



**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

<b>1</b>	<b>Modules</b>	<b>17</b>
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

<b>2</b>	<b>Functions</b>	<b>29</b>
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

**type** – 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 outfl  
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 outfl  
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\_interf\_matrix**

## NAME

*assemble\_interf\_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 outfl

**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 outfl

**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 outfl

**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_num`)

DESCRIPTION

Print mesh data structure.

ARGUMENTS

**P6** – integer (output unit)**dom** – mesh structure**subdom\_num** – 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`,`unassemb`,`assemb`)

DESCRIPTION

Average a field along the interface

ARGUMENTS

**intf** – interface descriptor structure  
**unassemb** – complex array (= unassembled field as input)  
**assemb** – 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,numb_subdom,typesol,&`  
`& symm,k_opt,typegalerkin)`

## DESCRIPTION

Form the matrix of domain.



## ARGUMENTS

**frequency** – real  
**subdom\_numb** – 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\_numb** – 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,pro1,pro2,lmorse,ncol,a,lda, &
& dim2,blo)
```

DESCRIPTION

Extract block associated to rows  $i1$  to  $i2$  and delimited by pointers  $pro1$  et  $pro2$  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.