Post process results.

ARGUMENTS

None

MODULES

USE solver_param
USE model**2.134 print_vector**

NAME

print_vector

SYNOPSIS

SUBROUTINE print_vector (n,v)

DESCRIPTION

Print vector.

ARGUMENTS

n – integer
v – complex array

MODULES

USE outfil

2.135 proband

NAME

proband

SYNOPSIS

SUBROUTINE proband (n,i1,i2,prow1,prow2,lmorse,ncol,a,x,y)

DESCRIPTION

Compute product by a rectangular sub-block of a symmetric sparse matrix.

ARGUMENTS

n – integer
i1 – integer
i2 – integer
prowl – integer array
prow2 – integer array
lmorse – integer
ncol – integer array
a – complex array
x – complex array
y – complex array

2.136 **probit**

NAME

probit

SYNOPSIS

SUBROUTINE `probit` (`intf,w,bitw`)

DESCRIPTION

Compute the local right-hand-side associated with a Neumann boundary condition on the interface.

ARGUMENTS

intf – interface descriptor structure
w – complex array
bitw – complex array

MODULES

USE `interf_descriptor_struct`
 USE `outfil`

2.137 **problem_form**

NAME

problem_form

SYNOPSIS

SUBROUTINE `problem_form` (`frequency,subdom_num,subdom,typesol,&`
 & `symm,k_opt,typegalerkin`)

DESCRIPTION

Form the matrix of domain.

ARGUMENTS

frequency – real
subdom_num – integer
numb_subdom – integer
typesol – integer
symm – integer
k_opt – integer
typegalerkin – integer

MODULES

USE model
 USE outfil
 USE feti_data

2.138 **problo**

NAME

problo

SYNOPSIS

SUBROUTINE *problo* (n,i1,i2,prow1,prow2,lmorse,ncol,a,x,y)

DESCRIPTION

Compute product by a sub-block of a sparse matrix.

ARGUMENTS

n – integer
i1 – integer
i2 – integer
prow1 – integer array
prow2 – integer array
lmorse – integer
ncol – integer array
a – complex array
x – complex array
y – complex array

2.139 **problot**

NAME

problot

SYNOPSIS

SUBROUTINE *problot* (n,i1,i2,prow1,prow2,lmorse,ncol,a,x,y)

DESCRIPTION

Compute product by a transposed sub-block of a sparse matrix.

ARGUMENTS

n – integer
i1 – integer
i2 – integer
proW1 – integer array
proW2 – integer array
lmorse – integer
ncol – integer array
a – complex array
x – complex array
y – complex array

2.140 **pt_extract**

NAME

pt_extract

SYNOPSIS

```
SUBROUTINE pt_extract (n,i1,i2,j0,proW1,proW2,lmorse,ncol,a,dim2, &
& dim1,blo)
```

DESCRIPTION

Extract transposed renumbered block associated to rows i1 to i2 and delimited by pointers proW1 et proW2 from sparse matrix stored by row.

ARGUMENTS

n – integer
i1 – integer
i2 – integer
j0 – integer
proW1 – integer array
proW2 – integer array
lmorse – integer
ncol – integer array
a – complex array
dim2 – integer
dim1 – integer
blo – complex array

2.141 **read_mesh**

NAME

read_mesh

SYNOPSIS

```
SUBROUTINE read_mesh (file_num,dom,max_numb_f,numb_f,f)
```

DESCRIPTION

Input mesh in free format.

ARGUMENTS

file_num – integer
dom – mesh structure
max_numb_f – integer
numb_f – integer
f – facet structure array

MODULES

```
USE mesh_struct
USE facet_struct
USE realloc
USE outfil
```

2.142 **realloc_c1**

NAME

realloc_c1

SYNOPSIS

```
SUBROUTINE realloc_c1 (tab,dim1,ierr)
```

DESCRIPTION

Reallocation of multi-dimensional complex pointer in the case where the pointer has been allocated.

ARGUMENTS

tab – complex array
dim1 – integer
ierr – integer

2.143 **realloc_c2**

NAME

realloc_c2

SYNOPSIS

```
SUBROUTINE realloc_c2 (tab,dim1,dim2,ierr)
```

DESCRIPTION

Reallocation of multi-dimensional complex pointer in the case where the pointer has been allocated.

ARGUMENTS

tab – complex array
dim1 – integer
dim2 – integer
ierr – integer

2.144 **realloc_i1**

NAME

realloc_i1

SYNOPSIS

SUBROUTINE `realloc_i1` (`tab,dim1,ierr`)

DESCRIPTION

Reallocation of multi-dimensional integer pointer in the case where the pointer has been allocated.

ARGUMENTS

tab – integer array
dim1 – integer
ierr – integer

2.145 **realloc_i2**

NAME

realloc_i2

SYNOPSIS

SUBROUTINE `realloc_i2` (`tab,dim1,dim2,ierr`)

DESCRIPTION

Reallocation of multi-dimensional integer pointer in the case where the pointer has been allocated.

ARGUMENTS

tab – integer array
dim1 – integer
dim2 – integer
ierr – integer

2.146 realloc_r1

NAME

realloc_r1

SYNOPSIS

SUBROUTINE `realloc_r1 (tab,dim1,ierr)`

DESCRIPTION

Reallocation of multi-dimensional real pointer in the case where the pointer has been allocated.

ARGUMENTS

tab – real array
dim1 – integer
ierr – integer

2.147 realloc_r2

NAME

realloc_r2

SYNOPSIS

SUBROUTINE `realloc_r2 (tab,dim1,dim2,ierr)`

DESCRIPTION

Reallocation of multi-dimensional real pointer in the case where the pointer has been allocated.

ARGUMENTS

tab – real array
dim1 – integer
dim2 – integer
ierr – integer

2.148 ren_sym_sparse

NAME

ren_sym_sparse

SYNOPSIS

SUBROUTINE `ren_sym_sparse (neq,sparse_size,rowp,col,new2old,p_row, &
& col_num)`

DESCRIPTION

Construction of renumbered lower triangular part of the symmetric sparse structure stored row by row from the upper triangular symmetric one.

ARGUMENTS

rowp – pointer of rows of the upper triangular part of the initial matrix
col – list of column indices of rows of the initial matrix
new2old – new to old correspondance
p_row – pointer of rows of the lower triangular part of the symmetric matrix
col_numb – list of column indices of rows of the renumbered matrix
 ... – ...

MODULES

USE outfil

2.149 **ren_unsym_sparse**

NAME

ren_unsym_sparse

SYNOPSIS

```
SUBROUTINE ren_unsym_sparse (neq,sparse_size,rowp,col,new2old,      &
&                             p_row,col_numb)
```

DESCRIPTION

Construction of renumbered unsymmetric sparse structure stored row by row.

ARGUMENTS

rowp – pointer of rows of the initial matrix
col – list of column indices of rows of the initial matrix
new2old – new to old correspondance
p_row – pointer of rows of the unsymmetric matrix
col_numb – list of column indices of rows of the renumbered matrix

MODULES

USE outfil

2.150 **renum_eq_mpc**

NAME

renum_eq_mpc

SYNOPSIS

```
SUBROUTINE renum_eq_mpc (dom,dim_mpc,mpc)
```

DESCRIPTION

Change equation numbering to take into account multi-point constraints.

ARGUMENTS

dom – mesh structure
dim_mpc – integer
mpc – nodal list structure array

MODULES

```
USE outfil
USE mesh_struct
USE Nodal_List_struct
```

2.151 **renum_facet**

NAME

renum_facet

SYNOPSIS

```
SUBROUTINE renum_facet (dom,bc)
```

DESCRIPTION

Replace external to internal node number in list of nodes of facet.

ARGUMENTS

dom – mesh structure
bc – facet structure

MODULES

```
USE mesh_struct
USE facet_struct
```

2.152 **renum_nodal_list**

NAME

renum_nodal_list

SYNOPSIS

```
SUBROUTINE renum_nodal_list (dom,bc)
```

DESCRIPTION

Replace external to internal node number in nodal list.

ARGUMENTS

dom – mesh structure
bc – nodal list structure

MODULES

USE mesh_struct
 USE nodal_list_struct

2.153 solve_rhs

NAME

solve_rhs

SYNOPSIS

SUBROUTINE solve_rhs (subdom_num)

DESCRIPTION

Solve problem for a single right hand side.

ARGUMENTS

subdom_num – integer

MODULES

USE solver_param
 USE model
 USE outfil
 USE realloc
 USE feti_data
 USE feti_param
 USE mpif

2.154 son_3d

NAME

son_3d

SYNOPSIS

PROGRAM son_3d

DESCRIPTION

FETI demonstration code.

ARGUMENTS

None

MODULES

USE solver_param

2.155 **sort_list**

NAME

sort_list

SYNOPSIS

```
SUBROUTINE sort_list (dim,list)
```

DESCRIPTION

Sort list of interger in increasing ordering.

ARGUMENTS

dim – integer
list – integer array

2.156 **sparse_fwbw**

NAME

sparse_fwbw

SYNOPSIS

```
SUBROUTINE sparse_fwbw (neq,sparse_size,p_row,column_num,a,      &  
&                        p_diag,b,z)
```

DESCRIPTION

Sparse forward-backward solution of $Ax = b$ with $A = LU$.

ARGUMENTS

neq – integer
sparse_size – integer
p_row – integer array
column_num – integer array
a – complex array
p_diag – integer array
b – complex array
z – complex array

2.157 **sparse_matrix_equalization**

NAME

sparse_matrix_equalization

SYNOPSIS

SUBROUTINE `sparse_matrix_equalization` (b,a)

DESCRIPTION

Equalize sparse matrix.

ARGUMENTS

b – sparse matrix structure
a – sparse matrix structure

MODULES

USE `sparse_matrix_struct`
 USE `realloc`
 USE `outfil`

2.158 **sparse_matrix_equalization2**

NAME

sparse_matrix_equalization2

SYNOPSIS

SUBROUTINE `sparse_matrix_equalization2` (b,a)

DESCRIPTION

Equalize sparse matrix.

ARGUMENTS

b – sparse matrix structure
a – sparse matrix structure

MODULES

USE sparse_matrix_struct

2.159 **sparse_prod**

NAME

sparse_prod

SYNOPSIS

SUBROUTINE sparse_prod (a,x,y)

DESCRIPTION

Compute sparse matrix vector product in the case of a symmetric matrix with upper triangular part stored by rows.

ARGUMENTS

a – sparse matrix structure
x – complex array
y – complex array

MODULES

USE sparse_matrix_struct

2.160 **split_mesh**

NAME

split_mesh

SYNOPSIS

SUBROUTINE split_mesh (numb_subdom)

DESCRIPTION

Build elements to subdomain correspondance.

ARGUMENTS

numb_subdom – integer

MODULES

USE model
 USE feti_data
 USE outfil

2.161 stoprun

NAME

stoprun

SYNOPSIS

SUBROUTINE stoprun

DESCRIPTION

Abort run.

ARGUMENTS

None

MODULES

USE outfil

2.162 sym_approx_schur_cpmt

NAME

sym_approx_schur_cpmt

SYNOPSIS

SUBROUTINE sym_approx_schur_cpmt (a, ainit, reg, intf, intf_mesh, Diri, &
 & Dir)

DESCRIPTION

Compute Schur complement.

ARGUMENTS

a – sparse matrix structure
ainit – sparse matrix structure
reg – sparse matrix structure
intf – interface descriptor structure
intf_mesh – interface mesh structure
Diri – nodal list structure
Dir – nodal list structure

MODULES

```

USE mpif
USE solver_param
USE feti_data
USE sparse_matrix_struct
USE direct_symmetric
USE interf_mesh_struct
USE interf_descriptor_struct
USE nodal_list_struct
USE outfil
USE realloc

```

2.163 **sym_build_schur_cpmt**

NAME

sym_build_schur_cpmt

SYNOPSIS

```

SUBROUTINE sym_build_schur_cpmt (ainit,reg,intf,intf_mesh,Diri,   &
&                               Dir)

```

DESCRIPTION

Compute Schur complement.

ARGUMENTS

ainit – sparse matrix structure
reg – sparse matrix structure
intf – interface descriptor structure
intf_mesh – interface mesh structure
Diri – nodal list structure
Dir – nodal list structure

MODULES

```

USE mpif
USE solver_param
USE feti_data
USE sparse_matrix_struct
USE direct_symmetric
USE interf_mesh_struct
USE interf_descriptor_struct
USE nodal_list_struct
USE outfil

```

2.164 **sym_condens**

NAME

sym_condens

SYNOPSIS

```
SUBROUTINE sym_condens (lda,dim1,dim2,a,d,b,c,bt,i0,nop)
```

DESCRIPTION

Compute the Schur complement: $C = C - [L^{-1}B]^t[L^{-1}B]$ where L is the lower triangular part of the Choleski factorisation of A .

ARGUMENTS

lda – integer
dim1 – integer
dim2 – integer
a – complex array
d – complex array
b – complex array
c – complex array
bt – complex array
i0 – integer array
nop – real

MODULES

```
USE block_size
```

2.165 **sym_direct_solve**

NAME

sym_direct_solve

SYNOPSIS

```
SUBROUTINE sym_direct_solve (y,x)
```

DESCRIPTION

Forward-backward substitution using tridiagonal block LdU factorization.

ARGUMENTS

y – complex array
x – complex array

MODULES

```
USE model  

USE direct_symmetric
```

2.166 sym_extract

NAME

sym_extract

SYNOPSIS

```

SUBROUTINE sym_extract (n,i1,i2,j0,prow1,prow2,lmorse,ncol,a,lda, &
& dim2,blo)

```

DESCRIPTION

Extract block associated to rows *i1* to *i2* and delimited by pointers *prow1* et *prow2* from sparse matrix stored by row.

ARGUMENTS

n – integer
i1 – integer
i2 – integer
j0 – integer
prow1 – integer array
prow2 – integer array
lmorse – integer
ncol – integer
a – complex array
lda – integer
dim2 – integer
blo – complex array

2.167 sym_fill_tridiag_sparse

NAME

sym_fill_tridiag_sparse

SYNOPSIS

```

SUBROUTINE sym_fill_tridiag_sparse (a,ldu)

```

DESCRIPTION

Fill renumbered matrix with storage of lower triangular part.

ARGUMENTS

a – sparse matrix structure
ldu – factorized matrix structure

MODULES

```

USE sparse_matrix_struct
USE factorized_matrix_struct

```

2.168 **sym_fwbw_tri_dirichlet**

NAME

sym_fwbw_tri_dirichlet

SYNOPSIS

SUBROUTINE `sym_fwbw_tri_dirichlet` (`y,x`)

DESCRIPTION

Forward-backward substitution using tridiagonal block *LDU* factorization.

ARGUMENTS

`y` – complex array
`x` – complex array

MODULES

USE `model`
USE `direct_symmetric`**2.169** **sym_multi_level1_feti**

NAME

sym_multi_level1_feti

SYNOPSIS

SUBROUTINE `sym_multi_level1_feti` (`rhs,solution,a,a_init,` &
& `Dirichlet,interfine,typre,numb_dir,` &
& `max_numb_it,numb_restart,epsilon,` &
& `epstagn,multi_rhs`)

DESCRIPTION

Level-1 FETI solution with multi-Krylov projection.

ARGUMENTS

rhs – complex array
solution – complex array
a – sparse matrix structure
a_init – sparse matrix structure
Dirichlet – nodal list structure
interfine – interface descriptor structure
typre – integer
numb_dir – integer
max_numb_it – integer
numb_restart – integer
epsilon – real
epstagn – real
multi_rhs – integer

MODULES

USE mpif
 USE direct_symmetric
 USE factorized_matrix_struct
 USE sparse_matrix_struct
 USE interf_descriptor_struct
 USE nodal_list_struct
 USE outfil

2.170 **sym_multi_level2_feti**

NAME

sym_multi_level2_feti

SYNOPSIS

```

SUBROUTINE sym_multi_level2_feti (rhs,solution,a,a_init,b,      &
&                               Dirichlet,interfineP,typre,   &
&                               numb_dir,max_numb_it,numb_restart, &
&                               epsilon,epstagn,multi_rhs)

```

DESCRIPTION

Solution of interface problem using a Jacobi algorithm.

ARGUMENTS

rhs – complex array
solution – complex array
a – sparse matrix structure
a_init – sparse matrix structure
Dirichlet – nodal list structure
interfineP – interface descriptor structure
typre – integer
numb_dir – integer
max_numb_it – integer
numb_restart – integer
epsilon – real
epstagn – real
multi_rhs – integer

MODULES

USE mpif
 USE direct_symmetric
 USE factorized_matrix_struct
 USE sparse_matrix_struct
 USE interf_descriptor_struct
 USE nodal_list_struct
 USE feti_data
 USE outfil

2.171 **sym_proax**

NAME

sym_proax

SYNOPSIS

SUBROUTINE *sym_proax* (*neq,p_row,sparse_size,column_num,coef,x,y*)

DESCRIPTION

Compute sparse matrix vector product in the case of a symmetric matrix with upper triangular part stored by rows.

ARGUMENTS

neq – integer
p_row – integer array
sparse_size – integer
column_num – integer array
coef – complex array
x – complex array
y – complex array

2.172 **sym_renumb_matrix_bc**

NAME

sym_renumb_matrix_bc

SYNOPSIS

SUBROUTINE `sym_renumb_matrix_bc` (`a`,`ldu`)

DESCRIPTION

Build renumbered matrix with storage of lower triangular part.

ARGUMENTS

`a` – sparse matrix structure
`ldu` – factorized matrix structure

MODULES

USE `sparse_matrix_struct`
USE `nodal_list_struct`
USE `outfil`
USE `realloc`
USE `factorized_matrix_struct`**2.173** **sym_tridiag_struct**

NAME

sym_tridiag_struct

SYNOPSIS

SUBROUTINE `sym_tridiag_struct` (`ldu`)

DESCRIPTION

Build block tridiagonal sparse matrix structure of symmetric matrix renumbered via Cuthill Mac Kee.

ARGUMENTS

`ldu` – factorized matrix structure

MODULES

USE `solver_param`
USE `outfil`
USE `realloc`
USE `factorized_matrix_struct`

2.174 symfwbw_tri_dirichlet

NAME

symfwbw_tri_dirichlet

SYNOPSIS

SUBROUTINE `symfwbw_tri_dirichlet` (`y,x,ldu`)

DESCRIPTION

Forward-backward substitution with tridiagonal block sparse matrix. Case of Dirichlet conditions for the last block.

ARGUMENTS

`y` – complex array
`x` – complex array
`ldu` – factorized matrix structure

MODULES

USE `solver_param`
 USE `outfil`

2.175 symunsym_sparse_mask

NAME

symunsym_sparse_mask

SYNOPSIS

SUBROUTINE `symunsym_sparse_mask` (`neq,rowp,lcol,col,p_row,` &
 & `sparse_size,col_numb,mask`)

DESCRIPTION

Construction of the un-symmetric sparse structure from the upper triangular part of the symmetric one stored row by row.

ARGUMENTS

`rowp` – pointer of rows of upper triangular part of symmetric matrix
`col` – list of column indices of rows of upper triangular part
`p_row` – pointer of rows of unsymmetric matrix
`col_numb` – list of column indices of rows of unsymmetric matrix
`mask` – mask of eliminated equations

MODULES

USE `outfil`

2.176 un_condens

NAME

un_condens

SYNOPSIS

```
SUBROUTINE un_condens (lda,dim1,dim2,a,d,e,f,c,v,ie0,if0,nop)
```

DESCRIPTION

Compute the Schur complement $C = C - [U^{-t}F]^t d^{-1} [L^{-1}E]$ LDU is the Gauss-Jordan factorisation of A .

ARGUMENTS

lda – integer
dim1 – integer
dim2 – integer
a – complex array
d – complex array
e – complex array
f – complex array
c – complex array
v – complex array
ie0 – integer array
if0 – integer array
nop – real

2.177 un_extract

NAME

un_extract

SYNOPSIS

```
SUBROUTINE un_extract(n,i1,i2,j0,prowl,prow2,lmorse,ncol,a,lda, &
& dim2,blo)
```

DESCRIPTION

Extract block associated to rows $i1$ to $i2$ and delimited by pointers $prowl$ et $prow2$ from sparse matrix stored by row

ARGUMENTS

n – integer
i1 – integer
i2 – integer
j0 – integer
proW1 – integer array
proW2 – integer array
lmorse – integer
ncol – integer array
a – complex array
lda – integer
dim2 – integer
blo – complex array

2.178 **unsym_approx_schur_cpmt**

NAME

unsym_approx_schur_cpmt

SYNOPSIS

```

SUBROUTINE unsym_approx_schur_cpmt(a, ainit, reg, intf, intf_mesh,    &
&                                     Diri, Dir)

```

DESCRIPTION

Compute Schur complement.

ARGUMENTS

a – sparse matrix structure
ainit – sparse matrix structure
reg – sparse matrix structure
intf – interface descriptor structure
intf_mesh – interface mesh structure
Diri – nodal list structure
Dir – nodal list structure

MODULES

```

USE mpif
USE solver_param
USE feti_data
USE sparse_matrix_struct
USE direct_unsymmetric
USE interf_mesh_struct
USE interf_descriptor_struct
USE nodal_list_struct
USE outfil
USE realloc

```

2.179 **unsym_build_schur_cpmt**

NAME

unsym_build_schur_cpmt

SYNOPSIS

```

SUBROUTINE unsym_build_schur_cpmt (ainit,reg,intf,intf_mesh,Diri, &
&                               Dir)

```

DESCRIPTION

Compute Schur complement

ARGUMENTS

ainit – sparse matrix structure
reg – sparse matrix structure
intf – interface descriptor structure
intf_mesh – interface mesh structure
Diri – nodal list structure
Dir – modal list structure

MODULES

```

USE mpif
USE solver_param
USE feti_data
USE sparse_matrix_struct
USE direct_undefined
USE interf_mesh_struct
USE interf_descriptor_struct
USE nodal_list_struct
USE outfil
USE realloc

```

2.180 **unsym_direct_solve**

NAME

unsym_direct_solve

SYNOPSIS

```

SUBROUTINE unsym_direct_solve (y,x)

```

DESCRIPTION

Forward-backward substitution using tridiagonal block LdU factorization

ARGUMENTS

y – complex array
x – complex array

MODULES

USE model
 USE direct_unsymmetric

2.181 unsym_fill_tridiag_sparse

NAME

unsym_fill_tridiag_sparse

SYNOPSIS

SUBROUTINE `unsym_fill_tridiag_sparse` (a,ldu)

DESCRIPTION

Fill renumbered matrix

ARGUMENTS

a – sparse matrix structure
ldu – factorized matrix structure

MODULES

USE sparse_matrix_struct
 USE factorized_matrix_struct

2.182 unsym_fwbw_tri_dirichlet

NAME

unsym_fwbw_tri_dirichlet

SYNOPSIS

SUBROUTINE `unsym_fwbw_tri_dirichlet`(y,x)

DESCRIPTION

Forward-backward substitution using tridiagonal block LdU factorization.

ARGUMENTS

y – complex array
x – complex array

MODULES

```
USE model
USE direct_unsymmetric
```

2.183 **unsym_multi_level1_feti**

NAME

unsym_multi_level1_feti

SYNOPSIS

```
SUBROUTINE unsym_multi_level1_feti(rhs,solution,a,a_init,      &
&                               Dirichlet, interfine,type,    &
&                               numb_dir,max_numb_it,numb_restart, &
&                               epsilon,epstagn,multi_rhs)
```

DESCRIPTION

Level-1 FETI solution with multi-Krylov projection

ARGUMENTS

rhs – complex array
solution – complex array
a – sparse matrix structure
a_init – sparse matrix structure
Dirichlet – nodal list structure
interfine – interface descriptor structure
type – integer
numb_dir – integer
max_numb_it – integer
numb_restart – integer
epsilon – real
epstagn – real
multi_rhs – integer

MODULES

```
USE mpif
USE direct_unsymmetric
USE factorized_matrix_struct
USE sparse_matrix_struct
USE interf_descriptor_struct
USE nodal_list_struct
USE outfil
```

2.184 **unsym_multi_level2_feti**

NAME

unsym_multi_level2_feti

SYNOPSIS

```

SUBROUTINE unsym_multi_level2_feti (rhs,solution,a,a_init,b,      &
&                                  Dirichlet,interfine,  typre,    &
&                                  numb_dir,max_numb_it,numb_restart, &
&                                  epsilon,epstagn,multi_rhs)

```

DESCRIPTION

Level-2 FETI solution with multi-Krylov projection.

ARGUMENTS

rhs – complex array
solution – complex array
a – sparse matrix structure
a_init – sparse matrix structure
Dirichlet – nodal list structure
interfineP – interface descriptor structure
typre – integer
numb_dir – integer
max_numb_it – integer
numb_restart – integer
epsilon – real
epstagn – real
multi_rhs – integer

MODULES

```

USE mpif
USE direct_unsymmetric
USE factorized_matrix_struct
USE sparse_matrix_struct
USE interf_descriptor_struct
USE interf_mesh_struct
USE nodal_list_struct
USE outfil

```

2.185 **unsym_proax**

NAME

unsym_proax

SYNOPSIS

```

SUBROUTINE unsym_proax (neq,p_row,sparse_size,column_num,coef,x,y)

```

DESCRIPTION

Compute sparse matrix vector product in the case of an unsymmetric matrix stored by rows.

ARGUMENTS

neq – integer
p_row – integer array
sparse_size – integer
column_numb – integer array
coef – complex array
x – complex array
y – complex array

2.186 **unsym_renumb_matrix_bc**

NAME

unsym_renumb_matrix_bc

SYNOPSIS

SUBROUTINE `unsym_renumb_matrix_bc` (`a`,`ldu`)

DESCRIPTION

Build renumbered matrix.

ARGUMENTS

a – sparse matrix structure
nrow – factorized matrix structure

MODULES

USE `sparse_matrix_struct`
 USE `nodal_list_struct`
 USE `outfil`
 USE `realloc`
 USE `factorized_matrix_struct`

2.187 **unsym_tridiag_struct**

NAME

unsym_tridiag_struct

SYNOPSIS

SUBROUTINE `unsym_tridiag_struct`(`ldu`)

DESCRIPTION

Build block tridiagonal sparse matrix structure of unsymmetric matrix renumbered via Cuthill Mac Kee.

ARGUMENTS

ldu – factorized matrix struct

MODULES

USE solver_param
 USE outfl
 USE realloc
 USE factorized_matrix_struct

2.188 **unsymfwbw_tri_dirichlet**

NAME

unsymfwbw_tri_dirichlet

SYNOPSIS

SUBROUTINE `unsymfwbw_tri_dirichlet` (`y,x,ldu`)

DESCRIPTION

Forward-backward substitution with tridiagonal block sparse matrix. Case of Dirichlet conditions for the last block.

ARGUMENTS

y – complex array
x – complex array
ldu – factorized matrix structure

MODULES

USE solver_param
 USE outfl
 USE factorized_matrix_struct

2.189 **unsymunsym_sparse_mask**

NAME

unsymunsym_sparse_mask

SYNOPSIS

SUBROUTINE `unsymunsym_sparse_mask`(`neq,rowp,lcol,col,p_row,` &
 & `sparse_size,col_numb,mask`)

DESCRIPTION

Construction of the non-symmetric sparse structure with elimination of non diagonal entries of clamped rows and columns.

ARGUMENTS

rowp – pointer of rows of non symmetric matrix
col – list of column indices of rows
p_row – pointer of rows of compressed matrix
col_numb – list of column indices of rows of compressed matrix
mask – mask of eliminated equations

MODULES

USE outfil

2.190 **update_facet**

NAME

update_facet

SYNOPSIS

SUBROUTINE update_facet (dom,bc)

DESCRIPTION

Find facets material number.

ARGUMENTS

dom – mesh structure
bc – facet structure

MODULES

USE mesh_struct
 USE facet_struct
 USE outfil

2.191 **Update_Nodal_List**

NAME

Update_Nodal_List

SYNOPSIS

SUBROUTINE Update_Nodal_List (dom,bc)

DESCRIPTION

Find neighboring elements of boundary nodes

ARGUMENTS

dom – mesh structure
bc – nodal list structure

MODULES

```

USE mesh_struct
USE nodal_list_struct
USE realloc
USE outfil

```

2.192 **update_region**

NAME

update_region

SYNOPSIS

```
SUBROUTINE update_region (dom,acou_mat,admi_mat,inf_mat)
```

DESCRIPTION

Change region id to internal region number for elements and build internal region number of adjacent element.

ARGUMENTS

dom – mesh structure
acou_mat – region property structure array
admi_mat – region property structure array
inf_mat – region property structure array

MODULES

```

USE mesh_struct
USE region_property_struct
USE outfil

```

2.193 **write_field**

NAME

write_field

SYNOPSIS

```
SUBROUTINE write_field (file_num,dm,field)
```

DESCRIPTION

Print field entries node by node

ARGUMENTS

file_num – integer
dm – mesh structure
field – complex array

MODULES

USE outfl
USE mesh_struct

2.194 write_field_femview

NAME

write_field_femview

SYNOPSIS

SUBROUTINE write_field_femview (file_num,dom,field)

DESCRIPTION

Write the complex components of pressure in a FEMVIEW file

ARGUMENTS

file_num – integer
dom – mesh structure
field – complex array

MODULES

USE outfl
USE mesh_struct

2.195 write_matrix

NAME

write_matrix

SYNOPSIS

SUBROUTINE write_matrix (a,frequency,subdom_num,numb_subdom, &
& directory)

DESCRIPTION

Print complex sparse matrix.

ARGUMENTS

a – sparse matrix structure
frequency – real
subdom_num – integer
numb_subdom – integer
directory – character string

MODULES

```
USE outfil
USE sparse_matrix_struct
```

2.196 write_meshfield_femvtk

NAME

write_meshfield_femvtk

SYNOPSIS

```
SUBROUTINE write_meshfield_femvtk (file_num,dom,field)
```

DESCRIPTION

Write the complex components of pressure in a VTK file

ARGUMENTS

```
file_num – integer
dom – mesh structure
field – complex array
```

MODULES

```
USE outfil
USE mesh_struct
```

2.197 zorthodir_solver

NAME

zorthodir_solver

SYNOPSIS

```
SUBROUTINE zorthodir_solver (rhs,x,a,Dirichlet,type_precond,      &
&                           numb_dir,max_numb_it,epsilon,      &
&                           subdom_numb)
```

DESCRIPTION

Iterative solution of complex problem by Orthodir algorithm with restarts.

ARGUMENTS

rhs – complex array
x – complex array
a – sparse matrix structure
Dirichlet – nodal list structure
type_precond – integer
numb_dir – integer
max_numb_it – integer
epsilon – real
subdom_numb – integer

MODULES

USE sparse_matrix_struct
 USE nodal_list_struct
 USE outfil

2.198 **zorthodir_solver_par**

NAME

zorthodir_solver_par

SYNOPSIS

```

SUBROUTINE zorthodir_solver_par (rhs,x,a,Dirichlet,intf,           &
&                               type_precond,numb_dir,max_numb_it,epsilon,&
&                               subdom_numb)

```

DESCRIPTION

Parallel Iterative solution of complex problem by Orthodir algorithm with restarts.

ARGUMENTS

rhs – complex array
x – complex array
a – sparse matrix structure
Dirichlet – nodal list structure
intf – interface descriptor structure
type_precond – integer
numb_dir – integer
max_numb_it – integer
epsilon – real
subdom_numb – integer

MODULES

USE sparse_matrix_struct
 USE nodal_list_struct
 USE interf_descriptor_struct
 USE outfil

Acknowledgements

The authors acknowledge partial financial support by the European Community under the Enhancing Access to Research Infrastructure action of the Improving Human Potential programme, contract number HPRI-1999-CT-0026.