

progress

0.0.1

Generated by Doxygen 1.8.11

Contents

1	README	1
2	Testing the Progress library	5
3	Todo List	7
4	Module Index	9
4.1	Modules	9
5	Modules Index	11
5.1	Modules List	11
6	Data Type Index	13
6.1	Data Types List	13
7	File Index	15
7.1	File List	15
8	Module Documentation	17
8.1	(LATTE related routines)	17
8.2	(PROGRESS related routines)	18
8.2.1	Detailed Description	19
8.3	(EXTERNAL related routines)	20
8.4	(High-level codes using PROGRESS/LATTE modules)	21

9	Module Documentation	23
9.1	prg_charges_mod Module Reference	23
9.1.1	Detailed Description	23
9.1.2	Function/Subroutine Documentation	23
9.1.2.1	prg_get_charges(rho_bml, over_bml, hindex, charges, numel, spindex, mdimin, threshold)	23
9.1.2.2	prg_get_hscf(ham0_bml, over_bml, ham_bml, spindex, hindex, hubbardu, charges, coulomb_pot_r, coulomb_pot_k, mdimin, threshold)	24
9.1.3	Variable Documentation	25
9.1.3.1	dp	25
9.2	prg_chebyshev_mod Module Reference	25
9.2.1	Detailed Description	26
9.2.2	Function/Subroutine Documentation	26
9.2.2.1	absmaxderivative(func, de)	26
9.2.2.2	fermi(e, ef, kbt)	26
9.2.2.3	jackson(ncoeffs, i, jon)	27
9.2.2.4	prg_build_density_cheb(ham_bml, rho_bml, athr, threshold, ncoeffs, kbt, ef, bndfil, jon, verbose)	27
9.2.2.5	prg_build_density_cheb_fermi(ham_bml, rho_bml, athr, threshold, ncoeffs, kbt, ef, bndfil, getef, fermitol, jon, npts, trkfunc, verbose)	28
9.2.2.6	prg_get_chebcoeffs(npts, kbt, ef, ncoeffs, coeffs, emin, emax)	29
9.2.2.7	prg_get_chebcoeffs_fermi_bs(npts, kbt, ef, tracesT, ncoeffs, coeffs, emin, emax, bndfil, norb, tol, jon, verbose)	30
9.2.2.8	prg_get_chebcoeffs_fermi_nt(npts, kbt, ef, tracesT, ncoeffs, coeffs, emin, emax, bndfil, norb, tol, jon, verbose)	31
9.2.2.9	prg_parse_cheb(chebdata, filename)	32
9.2.2.10	tr(r, x)	33
9.2.3	Variable Documentation	33
9.2.3.1	dp	33
9.2.3.2	pi	33
9.3	prg_densitymatrix_mod Module Reference	33
9.3.1	Detailed Description	34
9.3.2	Function/Subroutine Documentation	34

9.3.2.1	fermi(e, ef, kbt)	34
9.3.2.2	prg_build_atomic_density(rhoat_bml, numel, hindex, spindex, norb, bml_type)	35
9.3.2.3	prg_build_density_t(ham_bml, rho_bml, threshold, bndfil, kbt, ef)	35
9.3.2.4	prg_build_density_t0(ham_bml, rho_bml, threshold, bndfil)	36
9.3.2.5	prg_build_density_t_fermi(ham_bml, rho_bml, threshold, kbt, ef, verbose)	36
9.3.2.6	prg_check_idempotency(mat_bml, threshold, idempotency)	37
9.3.2.7	prg_get_eigenvalues(ham_bml, eigenvalues, verbose)	37
9.3.2.8	prg_get_flevel(eigenvalues, kbt, bndfil, tol, Ef)	38
9.3.3	Variable Documentation	38
9.3.3.1	dp	38
9.4	prg_dos_mod Module Reference	39
9.4.1	Detailed Description	39
9.4.2	Function/Subroutine Documentation	39
9.4.2.1	lorentz(energy, eigenvals, loads, Gamma)	39
9.4.2.2	prg_write_tdos(eigenvals, gamma, npts, emin, emax, filename)	40
9.4.3	Variable Documentation	40
9.4.3.1	dp	40
9.5	prg_extras_mod Module Reference	41
9.5.1	Detailed Description	41
9.5.2	Function/Subroutine Documentation	41
9.5.2.1	mls()	41
9.5.2.2	prg_delta(x, s, nn, dta)	42
9.5.2.3	prg_get_mem(procname, tag)	42
9.5.2.4	prg_print_matrix(matname, amat, i1, i2, j1, j2)	43
9.5.2.5	prg_twonorm(a, nn, norm2)	43
9.5.2.6	to_string_double(x)	43
9.5.2.7	to_string_integer(i)	44
9.5.2.8	to_string_long_long(i)	44
9.5.3	Variable Documentation	44
9.5.3.1	dp	44

9.6	<code>prg_genz_mod</code> Module Reference	45
9.6.1	Detailed Description	45
9.6.2	Function/Subroutine Documentation	45
9.6.2.1	<code>prg_allocatezspmat(self, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, norb, bml_type)</code>	45
9.6.2.2	<code>prg_buildzdiag(smat_bml, zmat_bml, threshold, mdimin, bml_type, verbose)</code>	46
9.6.2.3	<code>prg_buildzsparse(smat_bml, zmat_bml, igenz, mdim, bml_type, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, nfirst, nrefi, nreff, thresholdi, thresholdf, integration, verbose)</code>	46
9.6.2.4	<code>prg_generate(self, over_bml, zmat_bml, igenz, mdim, bml_type, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml)</code>	47
9.6.2.5	<code>prg_genz_sp_initial_zmat(smat_bml, zmat_bml, norb, mdim, bml_type_f, threshold)</code>	48
9.6.2.6	<code>prg_genz_sp_initialz0(smat_bml, zmat_bml, norb, mdim, bml_type_f, threshold)</code>	48
9.6.2.7	<code>prg_genz_sp_int(zmat_bml, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, igenz, norb, bml_type, threshold)</code>	49
9.6.2.8	<code>prg_genz_sp_ref(smat_bml, zmat_bml, nref, norb, bml_type, threshold)</code>	49
9.6.2.9	<code>prg_init(self, input)</code>	49
9.6.2.10	<code>prg_init_zspmat(igenz, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, norb, bml_type)</code>	50
9.6.2.11	<code>prg_parse_zsp(input, filename)</code>	50
9.6.3	Variable Documentation	50
9.6.3.1	<code>dp</code>	50
9.7	<code>prg_graph_mod</code> Module Reference	51
9.7.1	Detailed Description	51
9.7.2	Function/Subroutine Documentation	51
9.7.2.1	<code>prg_destroygraphpartitioning(gp)</code>	51
9.7.2.2	<code>prg_destroysubgraph(sg)</code>	52
9.7.2.3	<code>prg_equalgrouppartition(gp, hindex, ngroup, nodesPerPart, nnodes)</code>	53
9.7.2.4	<code>prg_equalpartition(gp, nodesPerPart, nnodes)</code>	54
9.7.2.5	<code>prg_filepartition(gp, partFile)</code>	54
9.7.2.6	<code>prg_fnormgraph(gp)</code>	55
9.7.2.7	<code>prg_initgraphpartitioning(gp, pname, np, nnodes, nnodes2)</code>	56
9.7.2.8	<code>prg_initsubgraph(sg, pnum, hsize)</code>	57

9.7.2.9	<code>prg_printgraphpartitioning(gp)</code>	58
9.7.2.10	<code>prg_readpart(gp, partFile)</code>	58
9.7.3	Variable Documentation	59
9.7.3.1	<code>dp</code>	59
9.8	<code>prg_graphsp2parser_mod</code> Module Reference	59
9.8.1	Detailed Description	60
9.8.2	Function/Subroutine Documentation	60
9.8.2.1	<code>prg_parse_gsp2(gsp2data, filename)</code>	60
9.8.3	Variable Documentation	60
9.8.3.1	<code>dp</code>	60
9.9	<code>prg_homolumo_mod</code> Module Reference	61
9.9.1	Detailed Description	61
9.9.2	Function/Subroutine Documentation	61
9.9.2.1	<code>prg_homolumogap(vv, imax, pp, mineval, maxeval, ehomo, elumo, egap, verbose)</code>	61
9.9.2.2	<code>prg_sp2sequence(pp, imax, mineval, maxeval, ehomo, elumo, errlimit, verbose)</code>	61
9.9.3	Variable Documentation	61
9.9.3.1	<code>dp</code>	61
9.10	<code>prg_implicit_fermi_mod</code> Module Reference	61
9.10.1	Function/Subroutine Documentation	62
9.10.1.1	<code>prg_implicit_fermi(h_bml, xi0_bml, p_bml, nsteps, nocc, mu, beta, osteps, occ↔ ErrLimit, threshold)</code>	62
9.10.2	Variable Documentation	62
9.10.2.1	<code>dp</code>	62
9.11	<code>prg_initmatrices_mod</code> Module Reference	62
9.11.1	Detailed Description	63
9.11.2	Function/Subroutine Documentation	63
9.11.2.1	<code>prg_init_hsmat(ham_bml, over_bml, bml_type, mdim, norb)</code>	63
9.11.2.2	<code>prg_init_ortho(orthoh_bml, orthop_bml, bml_type, mdim, norb)</code>	63
9.11.2.3	<code>prg_init_pzmat(rho_bml, zmat_bml, bml_type, mdim, norb)</code>	64
9.11.3	Variable Documentation	64
9.11.3.1	<code>dp</code>	64

9.12 prg_kernelparser_mod Module Reference	64
9.12.1 Detailed Description	65
9.12.2 Function/Subroutine Documentation	65
9.12.2.1 prg_check_valid(invalidc)	65
9.12.2.2 prg_parsing_kernel(keyvector_char, valvector_char, keyvector_int, valvector_int, keyvector_re, valvector_re, keyvector_log, valvector_log, filename, startstop) . .	66
9.12.3 Variable Documentation	67
9.12.3.1 dp	67
9.13 prg_nonortho_mod Module Reference	67
9.13.1 Detailed Description	68
9.13.2 Function/Subroutine Documentation	68
9.13.2.1 prg_deorthogonalize(orthoA_bml, zmat_bml, a_bml, threshold, bml_type, verbose)	68
9.13.2.2 prg_orthogonalize(A_bml, zmat_bml, orthoA_bml, threshold, bml_type, verbose)	69
9.13.3 Variable Documentation	69
9.13.3.1 dp	69
9.14 prg_normalize_mod Module Reference	69
9.14.1 Detailed Description	70
9.14.2 Function/Subroutine Documentation	70
9.14.2.1 prg_gershgorinreduction(gp)	70
9.14.2.2 prg_normalize(h_bml)	70
9.14.2.3 prg_normalize_cheb(h_bml, mu, emin, emax, alpha, scaledmu)	71
9.14.2.4 prg_normalize_fermi(h_bml, h1, hN, mu)	72
9.14.2.5 prg_normalize_implicit_fermi(h_bml, cnst, mu)	73
9.14.3 Variable Documentation	73
9.14.3.1 dp	73
9.15 prg_openfiles_mod Module Reference	74
9.15.1 Detailed Description	74
9.15.2 Function/Subroutine Documentation	74
9.15.2.1 get_file_unit(io_max)	74
9.15.2.2 prg_open_file(io, name)	75
9.15.2.3 prg_open_file_to_read(io, name)	76

9.16 prg_parallel_mod Module Reference	77
9.16.1 Detailed Description	78
9.16.2 Function/Subroutine Documentation	78
9.16.2.1 allgatherintparallel(sendBuf, sendLen, recvBuf, recvLen)	78
9.16.2.2 allgatherrealparallel(sendBuf, sendLen, recvBuf, recvLen)	79
9.16.2.3 allgathervintparallel(sendBuf, sendLen, recvBuf, recvLen, recvDispl)	79
9.16.2.4 allgathervrealparallel(sendBuf, sendLen, recvBuf, recvLen, recvDispl)	79
9.16.2.5 getmyrank()	79
9.16.2.6 getnranks()	80
9.16.2.7 isendparallel(sendBuf, sendLen, dest)	80
9.16.2.8 maxintparallel(sendBuf, recvBuf, icount)	81
9.16.2.9 maxrankrealparallel(sendBuf, recvBuf, icount)	81
9.16.2.10 maxrealparallel(sendBuf, recvBuf, icount)	81
9.16.2.11 minintparallel(sendBuf, recvBuf, icount)	81
9.16.2.12 minrankrealparallel(sendBuf, recvBuf, icount)	82
9.16.2.13 minrealparallel(sendBuf, recvBuf, icount)	82
9.16.2.14 prg_allgatherparallel(a)	82
9.16.2.15 prg_allsumintreduceparallel(buf, buflen)	83
9.16.2.16 prg_allsumrealreduceparallel(buf, buflen)	83
9.16.2.17 prg_barrierparallel()	84
9.16.2.18 prg_bcastparallel(buf, blen, root)	84
9.16.2.19 prg_initparallel()	84
9.16.2.20 prg_iprg_recvparallel(recvBuf, recvLen, rind)	84
9.16.2.21 prg_maxintreduce2(value1, value2)	85
9.16.2.22 prg_maxrealreduce(rvalue)	85
9.16.2.23 prg_minrealreduce(rvalue)	86
9.16.2.24 prg_recvparallel(recvBuf, recvLen)	86
9.16.2.25 prg_shutdownparallel()	86
9.16.2.26 prg_sumintreduce2(value1, value2)	87
9.16.2.27 prg_sumintreducen(valueVec, N)	87

9.16.2.28	<code>prg_sumrealreduce(value1)</code>	87
9.16.2.29	<code>prg_sumrealreduce2(value1, value2)</code>	88
9.16.2.30	<code>prg_sumrealreduce3(value1, value2, value3)</code>	88
9.16.2.31	<code>prg_sumrealreducen(valueVec, N)</code>	89
9.16.2.32	<code>prg_wait()</code>	89
9.16.2.33	<code>printrank()</code>	90
9.16.2.34	<code>saverequest(irequest)</code>	90
9.16.2.35	<code>sendparallel(sendBuf, sendLen, dest)</code>	91
9.16.2.36	<code>sendreceiveparallel(sendBuf, sendLen, dest, recvBuf, recvLen, source, nreceived)</code>	91
9.16.2.37	<code>sumintparallel(sendBuf, recvBuf, icount)</code>	91
9.16.2.38	<code>sumrealparallel(sendBuf, recvBuf, icount)</code>	92
9.16.3	Variable Documentation	92
9.16.3.1	<code>dp</code>	92
9.16.3.2	<code>ierr</code>	92
9.16.3.3	<code>myrank</code>	92
9.16.3.4	<code>nranks</code>	92
9.16.3.5	<code>reqcount</code>	93
9.16.3.6	<code>requestlist</code>	93
9.16.3.7	<code>rused</code>	93
9.17	<code>prg_partition_mod</code> Module Reference	93
9.17.1	Detailed Description	94
9.17.2	Function/Subroutine Documentation	94
9.17.2.1	<code>prg_accept_prob(it, prg_delta, r)</code>	94
9.17.2.2	<code>prg_check_arrays(gp, core_count, CH_count, Halo_count)</code>	95
9.17.2.3	<code>prg_costindex(cost, sumCubes, maxCH, smooth_maxCH, obj_fun)</code>	95
9.17.2.4	<code>prg_costpartition(gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)</code>	96
9.17.2.5	<code>prg_find_best_move(gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, best_node, best_part)</code>	97
9.17.2.6	<code>prg_get_largest_hedge_in_part(gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, search_part, largest_Hedge)</code>	98

9.17.2.7	<code>prg_kernlin(gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, nconverg, seed)</code>	98
9.17.2.8	<code>prg_kernlin2(gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)</code>	100
9.17.2.9	<code>prg_kernlin_queue(gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)</code>	101
9.17.2.10	<code>prg_metispartition(gp, ngroups, nnodes, xadj, adjncy, nparts, part, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)</code>	102
9.17.2.11	<code>prg_rand_node(gp, node, seed)</code>	103
9.17.2.12	<code>prg_rand_shuffle(array, seed)</code>	104
9.17.2.13	<code>prg_simannealing(gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, niter, seed)</code>	104
9.17.2.14	<code>prg_simannealing_old(gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, niter, seed)</code>	105
9.17.2.15	<code>prg_update_gp(gp, partNumber, core_count)</code>	106
9.17.2.16	<code>update_prg_costpartition(gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, node, new_part)</code>	107
9.17.3	Variable Documentation	108
9.17.3.1	<code>dp</code>	108
9.17.3.2	<code>metis_index_kind</code>	108
9.17.3.3	<code>metis_real_kind</code>	108
9.18	<code>prg_progress_mod</code> Module Reference	108
9.18.1	Detailed Description	109
9.18.2	Function/Subroutine Documentation	109
9.18.2.1	<code>prg_progress_init()</code>	109
9.18.2.2	<code>prg_progress_shutdown()</code>	110
9.18.3	Variable Documentation	110
9.18.3.1	<code>dp</code>	110
9.19	<code>prg_ptable_mod</code> Module Reference	110
9.19.1	Detailed Description	112
9.19.2	Function/Subroutine Documentation	113
9.19.2.1	<code>element_atomic_number(symbol)</code>	113
9.19.2.2	<code>element_atomic_number_upper(symbol)</code>	113

9.19.3	Variable Documentation	113
9.19.3.1	atom_en	113
9.19.3.2	dp	114
9.19.3.3	element_covr	114
9.19.3.4	element_ea	114
9.19.3.5	element_econf	114
9.19.3.6	element_ip	115
9.19.3.7	element_mass	115
9.19.3.8	element_maxbonds	115
9.19.3.9	element_name	115
9.19.3.10	element_numel	116
9.19.3.11	element_symbol	116
9.19.3.12	element_symbol_upper	116
9.19.3.13	element_vdwr	116
9.19.3.14	nz	116
9.20	prg_pulaycomponent_mod Module Reference	117
9.20.1	Detailed Description	117
9.20.2	Function/Subroutine Documentation	117
9.20.2.1	prg_get_pulayforce(nats, zmat_bml, ham_bml, rho_bml, dSx_bml, dSy_bml, d←Sz_bml, hindex, FPUL, threshold)	117
9.20.2.2	prg_pulaycomponent0(rho_bml, ham_bml, pcm_bml, threshold, M, bml_type, verbose)	118
9.20.2.3	prg_pulaycomponentt(rho_bml, ham_bml, zmat_bml, pcm_bml, threshold, M, bml_type, verbose)	118
9.20.3	Variable Documentation	118
9.20.3.1	dp	118
9.21	prg_pulaymixer_mod Module Reference	119
9.21.1	Detailed Description	119
9.21.2	Function/Subroutine Documentation	119
9.21.2.1	prg_linearmixer(charges, oldcharges, scferror, linmixcoef, verbose)	119
9.21.2.2	prg_parse_mixer(input, filename)	120

9.21.2.3	<code>prg_qmixer(charges, oldcharges, dqin, dqout, scferror, piter, pulaycoef, mpulay, verbose)</code>	120
9.21.3	Variable Documentation	120
9.21.3.1	<code>dp</code>	120
9.22	<code>prg_quantumdynamics_mod</code> Module Reference	121
9.22.1	Detailed Description	121
9.22.2	Function/Subroutine Documentation	122
9.22.2.1	<code>prg_excitation(fill_mat, orbit_orig, orbit_exci)</code>	122
9.22.2.2	<code>prg_get_sparsity_cplxmat(matrix_type, element_type, thresh, a_dense)</code>	122
9.22.2.3	<code>prg_get_sparsity_realmat(matrix_type, element_type, thresh, a_dense)</code>	122
9.22.2.4	<code>prg_getcharge(rho_bml, s_bml, charges, aux_bml, z, spindex, N, nats, thresh)</code>	122
9.22.2.5	<code>prg_getdipole(charges, r, mu)</code>	123
9.22.2.6	<code>prg_kick_density(kick_direct, kick_mag, dens, norbs, mdim, S, SINV, which_atom, r, bmltype, thresh)</code>	123
9.22.2.7	<code>prg_kick_density_bml(kick_direct, kick_mag, rho_bml, s_bml, sinv_bml, mdim, which_atom, r, matrix_type, thresh)</code>	124
9.22.2.8	<code>prg_lvni_bml(h1_bml, sinv_bml, dt, hbar, rhoold_bml, rho_bml, aux_bml, matrix_type, mdim, thresh)</code>	124
9.22.3	Variable Documentation	125
9.22.3.1	<code>dp</code>	125
9.23	<code>prg_response_mod</code> Module Reference	125
9.23.1	Detailed Description	126
9.23.2	Function/Subroutine Documentation	126
9.23.2.1	<code>prg_compute_dipole(charges, coordinate, dipoleMoment, factor, verbose)</code>	126
9.23.2.2	<code>prg_compute_polarizability(rsp_bml, prt_bml, polarizability, factor, verbose)</code>	127
9.23.2.3	<code>prg_compute_response_fd(ham_bml, prt_bml, rsp_bml, prg_delta, bndfil, threshold, verbose)</code>	127
9.23.2.4	<code>prg_compute_response_rs(ham_bml, prt_bml, rsp_bml, lambda, bndfil, threshold, verbose)</code>	127
9.23.2.5	<code>prg_compute_response_sp2(ham_bml, prt_bml, rsp_bml, rho_bml, lambda, bndfil, minsp2iter, maxsp2iter, sp2conv, idemt看, threshold, verbose)</code>	128
9.23.2.6	<code>prg_parse_response(RespData, filename)</code>	128
9.23.2.7	<code>prg_pert_constant_field(field, intensity, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)</code>	128

9.23.2.8	<code>prg_pert_cos_pot(direction, lx, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)</code>	129
9.23.2.9	<code>prg_pert_from_file(prt_bml, norb)</code>	129
9.23.2.10	<code>prg_pert_sin_pot(direction, lx, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)</code>	130
9.23.2.11	<code>prg_project_response(rsp_bml, over_bml, spindex, norbi, coordinates, rspfunc, verbose)</code>	130
9.23.2.12	<code>prg_write_dipole_tcl(dipoleMoment, file, factor, verbose)</code>	131
9.23.3	Variable Documentation	131
9.23.3.1	<code>dp</code>	131
9.23.3.2	<code>pi</code>	131
9.24	<code>prg_sp2_fermi_mod</code> Module Reference	131
9.24.1	Detailed Description	132
9.24.2	Function/Subroutine Documentation	132
9.24.2.1	<code>absmaxderivative(func, de)</code>	132
9.24.2.2	<code>prg_sp2_entropy_function(mu, h1, hN, nsteps, sgnlist, GG, ee)</code>	133
9.24.2.3	<code>prg_sp2_fermi(h_bml, osteps, nsteps, nocc, mu, beta, h1, hN, sgnlist, threshold, eps, traceLimit, x_bml)</code>	133
9.24.2.4	<code>prg_sp2_fermi_init(h_bml, nsteps, nocc, tscale, threshold, occErrLimit, traceLimit, x_bml, mu, beta, h1, hN, sgnlist)</code>	134
9.24.2.5	<code>prg_sp2_fermi_init_norecs(h_bml, nsteps, nocc, tscale, threshold, occErrLimit, traceLimit, x_bml, mu, beta, h1, hN, sgnlist, verbose)</code>	135
9.24.2.6	<code>sp2_entropy_ts(D0_bml, GG, ee)</code>	136
9.24.2.7	<code>sp2_inverse(f, mu, h1, hN, nsteps, sgnlist)</code>	136
9.24.3	Variable Documentation	137
9.24.3.1	<code>dp</code>	137
9.25	<code>prg_sp2_mod</code> Module Reference	137
9.25.1	Detailed Description	138
9.25.2	Function/Subroutine Documentation	138
9.25.2.1	<code>prg_prg_sp2_alg1_seq_inplace(rho_bml, threshold, pp, icount, vv, mineval, maxeval)</code>	138
9.25.2.2	<code>prg_prg_sp2_alg2_seq_inplace(rho_bml, threshold, pp, icount, vv, mineval, maxeval, verbose)</code>	138

9.25.2.3	<code>prg_sp2_alg1(h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)</code>	139
9.25.2.4	<code>prg_sp2_alg1_genseq(h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, pp, icount, vv)</code>	139
9.25.2.5	<code>prg_sp2_alg1_seq(h_bml, rho_bml, threshold, pp, icount, vv)</code>	140
9.25.2.6	<code>prg_sp2_alg2(h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)</code>	140
9.25.2.7	<code>prg_sp2_alg2_genseq(h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, pp, icount, vv, verbose)</code>	141
9.25.2.8	<code>prg_sp2_alg2_seq(h_bml, rho_bml, threshold, pp, icount, vv, verbose)</code>	141
9.25.2.9	<code>prg_sp2_basic(h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)</code>	142
9.25.2.10	<code>prg_sp2_submatrix(ham_bml, rho_bml, threshold, pp, icount, vv, mineval, maxeval, core_size)</code>	142
9.25.2.11	<code>prg_sp2_submatrix_inplace(rho_bml, threshold, pp, icount, vv, mineval, maxeval, core_size)</code>	143
9.25.3	Variable Documentation	143
9.25.3.1	<code>dp</code>	143
9.26	<code>prg_sp2parser_mod</code> Module Reference	143
9.26.1	Detailed Description	144
9.26.2	Function/Subroutine Documentation	144
9.26.2.1	<code>prg_parse_sp2(sp2data, filename)</code>	144
9.26.3	Variable Documentation	144
9.26.3.1	<code>dp</code>	144
9.27	<code>prg_subgraphloop_mod</code> Module Reference	144
9.27.1	Detailed Description	145
9.27.2	Function/Subroutine Documentation	145
9.27.2.1	<code>prg_balanceparts(gp)</code>	145
9.27.2.2	<code>prg_collectmatrixfromparts(gp, rho_bml)</code>	146
9.27.2.3	<code>prg_getgrouppartitionhalosfromgraph(gp, g_bml, hnode, djflag)</code>	146
9.27.2.4	<code>prg_getpartitionhalosfromgraph(gp, g_bml, djflag)</code>	147
9.27.2.5	<code>prg_partordering(gp)</code>	147
9.27.2.6	<code>prg_subgraphsp2loop(h_bml, g_bml, rho_bml, gp, threshold)</code>	147

9.27.3	Variable Documentation	148
9.27.3.1	dp	148
9.28	prg_syrotation_mod Module Reference	148
9.28.1	Detailed Description	149
9.28.2	Function/Subroutine Documentation	149
9.28.2.1	prg_parse_rotation(rot, filename)	149
9.28.2.2	prg_rotate(rot, r, verbose)	149
9.28.3	Variable Documentation	150
9.28.3.1	dp	150
9.29	prg_system_mod Module Reference	150
9.29.1	Detailed Description	152
9.29.2	Function/Subroutine Documentation	152
9.29.2.1	prg_adj2bml(xadj, adjncy, bml_type, g_bml)	152
9.29.2.2	prg_centeratbox(coords, lattice_vectors, verbose)	152
9.29.2.3	prg_collect_graph_p(rho_bml, nc, nats, hindex, chindex, graph_p, threshold, mdimin, verbose)	153
9.29.2.4	prg_destroy_subsystems(sbsy, verbose)	153
9.29.2.5	prg_get_covgraph(sy, nnStructMindist, nnStruct, nrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)	153
9.29.2.6	prg_get_covgraph_h(sy, nnStructMindist, nnStruct, nrnnstruct, rcut, graph_↔h, mdimin, verbose)	154
9.29.2.7	prg_get_covgraph_int(sy, nnStructMindist, nnStruct, nrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)	154
9.29.2.8	prg_get_dihedral(coords, id1, id2, id3, id4, dihedral)	154
9.29.2.9	prg_get_distancematrix(coords, dmat)	155
9.29.2.10	prg_get_nameandext(fullfilename, filename, ext)	155
9.29.2.11	prg_get_origin(coords, origin)	155
9.29.2.12	prg_get_partial_atomgraph(rho_bml, hindex, gch_bml, threshold, verbose)	156
9.29.2.13	prg_get_recip_vects(lattice_vectors, recip_vectors, volr, volk)	156
9.29.2.14	prg_get_subsystem(sy, lsize, indices, sbsy, verbose)	156
9.29.2.15	prg_graph2bml(graph, bml_type, g_bml)	157
9.29.2.16	prg_graph2vector(graph, vector, maxnz)	157

9.29.2.17	<code>prg_make_random_system(system, nats, seed, lx, ly, lz)</code>	157
9.29.2.18	<code>prg_merge_graph(graph_p, graph_h)</code>	158
9.29.2.19	<code>prg_merge_graph_adj(graph_p, graph_h, xadj, adjncy)</code>	158
9.29.2.20	<code>prg_molpartition(sy, npart, nnStructMindist, nnStruct, nrrnstruct, hetatm, gp, verbose)</code>	158
9.29.2.21	<code>prg_parameters_to_vectors(abc_angles, lattice_vector)</code>	159
9.29.2.22	<code>prg_parse_system(system, filename, extin)</code>	159
9.29.2.23	<code>prg_replicate(coords, symbols, lattice_vectors, nx, ny, nz)</code>	160
9.29.2.24	<code>prg_sortadj(xadj, adjncy)</code>	161
9.29.2.25	<code>prg_translateandfoldtobox(coords, lattice_vectors, origin, verbose)</code>	161
9.29.2.26	<code>prg_translatetogeomcandfoldtobox(coords, lattice_vectors, origin)</code>	161
9.29.2.27	<code>prg_vector2graph(vector, graph, maxnz)</code>	161
9.29.2.28	<code>prg_vectors_to_parameters(lattice_vector, abc_angles)</code>	162
9.29.2.29	<code>prg_wraparound(coords, lattice_vectors, index, verbose)</code>	162
9.29.2.30	<code>prg_write_system(system, filename, extension)</code>	163
9.29.2.31	<code>prg_write_trajectory(system, iter, each, prg_deltat, filename, extension)</code>	163
9.29.2.32	<code>prg_write_trajectoryandproperty(system, iter, each, prg_deltat, scalarprop, filename, extension)</code>	164
9.29.3	Variable Documentation	165
9.29.3.1	<code>dp</code>	165
9.30	<code>prg_timer_mod</code> Module Reference	165
9.30.1	Detailed Description	166
9.30.2	Function/Subroutine Documentation	166
9.30.2.1	<code>int2char(ival)</code>	166
9.30.2.2	<code>prg_print_date_and_time(tag)</code>	167
9.30.2.3	<code>prg_timer_collect()</code>	167
9.30.2.4	<code>prg_timer_getid()</code>	168
9.30.2.5	<code>prg_timer_results()</code>	168
9.30.2.6	<code>prg_timer_shutdown()</code>	168
9.30.2.7	<code>prg_timer_start(itimer, tag)</code>	169
9.30.2.8	<code>prg_timer_stop(itimer, verbose)</code>	169

9.30.2.9	time2milliseconds()	170
9.30.2.10	timer_prg_init()	170
9.30.3	Variable Documentation	170
9.30.3.1	badd_timer	170
9.30.3.2	bmult_timer	171
9.30.3.3	buildz_timer	171
9.30.3.4	deortho_timer	171
9.30.3.5	dp	171
9.30.3.6	dyn_timer	171
9.30.3.7	genx_timer	171
9.30.3.8	graphsp2_timer	171
9.30.3.9	halfverlet_timer	171
9.30.3.10	loop_timer	171
9.30.3.11	mdloop_timer	171
9.30.3.12	nlist_timer	172
9.30.3.13	num_timers	172
9.30.3.14	ortho_timer	172
9.30.3.15	pairpot_timer	172
9.30.3.16	part_timer	172
9.30.3.17	pos_timer	172
9.30.3.18	ptimer	172
9.30.3.19	realcoul_timer	172
9.30.3.20	recipcoul_timer	172
9.30.3.21	sp2_timer	172
9.30.3.22	suball_timer	173
9.30.3.23	subext_timer	173
9.30.3.24	subgraph_timer	173
9.30.3.25	subind_timer	173
9.30.3.26	subsp2_timer	173
9.30.3.27	tclock_max	173

9.30.3.28	<code>tclock_rate</code>	173
9.30.3.29	<code>tstart_clock</code>	173
9.30.3.30	<code>tstop_clock</code>	173
9.30.3.31	<code>zdiag_timer</code>	173
9.31	<code>prg_xlbo_mod</code> Module Reference	174
9.31.1	Detailed Description	174
9.31.2	Function/Subroutine Documentation	175
9.31.2.1	<code>prg_parse_xlbo(xlbo, filename)</code>	175
9.31.2.2	<code>prg_xlbo_fcoulupdate(fcoul, charges, n)</code>	175
9.31.2.3	<code>prg_xlbo_nint(charges, n, n_0, n_1, n_2, n_3, n_4, n_5, mdstep, xl)</code>	175
9.31.3	Variable Documentation	175
9.31.3.1	<code>alpha</code>	176
9.31.3.2	<code>c0</code>	176
9.31.3.3	<code>c1</code>	176
9.31.3.4	<code>c2</code>	176
9.31.3.5	<code>c3</code>	176
9.31.3.6	<code>c4</code>	176
9.31.3.7	<code>c5</code>	176
9.31.3.8	<code>cc</code>	176
9.31.3.9	<code>dp</code>	176
9.31.3.10	<code>kappa</code>	176
9.32	<code>prg_xlkernel_mod</code> Module Reference	177
9.32.1	Detailed Description	177
9.32.2	Function/Subroutine Documentation	177
9.32.2.1	<code>prg_eig(A, Q, ee, type, HDIM)</code>	177
9.32.2.2	<code>prg_fermi(D0, QQ, ee, gap, Fe_vec, mu0, H, Z, Nocc, T, OccErrLim, MaxIt, HDIM)</code>	178
9.32.2.3	<code>prg_get_deriv_finite_temp(P1, H0, H1, Nocc, T, Q, ev, fe, mu0, eps, HDIM)</code>	178
9.32.2.4	<code>prg_inv(X, XI, HDIM)</code>	178
9.32.2.5	<code>prg_kernel_fermi_full(KK, JJ, D0, mu0, mu1, T, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_atoms, MaxIt, eps, m, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERATIO, nnRx, nnRy, nnRz, nrrnlist, nnType, H_INDEX_START, H_INDEX_END, H, S, Z, Nocc, Znuc, QQ, ee, Fe_vec)</code>	178
9.32.2.6	<code>prg_mmult(alpha, A, B, beta, C, TA, TB, HDIM)</code>	178
9.32.2.7	<code>prg_parse_xlkernel(input, filename)</code>	179
9.32.2.8	<code>prg_rank1(verbose)</code>	179
9.32.2.9	<code>prg_v_kernel_fermi(D0, dq_dv, v, mu0, mu1, T, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_atoms, MaxIt, eps, m, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERATIO, nnRx, nnRy, nnRz, nrrnlist, nnType, H_INDEX_START, H_INDEX_END, H, S, Z, Nocc, Znuc, QQ, ee, Fe_vec)</code>	179
9.32.3	Variable Documentation	180
9.32.3.1	<code>dp</code>	180

10 Data Type Documentation	181
10.1 prg_chebyshev_mod::chebdata_type Type Reference	181
10.1.1 Detailed Description	181
10.1.2 Member Data Documentation	182
10.1.2.1 atr	182
10.1.2.2 bml_type	182
10.1.2.3 bndfil	182
10.1.2.4 ef	182
10.1.2.5 estep	182
10.1.2.6 fermitol	182
10.1.2.7 flavor	182
10.1.2.8 getef	182
10.1.2.9 jobname	182
10.1.2.10 jon	182
10.1.2.11 kbt	183
10.1.2.12 mdim	183
10.1.2.13 ncoeffs	183
10.1.2.14 ndim	183
10.1.2.15 npts	183
10.1.2.16 threshold	183
10.1.2.17 trkfunc	183
10.1.2.18 verbose	183
10.2 prg_system_mod::estruct_type Type Reference	183
10.2.1 Detailed Description	184
10.2.2 Member Data Documentation	184
10.2.2.1 coul_pot_k	184
10.2.2.2 coul_pot_r	185
10.2.2.3 eband	185
10.2.2.4 fpul	185
10.2.2.5 fscoul	185

10.2.2.6	ham	185
10.2.2.7	ham0	185
10.2.2.8	hindex	185
10.2.2.9	nel	185
10.2.2.10	norbs	186
10.2.2.11	oham	186
10.2.2.12	orho	186
10.2.2.13	over	186
10.2.2.14	rho	186
10.2.2.15	skforce	186
10.2.2.16	zmat	186
10.3	prg_genz_mod::genzspdata Type Reference	187
10.3.1	Detailed Description	187
10.3.2	Member Function/Subroutine Documentation	187
10.3.2.1	prg_allocatezspmat	187
10.3.2.2	prg_generate	187
10.3.2.3	prg_init	187
10.3.3	Member Data Documentation	187
10.3.3.1	integration	187
10.3.3.2	nfirst	188
10.3.3.3	nreff	188
10.3.3.4	nrefi	188
10.3.3.5	numthresf	188
10.3.3.6	numthresi	188
10.3.3.7	verbose	188
10.4	prg_genz_mod::genzspinp Type Reference	188
10.4.1	Detailed Description	189
10.4.2	Member Data Documentation	189
10.4.2.1	bml_type	189
10.4.2.2	igenz	189

10.4.2.3	integration	190
10.4.2.4	mdim	190
10.4.2.5	nfirst	190
10.4.2.6	nreff	190
10.4.2.7	nrefi	190
10.4.2.8	numthresf	190
10.4.2.9	numthresi	190
10.4.2.10	verbose	190
10.4.2.11	zsp	191
10.5	prg_graph_mod::graph_partitioning_t Type Reference	191
10.5.1	Detailed Description	192
10.5.2	Member Data Documentation	192
10.5.2.1	ehomo	192
10.5.2.2	elumo	193
10.5.2.3	globalpartextent	193
10.5.2.4	globalpartmax	193
10.5.2.5	globalpartmin	193
10.5.2.6	localpartextent	193
10.5.2.7	localpartmax	193
10.5.2.8	localpartmin	193
10.5.2.9	maxeval	193
10.5.2.10	maxiter	194
10.5.2.11	mineval	194
10.5.2.12	myrank	194
10.5.2.13	nnodesinpart	194
10.5.2.14	nnodesinpartall	194
10.5.2.15	nparts	194
10.5.2.16	order	194
10.5.2.17	pname	194
10.5.2.18	pp	195

10.5.2.19 reorder	195
10.5.2.20 sgraph	195
10.5.2.21 totalnodes	195
10.5.2.22 totalnodes2	195
10.5.2.23 totalparts	195
10.5.2.24 totalprocs	195
10.5.2.25 vv	196
10.6 prg_graphsp2parser_mod::gsp2data_type Type Reference	196
10.6.1 Detailed Description	196
10.6.2 Member Data Documentation	197
10.6.2.1 bml_type	197
10.6.2.2 bndfil	197
10.6.2.3 covgfact	197
10.6.2.4 double_jump	197
10.6.2.5 errlimit	197
10.6.2.6 graph_element	197
10.6.2.7 gthreshold	197
10.6.2.8 hamfile	197
10.6.2.9 jobname	197
10.6.2.10 maxsp2iter	197
10.6.2.11 mdim	198
10.6.2.12 minsp2iter	198
10.6.2.13 natoms	198
10.6.2.14 ndim	198
10.6.2.15 nlgcut	198
10.6.2.16 nodesperpart	198
10.6.2.17 parteach	198
10.6.2.18 partition_count	198
10.6.2.19 partition_refinement	198
10.6.2.20 partition_type	198

10.6.2.21	<code>pdim</code>	199
10.6.2.22	<code>sdim</code>	199
10.6.2.23	<code>sp2conv</code>	199
10.6.2.24	<code>sp2tol</code>	199
10.6.2.25	<code>threshold</code>	199
10.6.2.26	<code>verbose</code>	199
10.7	<code>prg_pulaymixer_mod::mx_type</code> Type Reference	199
10.7.1	Detailed Description	200
10.7.2	Member Data Documentation	200
10.7.2.1	<code>mixcoeff</code>	200
10.7.2.2	<code>mixeron</code>	200
10.7.2.3	<code>mixertype</code>	200
10.7.2.4	<code>mpulay</code>	200
10.7.2.5	<code>verbose</code>	200
10.8	<code>prg_extras_mod::prg_memory_consumption</code> Interface Reference	200
10.8.1	Detailed Description	201
10.8.2	Constructor & Destructor Documentation	201
10.8.2.1	<code>prg_memory_consumption(vm_peak, vm_size, pid, ppid)</code>	201
10.9	<code>prg_parallel_mod::rankreducedata_t</code> Type Reference	201
10.9.1	Detailed Description	201
10.9.2	Member Data Documentation	201
10.9.2.1	<code>rank</code>	201
10.9.2.2	<code>val</code>	202
10.10	<code>prg_response_mod::respdata_type</code> Type Reference	202
10.10.1	Detailed Description	202
10.10.2	Member Data Documentation	202
10.10.2.1	<code>bmltype</code>	202
10.10.2.2	<code>computedipole</code>	202
10.10.2.3	<code>field</code>	202
10.10.2.4	<code>fieldintensity</code>	202

10.10.2.5 getresponse	203
10.10.2.6 mdim	203
10.10.2.7 numthresh	203
10.10.2.8 respmode	203
10.10.2.9 typeofpert	203
10.11prg_syrotation_mod::rotation_type Type Reference	203
10.11.1 Detailed Description	204
10.11.2 Member Data Documentation	204
10.11.2.1 catom	204
10.11.2.2 catom2	204
10.11.2.3 jobname	204
10.11.2.4 patom1	204
10.11.2.5 patom2	204
10.11.2.6 pq1	204
10.11.2.7 pq2	205
10.11.2.8 rotate_atoms	205
10.11.2.9 typeofrot	205
10.11.2.10v1	205
10.11.2.11v2	205
10.11.2.12vq	205
10.12prg_sp2parser_mod::sp2data_type Type Reference	205
10.12.1 Detailed Description	206
10.12.2 Member Data Documentation	206
10.12.2.1 bml_type	206
10.12.2.2 bndfil	206
10.12.2.3 flavor	206
10.12.2.4 jobname	206
10.12.2.5 maxsp2iter	206
10.12.2.6 mdim	207
10.12.2.7 minsp2iter	207

10.12.2.8 ndim	207
10.12.2.9 pdim	207
10.12.2.10sdim	207
10.12.2.11sp2conv	207
10.12.2.12sp2tol	207
10.12.2.13threshold	207
10.12.2.14verbose	207
10.13prg_graph_mod::subgraph_t Type Reference	208
10.13.1 Detailed Description	208
10.13.2 Member Data Documentation	208
10.13.2.1 core_halo_index	208
10.13.2.2 hsize	208
10.13.2.3 lsize	208
10.13.2.4 lsize	209
10.13.2.5 nodeinpart	209
10.13.2.6 part	209
10.14prg_system_mod::system_type Type Reference	209
10.14.1 Detailed Description	210
10.14.2 Member Data Documentation	211
10.14.2.1 atomic_number	211
10.14.2.2 coordinate	211
10.14.2.3 estr	211
10.14.2.4 force	211
10.14.2.5 lattice_vector	211
10.14.2.6 mass	212
10.14.2.7 nats	212
10.14.2.8 net_charge	212
10.14.2.9 nsp	212
10.14.2.10recip_vector	212
10.14.2.11resindex	213

10.14.2.12 <code>spatnum</code>	213
10.14.2.13 <code>spindex</code>	213
10.14.2.14 <code>splist</code>	213
10.14.2.15 <code>spmass</code>	213
10.14.2.16 <code>symbol</code>	214
10.14.2.17 <code>userdef</code>	214
10.14.2.18 <code>velocity</code>	214
10.14.2.19 <code>volk</code>	214
10.14.2.20 <code>volr</code>	214
10.15 <code>prg_timer_mod::timer_status_t</code> Type Reference	215
10.15.1 Detailed Description	215
10.15.2 Member Data Documentation	215
10.15.2.1 <code>maxrank</code>	215
10.15.2.2 <code>maxvalue</code>	216
10.15.2.3 <code>minrank</code>	216
10.15.2.4 <code>minvalue</code>	216
10.15.2.5 <code>tavg</code>	216
10.15.2.6 <code>tcount</code>	216
10.15.2.7 <code>tname</code>	216
10.15.2.8 <code>tpercent</code>	216
10.15.2.9 <code>tstart</code>	216
10.15.2.10 <code>tstdev</code>	217
10.15.2.11 <code>tsum</code>	217
10.15.2.12 <code>total</code>	217
10.16 <code>prg_extras_mod::to_string</code> Interface Reference	217
10.16.1 Detailed Description	217
10.16.2 Member Function/Subroutine Documentation	217
10.16.2.1 <code>to_string_double(x)</code>	217
10.16.2.2 <code>to_string_integer(i)</code>	218
10.16.2.3 <code>to_string_long_long(i)</code>	218

10.17prg_xlbo_mod::xlbo_type Type Reference	218
10.17.1 Detailed Description	219
10.17.2 Member Data Documentation	219
10.17.2.1 cc	219
10.17.2.2 jobname	219
10.17.2.3 maxscfinititer	219
10.17.2.4 maxscfiter	219
10.17.2.5 minit	220
10.17.2.6 threshold	220
10.17.2.7 verbose	220
10.18prg_xlkernel_mod::xlk_type Type Reference	220
10.18.1 Detailed Description	220
10.18.2 Member Data Documentation	220
10.18.2.1 kerneltype	220
10.18.2.2 nrank	220
10.18.2.3 scalecoeff	221
10.18.2.4 verbose	221
11 File Documentation	223
11.1 /home/christian/qmd-progress/README.md File Reference	223
11.2 /home/christian/qmd-progress/tests/README.md File Reference	223
11.3 /home/christian/qmd-progress/src/prg_charges_mod.F90 File Reference	223
11.4 /home/christian/qmd-progress/src/prg_chebyshev_mod.F90 File Reference	223
11.5 /home/christian/qmd-progress/src/prg_densitymatrix_mod.F90 File Reference	224
11.6 /home/christian/qmd-progress/src/prg_dos_mod.F90 File Reference	225
11.7 /home/christian/qmd-progress/src/prg_doxy_mod.F90 File Reference	226
11.8 /home/christian/qmd-progress/src/prg_extras_mod.F90 File Reference	226
11.9 /home/christian/qmd-progress/src/prg_genz_mod.F90 File Reference	226
11.10/home/christian/qmd-progress/src/prg_graph_mod.F90 File Reference	227
11.11/home/christian/qmd-progress/src/prg_graphsp2parser_mod.F90 File Reference	228
11.12/home/christian/qmd-progress/src/prg_homolumo_mod.F90 File Reference	229

11.13/home/christian/qmd-progress/src/prg_implicit_fermi_mod.F90 File Reference	229
11.14/home/christian/qmd-progress/src/prg_initmatrices_mod.F90 File Reference	229
11.15/home/christian/qmd-progress/src/prg_kernelparser_mod.F90 File Reference	230
11.16/home/christian/qmd-progress/src/prg_nonortho_mod.F90 File Reference	230
11.17/home/christian/qmd-progress/src/prg_normalize_mod.F90 File Reference	231
11.18/home/christian/qmd-progress/src/prg_openfiles_mod.F90 File Reference	231
11.19/home/christian/qmd-progress/src/prg_parallel_mod.F90 File Reference	232
11.20/home/christian/qmd-progress/src/prg_partition_mod.F90 File Reference	233
11.21/home/christian/qmd-progress/src/prg_progress_mod.F90 File Reference	234
11.22/home/christian/qmd-progress/src/prg_ptable_mod.F90 File Reference	234
11.23/home/christian/qmd-progress/src/prg_pulaycomponent_mod.F90 File Reference	237
11.24/home/christian/qmd-progress/src/prg_pulaymixer_mod.F90 File Reference	237
11.25/home/christian/qmd-progress/src/prg_quantumdynamics_mod.F90 File Reference	238
11.26/home/christian/qmd-progress/src/prg_response_mod.F90 File Reference	239
11.27/home/christian/qmd-progress/src/prg_sp2_fermi_mod.F90 File Reference	240
11.28/home/christian/qmd-progress/src/prg_sp2_mod.F90 File Reference	241
11.29/home/christian/qmd-progress/src/prg_sp2parser_mod.F90 File Reference	242
11.30/home/christian/qmd-progress/src/prg_subgraphloop_mod.F90 File Reference	242
11.31/home/christian/qmd-progress/src/prg_syrotation_mod.F90 File Reference	243
11.32/home/christian/qmd-progress/src/prg_system_mod.F90 File Reference	243
11.33/home/christian/qmd-progress/src/prg_timer_mod.F90 File Reference	245
11.34/home/christian/qmd-progress/src/prg_xlbo_mod.F90 File Reference	246
11.35/home/christian/qmd-progress/src/prg_xlkernel_mod.F90 File Reference	247
 Bibliography	 249
 Index	 251

Chapter 1

README

A library for quantum chemistry solvers.

PROGRESS: Parallel, Rapid $O(N)$ and Graph-based Recursive Electronic Structure Solver. **LA-CC-16-068**

- This library is focused on the development of general solvers that are commonly used in *quantum chemistry packages*.
- This library has to be compiled with the *Basic Matrix Library* (BML).
- Our webpage can be found at <https://lanl.github.io/qmd-progress/>

Authors

(in alphabetical order)

- Anders M. N. Niklasson amn@lanl.gov
- Christian F. A. Negre cnegre@lanl.gov
- Marc J. Cawkwell cawkwell@lanl.gov
- Nicolas Bock nicolasbock@gmail.com
- Susan M. Mniszewski smm@lanl.gov
- Michael E. Wall mewall@lanl.gov

Contributors

- Jesse Grindstaff grindstaff@lanl.gov
- Alicia Welden welden@umich.edu

Build Dependencies

- `>=OpenMP-3.1`
- `>=metis-5.0` if building with `PROGRESS_GRAPHLIB`

(On some distributions, metis is available as a package. Make sure you install the `-dev` package. For example, Ubuntu requires `libmetis-dev`.)

Build and Install Instructions

How to build

```
$ CMAKE_PREFIX_PATH=<BML install path> ./build.sh
```

How to install

```
$ cd build
$ sudo make install
```

To specify the Intel Fortran compiler:

```
$ FC=ifort PKG_CONFIG_PATH=<BML install path>/lib/pkgconfig ./build.sh
```

To build with the gfortran compiler and OpenMP:

```
$ CC=gcc FC=gfortran \
  CMAKE_BUILD_TYPE=Release \
  PROGRESS_OPENMP=yes \
  CMAKE_PREFIX_PATH=<BML install path> \
  CMAKE_INSTALL_PREFIX=<PROGRESS install path> \
  ./build.sh configure
```

To build with OpenMP, MPI and testing enabled:

```
$ CC=mpicc FC=mpif90 \
  CMAKE_BUILD_TYPE=Release \
  PROGRESS_OPENMP=yes \
  PROGRESS_MPI=yes \
  PROGRESS_TESTING=yes \
  CMAKE_PREFIX_PATH=<BML install path> \
  CMAKE_INSTALL_PREFIX=<PROGRESS install path> \
  ./build.sh configure
```

To build with OpenMP, MPI, testing enabled and example programs built:

```
$ CC=mpicc FC=mpif90 \
  CMAKE_BUILD_TYPE=Release \
  PROGRESS_OPENMP=yes \
  PROGRESS_MPI=yes \
  PROGRESS_TESTING=yes \
  PROGRESS_EXAMPLES=yes \
  CMAKE_PREFIX_PATH=<BML install path> \
  CMAKE_INSTALL_PREFIX=<PROGRESS install path> \
  ./build.sh configure
```

To build with OpenMP and MPI and testing enabled and example programs built and the METIS graph partitioning library:

```
$ CC=mpicc FC=mpif90 \
  CMAKE_BUILD_TYPE=Release \
  PROGRESS_OPENMP=yes \
  PROGRESS_MPI=yes \
  PROGRESS_GRAPHLIB=yes \
  PROGRESS_TESTING=yes \
  PROGRESS_EXAMPLES=yes \
  CMAKE_PREFIX_PATH=<BML install path> \
  CMAKE_INSTALL_PREFIX=<PROGRESS install path> \
  ./build.sh configure
```

Citing

```
@misc{2016progress,  
  title={\textrm{PROGRESS} Version 1.0},  
  author={Niklasson, Anders M. and Mniszewski, Susan M and Negre, Christian F. A. and Wall, Michael E. and C},  
  year={2016},  
  url = {https://github.com/lanl/qmd-progress},  
  institution={Los Alamos National Laboratory (LANL), Los Alamos, NM (United States)}  
}
```


Chapter 2

Testing the Progress library

Testing program for the progress library

To run the tests:

Go into the build folder and type:

```
make test
```

To run the tests in verbose mode:

```
make test ARGS="-V"
```

To run a single test:

To run a test on its own (in build) we just need to type:

```
/qmd-progress/build/main <test_name>
```

, where "test_name" is the name of the test we want to run. Right now the keywords (test_name) we can pass are the following:

- density : Tests the diagonalization routine to build the density.
- sp2_short : Tests the first version of sp2
- sp2_alg1 : Algorithm 1 for sp2
- sp2_alg2 : Algorithm 2 for sp2
- sp2_alg2_ellpack : Algorithm 2 for sp2 with ellpack
- sp2_alg1_seq : See sp2_mod.F90 source file
- sp2_alg2_seq : See sp2_mod.F90 source file
- deorthogonalize_dense: See nonortho.F90 source file
- orthogonalize_dense: See nonortho.F90 source file
- buildzdiag: See genz_mod.F90 source file

To add a test:

- add the corresponding name of the test in `/progress/tests/CMakeLists.txt`
- add the corresponding keyword and test in `/progress/tests/src/main.F90`
- Copy any file that is necessary to run (data) in `/progress/tests/tests_data/`
- reconfigure and recompile

Chapter 3

Todo List

Module `prg_dos_mod`

Add LDOS.

Subprogram `prg_pulaycomponent_mod::prg_pulaycomponent0` (rho_bml, ham_bml, pcm_bml, threshold, M, bml_type, verbose)

M and bml_type will have to be removed from the input parameter.

Subprogram `prg_pulaycomponent_mod::prg_pulaycomponentt` (rho_bml, ham_bml, zmat_bml, pcm_bml, threshold, M, bml_type, verbose)

M and bml_type will have to be removed from the input parameter.

Module `prg_pulaymixer_mod`

add the density matrix mixer.

Module `prg_response_mod`

Add the response scf

Change name response_SP2 to dm_prt_response

Change name response_rs to rs_prt_response

Subprogram `prg_response_mod::prg_pert_from_file` (prt_bml, norb)

Add read perturbation from file

Subprogram `prg_system_mod::prg_parse_system` (system, filename, extin)

Integrate this loop in the loop for building the splist.

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

(LATTE related routines)	17
(PROGRESS related routines)	18
(EXTERNAL related routines)	20
(High-level codes using PROGRESS/LATTE modules)	21

Chapter 5

Modules Index

5.1 Modules List

Here is a list of all modules with brief descriptions:

prg_charges_mod	A module to compute the Mulliken charges of a chemical system	23
prg_chebyshev_mod	Module to obtain the density matrix by applying a Chebyshev polynomial expansion	25
prg_densitymatrix_mod	Module to obtain the density matrix by diagonalizing an prg_orthogonalized Hamiltonian	33
prg_dos_mod	A module to compute the Density of state (DOS) and IDOS	39
prg_extras_mod	Extra routines:	41
prg_genz_mod	To produce a matrix Z which is needed to orthogonalize H	45
prg_graph_mod	The graph module	51
prg_graphsp2parser_mod	Graph partitioning SP2 parser	59
prg_homolumo_mod	The homolumo module	61
prg_implicit_fermi_mod		61
prg_initmatrices_mod	Initialization module	62
prg_kernelparser_mod	Some general parsing functions	64
prg_nonortho_mod	Module to prg_orthogonalize and prg_deorthogonalize any operator	67
prg_normalize_mod	The prg_normalize module	69
prg_openfiles_mod	Module to handle input output files for the PROGRESS lib	74
prg_parallel_mod	The parallel module	77
prg_partition_mod	The partition module	93
prg_progress_mod	The progress module	108

prg_ptable_mod	Periodic table of elements	110
prg_pulaycomponent_mod	Produces a matrix to get the Pulay Component of the forces	117
prg_pulaymixer_mod	Pulay mixer mode	119
prg_quantumdynamics_mod	A module to add in common quantum dynamical operations	121
prg_response_mod	Module to compute the density matrix response and related quantities	125
prg_sp2_fermi_mod	The SP2 Fermi module	131
prg_sp2_mod	The SP2 module	137
prg_sp2parser_mod	SP2 parser	143
prg_subgraphloop_mod	The subgraphloop module	144
prg_syrotation_mod	A module to rotate the coordinates of a sybsystem in chemical systems	148
prg_system_mod	A module to read and handle chemical systems	150
prg_timer_mod	The timer module	165
prg_xlbo_mod	A module to perform XLBO integration	174
prg_xlkernel_mod	Add name	177

Chapter 6

Data Type Index

6.1 Data Types List

Here are the data types with brief descriptions:

prg_chebyshev_mod::chebdata_type	
General Cheb solver type	181
prg_system_mod::estruct_type	
Electronic structure type	183
prg_genz_mod::genzspdata	
Data for the genZ driver	187
prg_genz_mod::genzspinp	
Input for the genz driver	188
prg_graph_mod::graph_partitioning_t	
Trace per iteration	191
prg_graphsp2parser_mod::gsp2data_type	
General SP2 solver type	196
prg_pulaymixer_mod::mx_type	
.	199
prg_extras_mod::prg_memory_consumption	
.	200
prg_parallel_mod::rankreducedata_t	
Data structure for rection over MPI ranks	201
prg_response_mod::respdata_type	
.	202
prg_syrotation_mod::rotation_type	
Rotation type	203
prg_sp2parser_mod::sp2data_type	
General SP2 solver type	205
prg_graph_mod::subgraph_t	
Subgraph type	208
prg_system_mod::system_type	
System type	209
prg_timer_mod::timer_status_t	
Timer status type	215
prg_extras_mod::to_string	
.	217
prg_xlbo_mod::xlbo_type	
General xlbo solver type	218
prg_xlkernel_mod::xlk_type	
.	220

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

/home/christian/qmd-progress/src/prg_charges_mod.F90	223
/home/christian/qmd-progress/src/prg_chebyshev_mod.F90	223
/home/christian/qmd-progress/src/prg_densitymatrix_mod.F90	224
/home/christian/qmd-progress/src/prg_dos_mod.F90	225
/home/christian/qmd-progress/src/prg_doxy_mod.F90	226
/home/christian/qmd-progress/src/prg_extras_mod.F90	226
/home/christian/qmd-progress/src/prg_genz_mod.F90	226
/home/christian/qmd-progress/src/prg_graph_mod.F90	227
/home/christian/qmd-progress/src/prg_graphsp2parser_mod.F90	228
/home/christian/qmd-progress/src/prg_homolumo_mod.F90	229
/home/christian/qmd-progress/src/prg_implicit_fermi_mod.F90	229
/home/christian/qmd-progress/src/prg_initmatrices_mod.F90	229
/home/christian/qmd-progress/src/prg_kernelparser_mod.F90	230
/home/christian/qmd-progress/src/prg_nonortho_mod.F90	230
/home/christian/qmd-progress/src/prg_normalize_mod.F90	231
/home/christian/qmd-progress/src/prg_openfiles_mod.F90	231
/home/christian/qmd-progress/src/prg_parallel_mod.F90	232
/home/christian/qmd-progress/src/prg_partition_mod.F90	233
/home/christian/qmd-progress/src/prg_progress_mod.F90	234
/home/christian/qmd-progress/src/prg_ptable_mod.F90	234
/home/christian/qmd-progress/src/prg_pulaycomponent_mod.F90	237
/home/christian/qmd-progress/src/prg_pulaymixer_mod.F90	237
/home/christian/qmd-progress/src/prg_quantumdynamics_mod.F90	238
/home/christian/qmd-progress/src/prg_response_mod.F90	239
/home/christian/qmd-progress/src/prg_sp2_fermi_mod.F90	240
/home/christian/qmd-progress/src/prg_sp2_mod.F90	241
/home/christian/qmd-progress/src/prg_sp2parser_mod.F90	242
/home/christian/qmd-progress/src/prg_subgraphloop_mod.F90	242
/home/christian/qmd-progress/src/prg_syrotation_mod.F90	243
/home/christian/qmd-progress/src/prg_system_mod.F90	243
/home/christian/qmd-progress/src/prg_timer_mod.F90	245
/home/christian/qmd-progress/src/prg_xlbo_mod.F90	246
/home/christian/qmd-progress/src/prg_xlkernel_mod.F90	247

Chapter 8

Module Documentation

8.1 (LATTE related routines)

8.2 (PROGRESS related routines)

Modules

- module [prg_charges_mod](#)
A module to compute the Mulliken charges of a chemical system.
- module [prg_chebyshev_mod](#)
Module to obtain the density matrix by applying a Chebyshev polynomial expansion.
- module [prg_densitymatrix_mod](#)
Module to obtain the density matrix by diagonalizing an `prg_orthogonalized` Hamiltonian.
- module [prg_extras_mod](#)
Extra routines:
- module [prg_genz_mod](#)
To produce a matrix Z which is needed to orthogonalize H .
- module [prg_graph_mod](#)
The graph module.
- module [prg_graphsp2parser_mod](#)
Graph partitioning SP2 parser.
- module [prg_homolumo_mod](#)
The homolumo module.
- module [prg_initmatrices_mod](#)
Initialization module.
- module [prg_kernelparser_mod](#)
Some general parsing functions.
- module [prg_nonortho_mod](#)
Module to `prg_orthogonalize` and `prg_deorthogonalize` any operator.
- module [prg_openfiles_mod](#)
Module to handle input output files for the PROGRESS lib.
- module [prg_parallel_mod](#)
The parallel module.
- module [prg_progress_mod](#)
The progress module.
- module [prg_ptable_mod](#)
Periodic table of elements.
- module [prg_pulaycomponent_mod](#)
Produces a matrix to get the Pulay Component of the forces.
- module [prg_pulaymixer_mod](#)
Pulay mixer mode.
- module [prg_quantumdynamics_mod](#)
A module to add in common quantum dynamical operations.
- module [prg_response_mod](#)
Module to compute the density matrix response and related quantities.
- module [prg_sp2_fermi_mod](#)
The SP2 Fermi module.
- module [prg_sp2_mod](#)
The SP2 module.
- module [prg_sp2parser_mod](#)
SP2 parser.
- module [prg_syrotation_mod](#)
A module to rotate the coordinates of a subsystem in chemical systems.

- module `prg_system_mod`
A module to read and handle chemical systems.
- module `prg_timer_mod`
The timer module.
- module `prg_xlbo_mod`
A module to perform XLBO integration.
- module `prg_xlkernel_mod`
Add name.

8.2.1 Detailed Description

8.3 (EXTERNAL related routines)

8.4 (High-level codes using PROGRESS/LATTE modules)

Chapter 9

Module Documentation

9.1 prg_charges_mod Module Reference

A module to compute the Mulliken charges of a chemical system.

Functions/Subroutines

- subroutine, public [prg_get_charges](#) (rho_bml, over_bml, hindex, charges, numel, spindex, mdimin, threshold)
Constructs the charges from the density matrix.
- subroutine, public [prg_get_hscf](#) (ham0_bml, over_bml, ham_bml, spindex, hindex, hubbardu, charges, coulomb_pot_r, coulomb_pot_k, mdimin, threshold)
Constructs the SCF hamiltonian given H0, HubbardU and charges. This routine does: $H = \sum_i U_i q_i + V_i$, where U is the Hubbard parameter for every atom i . V is the coulombic potential for every atom i .

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.1.1 Detailed Description

A module to compute the Mulliken charges of a chemical system.

This module contains routines that compute properties related to charges.

9.1.2 Function/Subroutine Documentation

9.1.2.1 subroutine, public prg_charges_mod::prg_get_charges (type(bml_matrix_t), intent(inout) rho_bml, type(bml_matrix_t), intent(inout) over_bml, integer, dimension(:, :), intent(in) hindex, real(dp), dimension(:), intent(inout), allocatable charges, real(dp), dimension(:), intent(in) numel, integer, dimension(:), intent(in) spindex, integer, intent(in) mdimin, real(dp), intent(in) threshold)

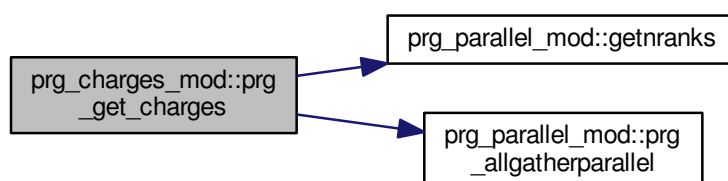
Constructs the charges from the density matrix.

Parameters

<i>rho_bml</i>	Density matrix in bml format.
<i>over_bml</i>	Overlap matrix in bml format.
<i>hindex</i>	Start and end index for every atom in the system.
<i>charges</i>	Output parameter that gives the vectorized charges.
<i>threshold</i>	Threshold value for matrix elements.

Definition at line 31 of file prg_charges_mod.F90.

Here is the call graph for this function:



9.1.2.2 subroutine, public prg_charges_mod::prg_get_hscf (type(bml_matrix_t), intent(in) *ham0_bml*, type(bml_matrix_t), intent(in) *over_bml*, type(bml_matrix_t), intent(inout) *ham_bml*, integer, dimension(:), intent(in) *spindex*, integer, dimension(:,:), intent(in) *hindex*, real(dp), dimension(:), intent(in) *hubbard_u*, real(dp), dimension(:), intent(in) *charges*, real(dp), dimension(:), intent(in) *coulomb_pot_r*, real(dp), dimension(:), intent(in) *coulomb_pot_k*, integer, intent(in) *mdimin*, real(dp), intent(in) *threshold*)

Constructs the SCF hamiltonian given H0, HubbardU and charges. This routine does: $H = \sum_i U_i q_i + V_i$, where U is the Hubbard parameter for every atom i . V is the coulombic potential for every atom i .

Parameters

<i>ham_bml</i>	Hamiltonian in bml format.
<i>over_bml</i>	Overlap in bml format.
<i>hindex</i>	Start and end index for every atom in the system.
<i>hubbard_u</i>	Hubbard parameter for every atom.
<i>charges</i>	Charges for every atom.
<i>coulomb_pot_r</i>	Coulombic potential (r contribution)
<i>coulomb_pot_k</i>	Coulombic potential (k contribution)
<i>mdim</i>	Maximum nonzeros elements per row for every row.
<i>threshold</i>	Threshold value for matrix elements.

Definition at line 101 of file prg_charges_mod.F90.

9.1.3 Variable Documentation

9.1.3.1 integer, parameter prg_charges_mod::dp = kind(1.0d0) [private]

Definition at line 17 of file prg_charges_mod.F90.

9.2 prg_chebyshev_mod Module Reference

Module to obtain the density matrix by applying a Chebyshev polynomial expansion.

Data Types

- type [chebdata_type](#)
General Cheb solver type.

Functions/Subroutines

- subroutine, public [prg_parse_cheb](#) (chebdata, filename)
Chebyshev parser. This module is used to parse all the input variables for the cheb electronic structure solver. Adding a new input keyword to the parser:
- subroutine, public [prg_build_density_cheb](#) (ham_bml, rho_bml, athr, threshold, ncoeffs, kbt, ef, bndfil, jon, verbose)
Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion.
- subroutine, public [prg_build_density_cheb_fermi](#) (ham_bml, rho_bml, athr, threshold, ncoeffs, kbt, ef, bndfil, getef, fermitol, jon, npts, trkfunc, verbose)
Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion. In this case the self-consistent recursion is applied to converge to the correct number of electrons and obtain the Fermi level.
- real(dp) function [jackson](#) (ncoeffs, i, jon)
Evaluates the Jackson Kernel Coefficients.
- subroutine [prg_get_chebcoeffs](#) (npts, kbt, ef, ncoeffs, coeffs, emin, emax)
Gets the coefficients of the Chebyshev expansion.
- subroutine [prg_get_chebcoeffs_fermi_bs](#) (npts, kbt, ef, tracesT, ncoeffs, coeffs, emin, emax, bndfil, norb, tol, jon, verbose)
Gets the coefficients of the Chebyshev expansion with Ef computation.
- subroutine [prg_get_chebcoeffs_fermi_nt](#) (npts, kbt, ef, tracesT, ncoeffs, coeffs, emin, emax, bndfil, norb, tol, jon, verbose)
Gets the coefficients of the Chebyshev expansion with Ef computation.
- real(dp) function [tr](#) (r, x)
Chebyshev polynomial obtained by recursion.
- real(dp) function [fermi](#) (e, ef, kbt)
Gives the Fermi distribution value for energy e.
- real(dp) function [absmaxderivative](#) (func, de)
Gets the absolute maximum of the derivative of a function.

Variables

- integer, parameter [dp](#) = kind(1.0d0)
- real(dp), parameter [pi](#) = 3.14159265358979323846264338327950_dp

9.2.1 Detailed Description

Module to obtain the density matrix by applying a Chebyshev polynomial expansion.

See Amparo Gil 2007 [**Amparo2007**] , See Silver et al [**Silver1996**] , See Weisse et al [**Weisse2006**]

9.2.2 Function/Subroutine Documentation

9.2.2.1 `real(dp) function prg_chebyshev_mod::absmaxderivative (real(dp), dimension(:), intent(in) func, real(dp), intent(in) de) [private]`

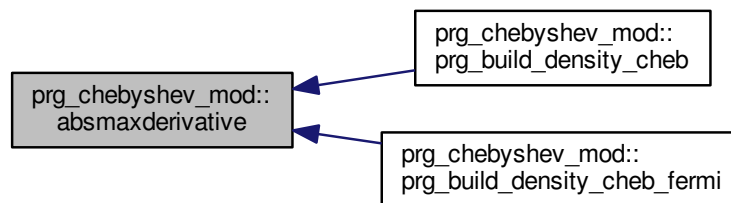
Gets the absolute maximum of the derivative of a function.

Parameters

<i>func.</i>	
<i>de</i>	Energy step.

Definition at line 802 of file prg_chebyshev_mod.F90.

Here is the caller graph for this function:



9.2.2.2 `real(dp) function prg_chebyshev_mod::fermi (real(dp), intent(in) e, real(dp), intent(in) ef, real(dp), intent(in) kbt) [private]`

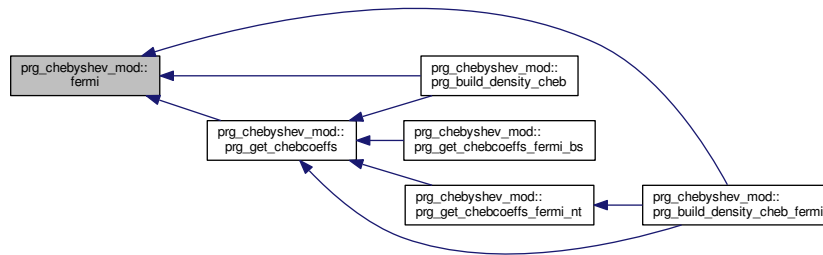
Gives the Fermi distribution value for energy *e*.

Parameters

<i>e</i>	Energy.
<i>ef</i>	Fermi energy.

Definition at line 790 of file prg_chebyshev_mod.F90.

Here is the caller graph for this function:



9.2.2.3 real(dp) function prg_chebyshev_mod::jackson (integer, intent(in) *ncoeffs*, integer, intent(in) *i*, logical, intent(in) *jon*) [private]

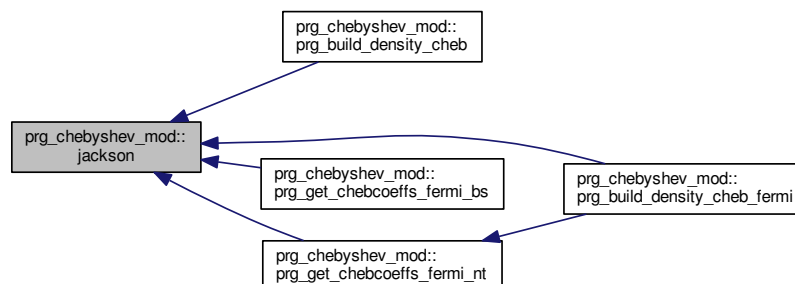
Evaluates the Jackson Kernel Coefficients.

Parameters

<i>ncoeffs</i>	Number of Chebyshev polynomial.
<i>i</i>	Coefficient number i.

Definition at line 532 of file prg_chebyshev_mod.F90.

Here is the caller graph for this function:



9.2.2.4 subroutine, public prg_chebyshev_mod::prg_build_density_cheb (type(bml_matrix_t), intent(in) *ham_bml*, type(bml_matrix_t), intent(inout) *rho_bml*, real(dp), intent(in) *athr*, real(dp), intent(in) *threshold*, integer, intent(in) *ncoeffs*, real(dp), intent(in) *kbt*, real(dp), intent(in) *ef*, real(dp), intent(in) *bndfil*, logical, intent(in) *jon*, integer, intent(in) *verbose*)

Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion.

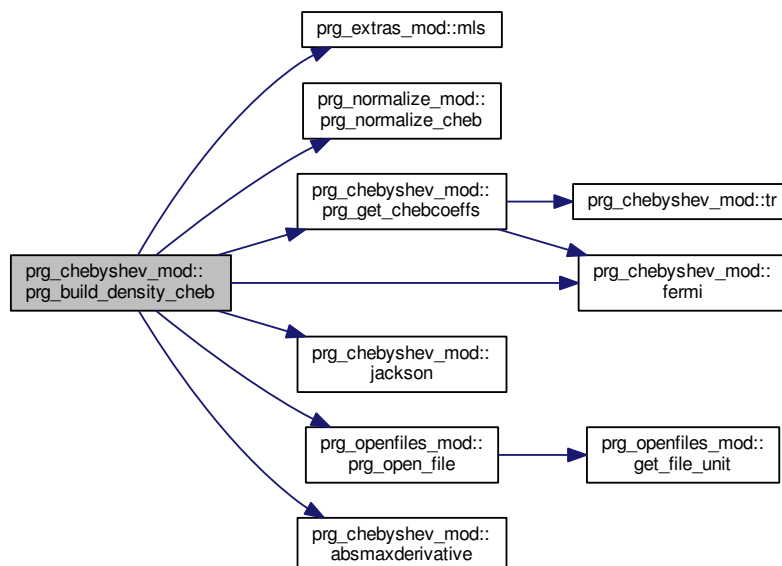
$\rho_{n+1} = b_{n+1}T_{n+1} + \rho_n$ Where, T_n is the nth Chebyshev polynomial and b_n is the nth coefficient of the expansion for the Fermi function. In the sparse version (when ellpack is used) the threshold can be varied linearly with the polynomial degree. The function is the following: $Thresh(n) = Thresh_0[a_{thr}(n - 1) + (1 - a_{thr})]$

Parameters

<i>ham_bml</i>	Input Orthogonalized Hamiltonian matrix.
<i>rho_bml</i>	Output density matrix.
<i>athr</i>	Threshold linear increasing constant.
<i>threshold</i>	Threshold for sparse matrix algebra.
<i>ncoeffs</i>	Number of Chebyshev coefficients.
<i>kbt</i>	Electronic temperature in the energy units of the Hamiltonian.
<i>ef</i>	Fermi level in the energy units of the Hamiltonian.
<i>bndfil</i>	Band filing factor.
<i>verbose</i>	Verbosity level.

Definition at line 143 of file prg_chebyshev_mod.F90.

Here is the call graph for this function:



9.2.2.5 subroutine, public `prg_chebyshev_mod::prg_build_density_cheb_fermi` (`type(bml_matrix_t)`, intent(in) *ham_bml*, `type(bml_matrix_t)`, intent(inout) *rho_bml*, `real(dp)`, intent(in) *athr*, `real(dp)`, intent(in) *threshold*, integer, intent(in) *ncoeffs*, `real(dp)`, intent(in) *kbt*, `real(dp)`, intent(inout) *ef*, `real(dp)`, intent(in) *bndfil*, logical, intent(in) *getef*, `real(dp)` *fermitol*, logical, intent(in) *jon*, integer *npts*, logical, intent(in) *trkfunc*, integer, intent(in) *verbose*)

Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion. In this case the self-consistent recursion is applied to converge to the correct number of electrons and obtain the Fermi level.

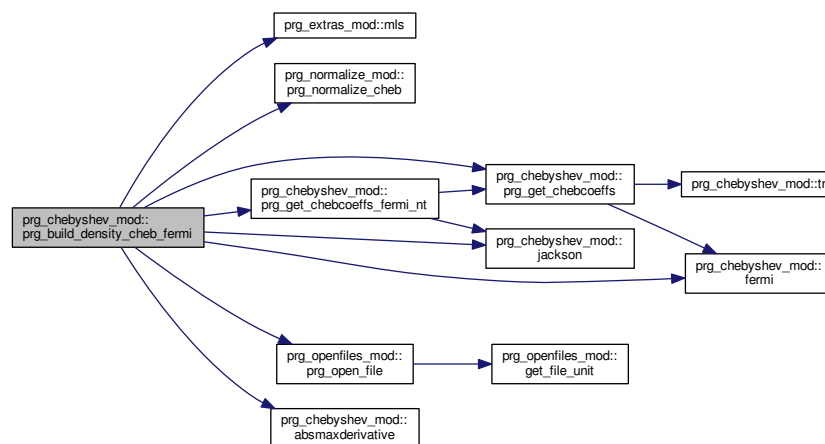
$\rho_{n+1} = b_{n+1}T_{n+1} + \rho_n$ Where, T_n is the n th Chebyshev polynomial and b_n is the n th coefficient of the expansion for the Fermi function. In the sparse version (when ellpack is used) the threshold can be varied linearly with the polynomial degree. The function is the following: $Thresh(n) = Thresh_0[a_{thr}(n - 1) + (1 - a_{thr})]$

Parameters

<i>ham_bml</i>	Input Orthogonalized Hamiltonian matrix.
<i>rho_bml</i>	Output density matrix.
<i>athr</i>	Threshold linear increasing constant.
<i>threshold</i>	Threshold for sparse matrix algebra.
<i>ncoeffs</i>	Number of Chebyshev coefficients.
<i>kbt</i>	Electronic temperature in the energy units of the Hamiltonian.
<i>ef</i>	Fermi level in the energy units of the Hamiltonian.
<i>bndfil</i>	Band filing factor.
<i>npts</i>	Number of energy point to compute the coefficients
<i>verbose</i>	Verbosity level.

Definition at line 310 of file prg_chebyshev_mod.F90.

Here is the call graph for this function:



9.2.2.6 subroutine prg_chebyshev_mod::prg_get_chebcoeffs (integer, intent(in) *npts*, real(dp), intent(in) *kbt*, real(dp), intent(in) *ef*, integer, intent(in) *ncoeffs*, real(dp), dimension(:), intent(inout) *coeffs*, real(dp), intent(in) *emin*, real(dp), intent(in) *emax*) [private]

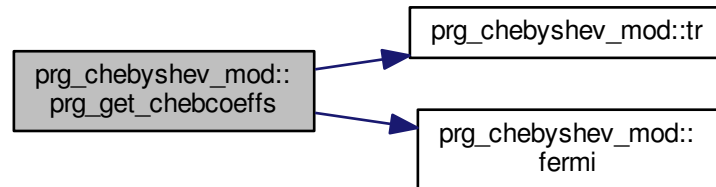
Gets the coefficients of the Chebyshev expansion.

Parameters

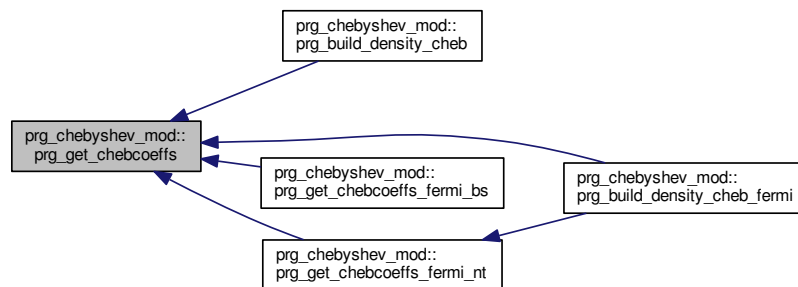
<i>npts</i>	Number of points for discretization.
<i>kbt</i>	Electronic temperature.
<i>ef</i>	Fermi level.
<i>ncoeffs</i>	Number of Chebyshev coefficients.
<i>coeffs</i>	Output vector for the Chebyshev coefficients.
<i>emin</i>	lowest boundary for the eigenvalues of H.
<i>emax</i>	highest boundary for the eigenvalues of H.

Definition at line 568 of file prg_chebyshev_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.2.2.7 subroutine `prg_chebyshev_mod::prg_get_chebcoeffs_fermi_bs` (*integer*, intent(in) *npts*, *real(dp)*, intent(in) *kbt*, *real(dp)*, intent(inout) *ef*, *real(dp)*, dimension(:), intent(in) *tracesT*, *integer*, intent(in) *ncoeffs*, *real(dp)*, dimension(:), intent(inout) *coeffs*, *real(dp)*, intent(in) *emin*, *real(dp)*, intent(in) *emax*, *real(dp)*, intent(in) *bndfil*, *integer*, intent(in) *norb*, *real(dp)*, intent(in) *tol*, *logical*, intent(in) *jon*, *integer*, intent(in) *verbose*) [private]

Gets the coefficients of the Chebyshev expansion with Ef computation.

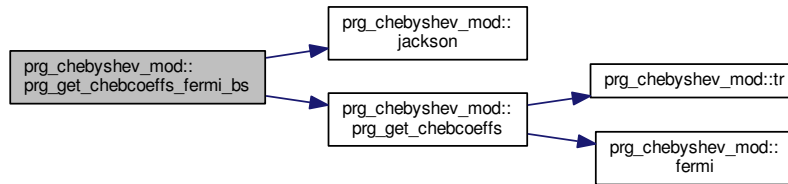
In this case we are applying the bisection method to find the root.

Parameters

<i>npts</i>	Number of points for the discretization.
<i>kbt</i>	Electronic temperature.
<i>ef</i>	Fermi level.
<i>tracesT</i>	Input traces for matrix polynomials.
<i>ncoeffs</i>	Number of Chebyshev coefficients.
<i>coeffs</i>	Output vector for the Chebyshev coefficients.
<i>emin</i>	lowest boundary for the eigenvalues of H.
<i>emax</i>	highest boundary for the eigenvalues of H.
<i>tol</i>	Tolerance for the bisection method.
<i>verbose</i>	Verbosity level.

Definition at line 620 of file prg_chebyshev_mod.F90.

Here is the call graph for this function:



9.2.2.8 subroutine prg_chebyshev_mod::prg_get_chebcoeffs_fermi_nt (integer, intent(in) *npts*, real(dp), intent(in) *kbt*, real(dp), intent(inout) *ef*, real(dp), dimension(:), intent(in) *tracesT*, integer, intent(in) *ncoeffs*, real(dp), dimension(:), intent(inout) *coeffs*, real(dp), intent(in) *emin*, real(dp), intent(in) *emax*, real(dp), intent(in) *bndfil*, integer, intent(in) *norb*, real(dp), intent(in) *tol*, logical, intent(in) *jon*, integer, intent(in) *verbose*) [private]

Gets the coefficients of the Chebyshev expansion with Ef computation.

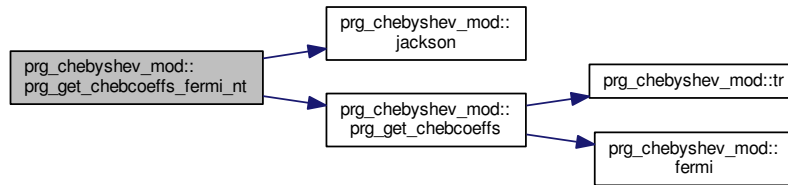
In this case the Newton-Raphson method is applied to find the root.

Parameters

<i>npst</i>	Number of points for the discretization.
<i>kbt</i>	Electronic temperature.
<i>ef</i>	Fermi level.
<i>tracesT</i>	Input traces for matrix polynomials.
<i>ncoeffs</i>	Number of Chebyshev coefficients.
<i>coeffs</i>	Output vector for the Chebyshev coefficients.
<i>emin</i>	lowest boundary for the eigenvalues of H.
<i>emax</i>	highest boundary for the eigenvalues of H.
<i>bndfil</i>	Band filling factor.
<i>norb</i>	Number of orbitals.
<i>tol</i>	Tolerance for NR method.
<i>verbose</i>	Verbosity level.

Definition at line 697 of file prg_chebyshev_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



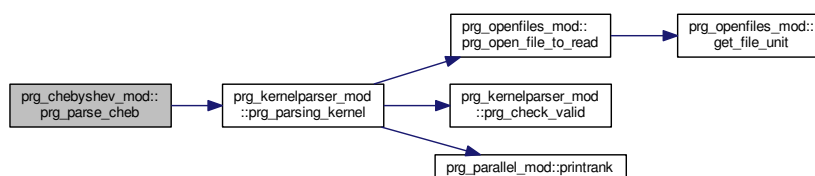
9.2.2.9 subroutine, public `prg_chebyshev_mod::prg_parse_cheb (type(chebdata_type), intent(inout) chebdata, character(len=*) filename)`

Chebyshev parser. This module is used to parse all the input variables for the cheb electronic structure solver. Adding a new input keyword to the parser:

- If the variable is real, we have to increase `nkey_re`.
- Add the keyword (character type) in the `keyvector_re` vector.
- Add a default value (real type) in the `valvector_re`.
- Define a new variable and pass the value through `valvector_re(num)` where `num` is the position of the new keyword in the vector.

Definition at line 54 of file `prg_chebyshev_mod.F90`.

Here is the call graph for this function:



9.2.2.10 `real(dp) function prg_chebyshev_mod::tr (integer, intent(in) r, real(dp), intent(in) x) [private]`

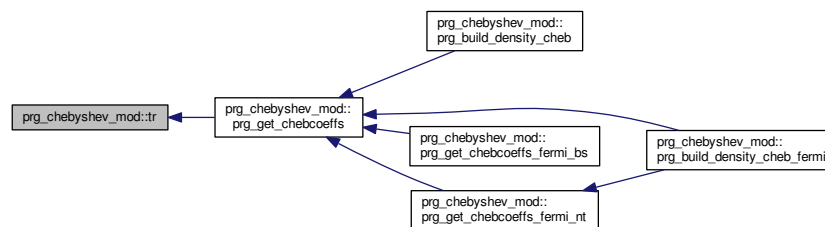
Chebyshev polynomial obtained by recursion.

Parameters

<i>r</i>	rth polynomial.
<i>x</i>	argument the evaluate the polynomial.

Definition at line 777 of file prg_chebyshev_mod.F90.

Here is the caller graph for this function:



9.2.3 Variable Documentation

9.2.3.1 `integer, parameter prg_chebyshev_mod::dp = kind(1.0d0) [private]`

Definition at line 23 of file prg_chebyshev_mod.F90.

9.2.3.2 `real(dp), parameter prg_chebyshev_mod::pi = 3.14159265358979323846264338327950_dp [private]`

Definition at line 24 of file prg_chebyshev_mod.F90.

9.3 prg_densitymatrix_mod Module Reference

Module to obtain the density matrix by diagonalizing an prg_orthogonalized Hamiltonian.

Functions/Subroutines

- subroutine, public [prg_build_density_t0](#) (ham_bml, rho_bml, threshold, bndfil)
Builds the density matrix from H_0 for zero electronic temperature. $\rho = C\Theta(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. $\Theta()$ is the Heaviside function.
- subroutine, public [prg_build_density_t](#) (ham_bml, rho_bml, threshold, bndfil, kbt, ef)
Builds the density matrix from H_0 for electronic temperature T . $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function.
- subroutine, public [prg_build_density_t_fermi](#) (ham_bml, rho_bml, threshold, kbt, ef, verbose)
Builds the density matrix from H_0 for electronic temperature T . $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function. In this routine the Fermi level is passed as an argument.
- subroutine, public [prg_build_atomic_density](#) (rhoat_bml, numel, hindex, spindex, norb, bml_type)
Builds the atomic density matrix. $\rho_{ii} = \text{mathcal{Z}}_{ii}$ Where, $\text{mathcal{Z}}_{ii}$ is the number of electrons for orbital i .
- subroutine, public [prg_get_flevel](#) (eigenvalues, kbt, bndfil, tol, Ef)
Routine to compute the Fermi level given a set of eigenvalues and a temperature. It applies the Bisection method over the function: $g(\mu) = \sum_k 2f(\epsilon_k - \mu) - N = 0$ Where $f(\epsilon_k - \mu) = \frac{1}{1 + \exp((\epsilon_k - \mu)/(k_b T))}$.
- subroutine, public [prg_get_eigenvalues](#) (ham_bml, eigenvalues, verbose)
Gets the eigenvalues of the Orthogonalized Hamiltonian.
- subroutine, public [prg_check_idempotency](#) (mat_bml, threshold, idempotency)
To check the idempotency error of a matrix. This is calculated as the Frobenius norm of $(A - A^2)$.
- real(dp) function [fermi](#) (e, ef, kbt)
Gives the Fermi distribution value for energy e .

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.3.1 Detailed Description

Module to obtain the density matrix by diagonalizing an prg_orthogonalized Hamiltonian.

9.3.2 Function/Subroutine Documentation

9.3.2.1 `real(dp) function prg_densitymatrix_mod::fermi (real(dp), intent(in) e, real(dp), intent(in) ef, real(dp), intent(in) kbt)`
[private]

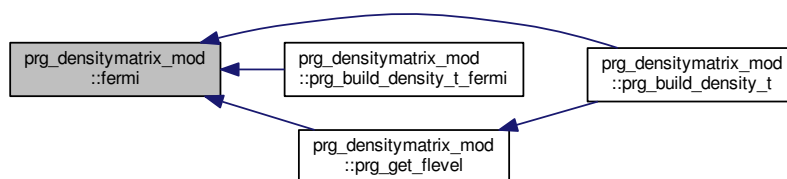
Gives the Fermi distribution value for energy e .

Parameters

<i>e</i>	Energy.
<i>ef</i>	Fermi energy.

Definition at line 413 of file prg_densitymatrix_mod.F90.

Here is the caller graph for this function:



9.3.2.2 subroutine, public prg_densitymatrix_mod::prg_build_atomic_density (type(bml_matrix_t), intent(inout) rhoat_bml, real(dp), dimension(:), intent(in) numel, integer, dimension(:,:), intent(in) hindex, integer, dimension(:), intent(in) spindex, integer, intent(in) norb, character(len=*), intent(in) bml_type)

Builds the atomic density matrix. $\rho_{ii} = \text{mathcal{Z}}_{ii}$ Where, $\text{mathcal{Z}}_{ii}$ is the number of electrons for orbital i.

Parameters

<i>rhoat</i>	Output atomic diagonal density matrix,
<i>hindex</i>	Start and end index for every atom in the system.
<i>numel</i>	Number of electrons per specie. It runs over the specie index.
<i>spindex</i>	Specie index.
<i>norbs</i>	Number of orbitals.

Definition at line 215 of file prg_densitymatrix_mod.F90.

9.3.2.3 subroutine, public prg_densitymatrix_mod::prg_build_density_t (type(bml_matrix_t), intent(in) ham_bml, type(bml_matrix_t), intent(inout) rho_bml, real(8), intent(in) threshold, real(8), intent(in) bndfil, real(8), intent(in) kbt, real(8), intent(inout) ef)

Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function.

Parameters

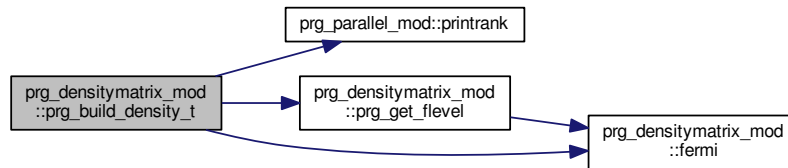
<i>ham_bml</i>	Input Orthogonalized Hamiltonian matrix.
<i>rho_bml</i>	Output density matrix,
<i>threshold</i>	Threshold for sparse matrix algebra.
<i>bndfil</i>	Filing factor.

Warning

This does not solve the generalized eigenvalue problem. The Hamiltonian that comes in has to be preprg_↔ orthogonalized.

Definition at line 94 of file prg_densitymatrix_mod.F90.

Here is the call graph for this function:



9.3.2.4 subroutine, public prg_densitymatrix_mod::prg_build_density_t0 (type(bml_matrix_t), intent(in) ham_bml, type(bml_matrix_t), intent(inout) rho_bml, real(8), intent(in) threshold, real(8), intent(in) bndfil)

Builds the density matrix from H_0 for zero electronic temperature. $\rho = C\Theta(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. $\Theta()$ is the Heaviside function.

Parameters

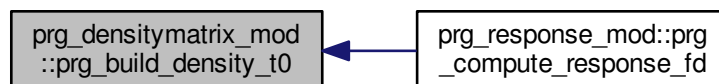
<i>ham_bml</i>	Input Orthogonalized Hamiltonian matrix.
<i>rho_bml</i>	Output density matrix,
<i>threshold</i>	Threshold for sparse matrix algebra.
<i>bndfil</i>	Filing factor.

Warning

This does not solve the generalized eigenvalue problem. The Hamiltonian that comes in has to be preprg_orthogonalized.

Definition at line 33 of file prg_densitymatrix_mod.F90.

Here is the caller graph for this function:



9.3.2.5 subroutine, public prg_densitymatrix_mod::prg_build_density_t_fermi (type(bml_matrix_t), intent(in) ham_bml, type(bml_matrix_t), intent(inout) rho_bml, real(8), intent(in) threshold, real(8), intent(in) kbt, real(8), intent(in) ef, integer, intent(in), optional verbose)

Builds the density matrix from H_0 for electronic temperature T . $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function. In this routine the Fermi level is passed as an argument.

Parameters

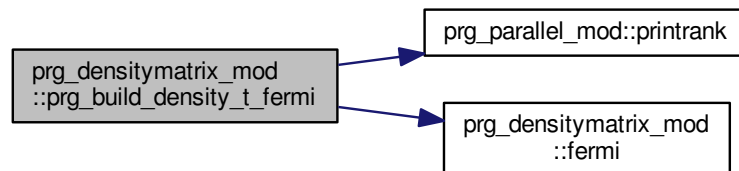
<i>ham_bml</i>	Input Orthogonalized Hamiltonian matrix.
<i>rho_bml</i>	Output density matrix,
<i>threshold</i>	Threshold for sparse matrix algebra.

Warning

This does not solve the generalized eigenvalue problem. The Hamiltonian that comes in has to be preprg_orthogonalized.

Definition at line 156 of file prg_densitymatrix_mod.F90.

Here is the call graph for this function:



9.3.2.6 subroutine, public prg_densitymatrix_mod::prg_check_idempotency (type(bml_matrix_t), intent(in) *mat_bml*, real(dp), intent(in) *threshold*, real(dp), intent(out) *idempotency*)

To check the idempotency error of a matrix. This is calculated as the Frobenius norm of $(A - A^2)$.

Parameters

<i>mat_bml</i>	Some bml matrix
<i>idempotency</i>	(Output value of the idempotency error)

Definition at line 389 of file prg_densitymatrix_mod.F90.

9.3.2.7 subroutine, public prg_densitymatrix_mod::prg_get_eigenvalues (type(bml_matrix_t), intent(in) *ham_bml*, real(dp), dimension(:), intent(inout), allocatable *eigenvalues*, integer, intent(in) *verbose*)

Gets the eigenvalues of the Orthogonalized Hamiltonian.

Parameters

<i>ham_bml</i>	Input Orthogonalized Hamiltonian matrix.
<i>eigenvalues</i>	Output eigenvalues of the system.
<i>verbose</i>	Verbosity level.

Definition at line 344 of file prg_densitymatrix_mod.F90.

9.3.2.8 subroutine, public prg_densitymatrix_mod::prg_get_flevel (real(dp), dimension(:), intent(in) *eigenvalues*, real(dp), intent(in) *kbt*, real(dp), intent(in) *bndfil*, real(dp) *tol*, real(dp), intent(inout) *Ef*)

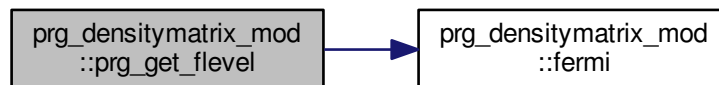
Routine to compute the Fermi level given a set of eigenvalues and a temperature. It applies the Bisection method over the function: $g(\mu) = \sum_k 2f(\epsilon_k - \mu) - N = 0$ Where $f(\epsilon_k - \mu) = \frac{1}{1 + \exp((\epsilon_k - \mu)/(k_b T))}$.

Parameters

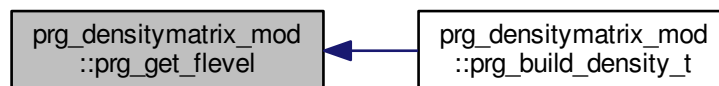
<i>eigenvalues</i>	Eigenvalues of the system ($\{\epsilon_k\}$).
<i>kbt</i>	Temperature times the Boltzmann's constant ($k_b T$).
<i>bndfil</i>	Filing factor ($N_{el}/(2 * N_{orbs})$).
<i>tol</i>	Tolerance for the bisection method.
<i>Ef</i>	Fermi level (μ).

Definition at line 280 of file prg_densitymatrix_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.3 Variable Documentation

9.3.3.1 integer, parameter prg_densitymatrix_mod::dp = kind(1.0d0) [private]

Definition at line 14 of file prg_densitymatrix_mod.F90.

9.4 prg_dos_mod Module Reference

A module to compute the Density of state (DOS) and IDOS.

Functions/Subroutines

- subroutine, public [prg_write_tdos](#) (eigenvals, gamma, npts, emin, emax, filename)
Writes the total DOS into a file. $DOS(\epsilon) = \sum_k L(\epsilon - \epsilon_k)$ Where $\int_{-\infty}^{\infty} DOS(\epsilon) = Nstates$.
- real(dp) function [lorentz](#) (energy, eigenvals, loads, Gamma)
Lorentzian Function.

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.4.1 Detailed Description

A module to compute the Density of state (DOS) and IDOS.

This module will be used to compute DOS and IDOS.

Todo Add LDOS.

9.4.2 Function/Subroutine Documentation

9.4.2.1 real(dp) function prg_dos_mod::lorentz (real(dp), intent(in) *energy*, real(dp), dimension(:), intent(in) *eigenvals*, real(dp), dimension(:), intent(in) *loads*, real(dp), intent(in) *Gamma*) [private]

Lorentzian Function.

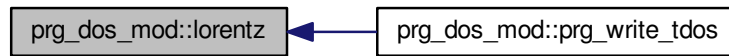
Computes: $L(\epsilon) = \sum_k \frac{\omega(k)\Gamma}{2\pi} \frac{1}{(\epsilon - \epsilon_k)^2 + (\Gamma/2)^2}$

Parameters

<i>energy</i>	Energy point.
<i>eigenvals</i>	Eigenvalues of the system.
<i>Gamma</i>	Lorentz function broadening.

Definition at line 78 of file prg_dos_mod.F90.

Here is the caller graph for this function:



9.4.2.2 subroutine, public `prg_dos_mod::prg_write_tdos` (`real(dp)`, `dimension(:)`, `intent(in)` *eigenvals*, `real(dp)`, `intent(in)` *gamma*, `integer`, `intent(in)` *npts*, `real(dp)`, `intent(in)` *emin*, `real(dp)`, `intent(in)` *emax*, `character(len=*)`, `intent(in)` *filename*)

Writes the total DOS into a file. $DOS(\epsilon) = \sum_k L(\epsilon - \epsilon_k)$ Where $\int_{-\infty}^{\infty} DOS(\epsilon) = Nstates$.

Note

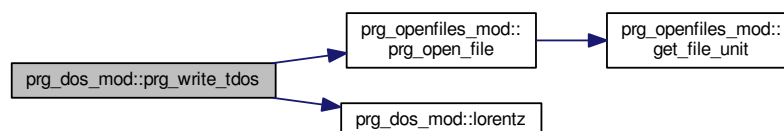
DOS is NOT shifted respect to Ef.

Parameters

<i>eigenvals</i>	Eigenvalues of the system. gamma Lorentzian width.
<i>npts</i>	Number of energy points.
<i>emin</i>	Minimum energy value.
<i>emax</i>	Maximum energy value.
<i>filename</i>	Filename to write the DOS.

Definition at line 36 of file `prg_dos_mod.F90`.

Here is the call graph for this function:



9.4.3 Variable Documentation

9.4.3.1 integer, parameter `prg_dos_mod::dp = kind(1.0d0)` [private]

Definition at line 17 of file `prg_dos_mod.F90`.

9.5 prg_extras_mod Module Reference

Extra routines:

Data Types

- interface [prg_memory_consumption](#)
- interface [to_string](#)

Functions/Subroutines

- character(len=:) function, allocatable [to_string_integer](#) (i)
Convert integer to string.
- character(len=:) function, allocatable [to_string_long_long](#) (i)
Convert integer to string.
- character(len=:) function, allocatable [to_string_double](#) (x)
Convert double to string.
- subroutine, public [prg_print_matrix](#) (matname, amat, i1, i2, j1, j2)
To write a dense matrix to screen.
- real(dp) function, public [mls](#) ()
To get the actual time in milliseconds.
- subroutine, public [prg_delta](#) (x, s, nn, dta)
Delta function $\|X^tSX - I\|$. CFAN, March 2015.
- subroutine, public [prg_get_mem](#) (procname, tag)
Get proc memory.
- subroutine [prg_twonorm](#) (a, nn, norm2)

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.5.1 Detailed Description

Extra routines:

A module to add any extra routine considered necessary but which is NOT essential for any other PROGRESS routines.

9.5.2 Function/Subroutine Documentation

9.5.2.1 real(dp) function, public prg_extras_mod::mls ()

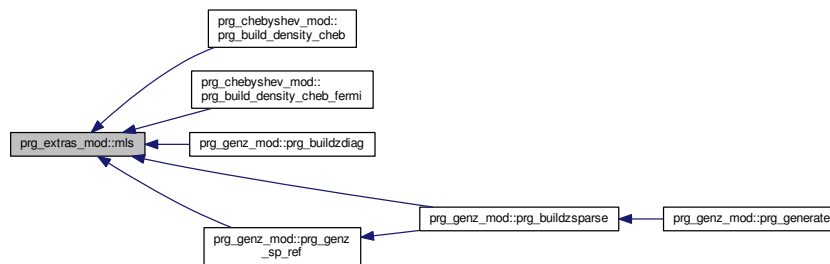
To get the actual time in milliseconds.

Parameters

<i>m/s</i>	Output value with the machine time in milliseconds.
------------	---

Definition at line 139 of file prg_extras_mod.F90.

Here is the caller graph for this function:



9.5.2.2 subroutine, public prg_extras_mod::prg_delta (real(dp), dimension(nn,nn) *x*, real(dp), dimension(nn,nn) *s*, integer *nn*, real(dp) *dta*)

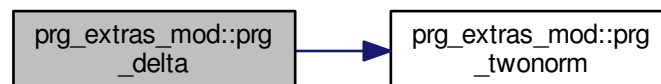
Delta function $||X^{tSX} - I||$. CFAN, March 2015.

Parameters

<i>x</i>	input matrix.
<i>s</i>	overlap matrix.
<i>dta</i>	Delta output value.

Definition at line 155 of file prg_extras_mod.F90.

Here is the call graph for this function:



9.5.2.3 subroutine, public prg_extras_mod::prg_get_mem (character(*), intent(in) *procname*, character(*), intent(in) *tag*)

Get proc memory.

Parameters

<i>procname</i>	Process name to get the mem usage.
<i>tag</i>	Tag to pprint the processor mem usage.

Definition at line 191 of file prg_extras_mod.F90.

9.5.2.4 subroutine, public prg_extras_mod::prg_print_matrix (character(len=*) *matname*, real(dp), dimension(:, :), intent(in) *amat*, integer, intent(in) *i1*, integer, intent(in) *i2*, integer, intent(in) *j1*, integer, intent(in) *j2*)

To write a dense matrix to screen.

Parameters

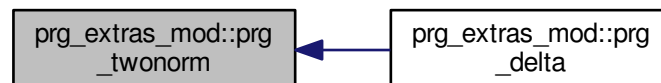
<i>matname</i>	Matrix name.
<i>amat</i>	Matrix to be printed.
<i>i1</i>	Print from row i1.
<i>i2</i>	Print up to from row i2.
<i>j1</i>	Print from column j1.
<i>j2</i>	Print up to column j2.

Definition at line 100 of file prg_extras_mod.F90.

9.5.2.5 subroutine prg_extras_mod::prg_twonorm (real(dp), dimension(nn,nn) *a*, integer *nn*, real(dp) *norm2*)
[private]

Definition at line 211 of file prg_extras_mod.F90.

Here is the caller graph for this function:



9.5.2.6 character(len=:) function, allocatable prg_extras_mod::to_string_double (double precision, intent(in) *x*)
[private]

Convert double to string.

Parameters

<i>x</i>	The double
----------	------------

Returns

The string

Definition at line 80 of file prg_extras_mod.F90.

9.5.2.7 `character(len=:)` function, allocatable `prg_extras_mod::to_string_integer (integer, intent(in) i)` `[private]`

Convert integer to string.

Parameters

<i>i</i>	The integer
----------	-------------

Returns

The string

Definition at line 46 of file prg_extras_mod.F90.

9.5.2.8 `character(len=:)` function, allocatable `prg_extras_mod::to_string_long_long (integer(kind=c_long_long), intent(in) i)` `[private]`

Convert integer to string.

Parameters

<i>i</i>	The integer
----------	-------------

Returns

The string

Definition at line 62 of file prg_extras_mod.F90.

9.5.3 Variable Documentation

9.5.3.1 `integer, parameter prg_extras_mod::dp = kind(1.0d0)` `[private]`

Definition at line 31 of file prg_extras_mod.F90.

9.6 prg_genz_mod Module Reference

To produce a matrix Z which is needed to orthogonalize H .

Data Types

- type `genzspdata`
contains the data for the genZ driver.
- type `genzspinp`
Input for the genz driver.

Functions/Subroutines

- subroutine, public `prg_parse_zsp` (input, filename)
The parser for genz solver.
- subroutine `prg_init` (self, input)
Initializes the genz input variables.
- subroutine `prg_allocatezspmat` (self, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, norb, bml_type)
Allocates the matrices for the XI integration of Z.
- subroutine, public `prg_init_zspmat` (igenz, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, norb, bml_type)
Initiates the matrices for the XI integration of Z.
- subroutine `prg_generate` (self, over_bml, zmat_bml, igenz, mdim, bml_type, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml)
Generates the Z matrix.
- subroutine, public `prg_buildzdiag` (smat_bml, zmat_bml, threshold, mdimin, bml_type, verbose)
Usual subroutine involving diagonalization.
- subroutine, public `prg_buildzsparse` (smat_bml, zmat_bml, igenz, mdim, bml_type, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, nfirst, nrefi, nreff, thresholdi, thresholdf, integration, verbose)
Inverse factorization using niklasson's algorithm.
- subroutine, public `prg_genz_sp_initialz0` (smat_bml, zmat_bml, norb, mdim, bml_type_f, threshold)
- subroutine, public `prg_genz_sp_initial_zmat` (smat_bml, zmat_bml, norb, mdim, bml_type_f, threshold)
Estimate Z matrix.
- subroutine `prg_genz_sp_int` (zmat_bml, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, igenz, norb, bml_type, threshold)
- subroutine, public `prg_genz_sp_ref` (smat_bml, zmat_bml, nref, norb, bml_type, threshold)

Variables

- integer, parameter `dp` = kind(1.0d0)

9.6.1 Detailed Description

To produce a matrix Z which is needed to orthogonalize H .

$$H_{orth} = Z^\dagger H Z \text{ See Negre 2016 [2]}$$

9.6.2 Function/Subroutine Documentation

9.6.2.1 subroutine `prg_genz_mod::prg_allocatezspmat` (class(`genzspdata`), intent(in) *self*, type(`bml_matrix_t`) *zk1_bml*, type(`bml_matrix_t`) *zk2_bml*, type(`bml_matrix_t`) *zk3_bml*, type(`bml_matrix_t`) *zk4_bml*, type(`bml_matrix_t`) *zk5_bml*, type(`bml_matrix_t`) *zk6_bml*, integer *norb*, character(20) *bml_type*) [private]

Allocates the matrices for the XI integration of Z.

Parameters

<i>self</i>	input zsp variables
<i>zk1_bml-zk6_bml</i>	history record of the previous Z matrices.
<i>norb</i>	number of orbitals.
<i>bml_type</i>	the bml format we are passing.

Definition at line 174 of file prg_genz_mod.F90.

9.6.2.2 subroutine, public prg_genz_mod::prg_buildzdiag (type(bml_matrix_t), intent(inout) *smat_bml*, type(bml_matrix_t) *zmat_bml*, real(dp) *threshold*, integer, intent(in) *mdimin*, character(len=*) *bml_type*, integer, intent(in), optional *verbose*)

Usual subroutine involving diagonalization.

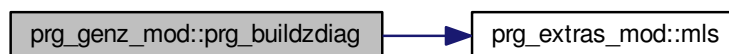
$Z = U\sqrt{s}U^\dagger$, where U = eigenvectors and s = eigenvalues. The purpose of this subroutine is to have an exact way of computing z for comparing with the sparse approach.

Parameters

<i>smat_bml</i>	Overlap matrix in bml format.
<i>zmat_bml</i>	Congruence transform in bml format.
<i>threshold</i>	Threshold value to use, in this case, only in the backtransformation to ellpack format.
<i>mdim</i>	Maximun nonzero to use, in this case, only in the backtransformation to ellpack format.
<i>bml_type</i>	the bml type we are passing.

Definition at line 281 of file prg_genz_mod.F90.

Here is the call graph for this function:



9.6.2.3 subroutine, public prg_genz_mod::prg_buildzsparse (type(bml_matrix_t) *smat_bml*, type(bml_matrix_t) *zmat_bml*, integer *igenz*, integer *mdim*, character(20) *bml_type*, type(bml_matrix_t) *zk1_bml*, type(bml_matrix_t) *zk2_bml*, type(bml_matrix_t) *zk3_bml*, type(bml_matrix_t) *zk4_bml*, type(bml_matrix_t) *zk5_bml*, type(bml_matrix_t) *zk6_bml*, integer *nfirst*, integer *nrefi*, integer *nreff*, real(dp) *thresholdi*, real(dp) *thresholdf*, logical *integration*, integer *verbose*)

Inverse factorization using niklasson's algorithm.

Parameters

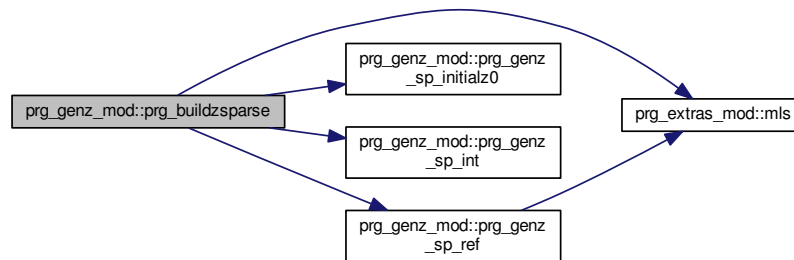
<i>smat_bml</i>	overlap matrix
-----------------	----------------

Parameters

<i>zmat_bml</i>	congruence transform to be updated or computed. (bml format)
<i>igenz</i>	counter to keep track of the calls to this subroutine.
<i>mdim</i>	dimension of the maxnonzero per row.
<i>zk1_bml-zk6_bml</i>	history of the past congruence transforms.
<i>nfirst</i>	first pre septs with nrefi and thresholdi.
<i>nrefi</i>	number of refinement iterations for the firsts "nfirst" steps.
<i>nreff</i>	number of refinement iterations for the rest of the steps.
<i>integration</i>	if we want to apply xl integration scheme for z (default is always .true.)
<i>verbose</i>	to print extra information.

Definition at line 421 of file prg_genz_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.6.2.4 subroutine `prg_genz_mod::prg_generate` (`class(genzspdata)`, `intent(in) self`, `type(bml_matrix_t)`, `intent(in) over_bml`, `type(bml_matrix_t)`, `intent(inout) zmat_bml`, `integer igenz`, `integer mdim`, `character(20) bml_type`, `type(bml_matrix_t)`, `intent(inout) zk1_bml`, `type(bml_matrix_t)`, `intent(inout) zk2_bml`, `type(bml_matrix_t)`, `intent(inout) zk3_bml`, `type(bml_matrix_t)`, `intent(inout) zk4_bml`, `type(bml_matrix_t)`, `intent(inout) zk5_bml`, `type(bml_matrix_t)`, `intent(inout) zk6_bml`) [`private`]

Generates the Z matrix.

Parameters

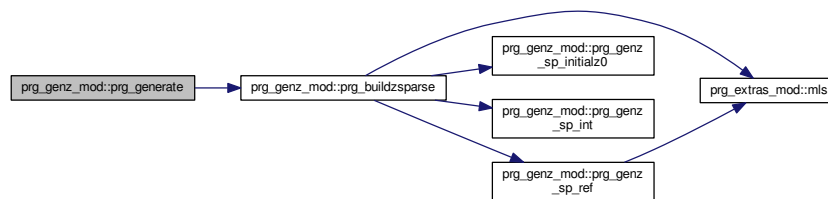
<i>over_bml</i>	Overlap matrix.
-----------------	-----------------

Parameters

<i>zmat_bml</i>	Congruence transform to be computed. (bml format)
<i>igenz</i>	Counter to keep track of the calls to this subroutine.
<i>mdim</i>	dimension of the maxnonzero per row.
<i>zk1_bml-zk6_bml</i>	history of the past congruence transforms.

Definition at line 242 of file prg_genz_mod.F90.

Here is the call graph for this function:



9.6.2.5 subroutine, public `prg_genz_mod::prg_genz_sp_initial_zmat` (`type(bml_matrix_t)`, intent(in) *smat_bml*, `type(bml_matrix_t)` *zmat_bml*, integer *norb*, integer *mdim*, character(20) *bml_type_f*, `real(dp)`, intent(in) *threshold*)

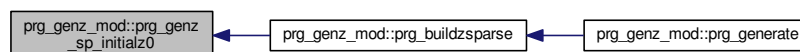
Estimate Z matrix.

Definition at line 600 of file prg_genz_mod.F90.

9.6.2.6 subroutine, public `prg_genz_mod::prg_genz_sp_initialz0` (`type(bml_matrix_t)`, intent(in) *smat_bml*, `type(bml_matrix_t)` *zmat_bml*, integer *norb*, integer *mdim*, character(20) *bml_type_f*, `real(dp)` *threshold*)

Definition at line 476 of file prg_genz_mod.F90.

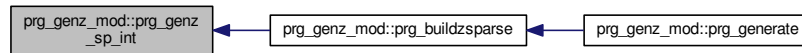
Here is the caller graph for this function:



9.6.2.7 subroutine prg_genz_mod::prg_genz_sp_int (type(bml_matrix_t) *zmat_bml*, type(bml_matrix_t) *zk1_bml*, type(bml_matrix_t) *zk2_bml*, type(bml_matrix_t) *zk3_bml*, type(bml_matrix_t) *zk4_bml*, type(bml_matrix_t) *zk5_bml*, type(bml_matrix_t) *zk6_bml*, integer *igenz*, integer *norb*, character(20) *bml_type*, real(dp) *threshold*) [private]

Definition at line 739 of file prg_genz_mod.F90.

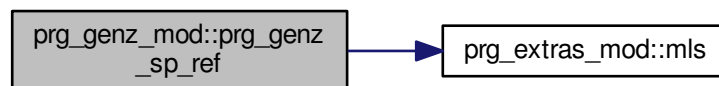
Here is the caller graph for this function:



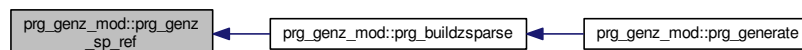
9.6.2.8 subroutine, public prg_genz_mod::prg_genz_sp_ref (type(bml_matrix_t), intent(in) *smat_bml*, type(bml_matrix_t), intent(inout) *zmat_bml*, integer, intent(in) *nref*, integer, intent(inout) *norb*, character(20), intent(in) *bml_type*, real(dp), intent(in) *threshold*)

Definition at line 806 of file prg_genz_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.6.2.9 subroutine prg_genz_mod::prg_init (class(genzspdata), intent(out) *self*, type(genzspinp), intent(in) *input*) [private]

Initializes the genz input variables.

Parameters

<i>self</i>	basic input parameters.
<i>input</i>	basic input parameters from the parser.

Definition at line 152 of file prg_genz_mod.F90.

9.6.2.10 subroutine, public prg_genz_mod::prg_init_zspmat (integer *igenz*, type(bml_matrix_t) *zk1_bml*, type(bml_matrix_t) *zk2_bml*, type(bml_matrix_t) *zk3_bml*, type(bml_matrix_t) *zk4_bml*, type(bml_matrix_t) *zk5_bml*, type(bml_matrix_t) *zk6_bml*, integer *norb*, character(20) *bml_type*)

Initiates the matrices for the XI integration of Z.

Parameters

<i>self</i>	input zsp variables
<i>zk1_bml</i> - <i>zk6_bml</i>	history record of the previous Z matrices.
<i>norb</i>	number of orbitals.
<i>bml_type</i>	the bml format we are passing.

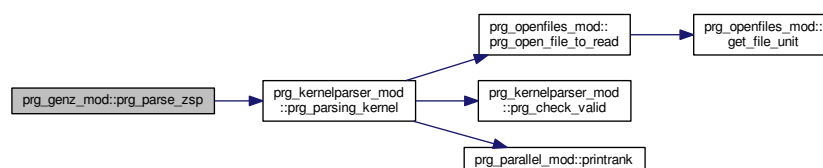
Definition at line 207 of file prg_genz_mod.F90.

9.6.2.11 subroutine, public prg_genz_mod::prg_parse_zsp (type(genzspinp), intent(inout) *input*, character(len=*) *filename*)

The parser for genz solver.

Definition at line 88 of file prg_genz_mod.F90.

Here is the call graph for this function:



9.6.3 Variable Documentation

9.6.3.1 integer, parameter prg_genz_mod::dp = kind(1.0d0) [private]

Definition at line 20 of file prg_genz_mod.F90.

9.7 prg_graph_mod Module Reference

The graph module.

Data Types

- type [graph_partitioning_t](#)
Trace per iteration.
- type [subgraph_t](#)
Subgraph type.

Functions/Subroutines

- subroutine, public [prg_initsubgraph](#) (sg, pnum, hsize)
Initialize subgraph.
- subroutine, public [prg_destroysubgraph](#) (sg)
Destroy subgraph.
- subroutine, public [prg_initgraphpartitioning](#) (gp, pname, np, nnodes, nnodes2)
Initialize graph partitioning.
- subroutine, public [prg_destroygraphpartitioning](#) (gp)
Destroy graph partitioning.
- subroutine, public [prg_printgraphpartitioning](#) (gp)
Print graph partitioning structure data.
- subroutine, public [prg_equalpartition](#) (gp, nodesPerPart, nnodes)
Create equal graph partitions, based on number of rows/orbitals.
- subroutine, public [prg_equalgrouppartition](#) (gp, hindex, ngroup, nodesPerPart, nnodes)
Create equal group graph partitions, based on number of atoms/groups.
- subroutine, public [prg_filepartition](#) (gp, partFile)
Read graph partitions from a file, based on number of rows/orbitals.
- subroutine [prg_readpart](#) (gp, partFile)
Read parts (core) from part file.
- subroutine, public [prg_fnormgraph](#) (gp)
Accumulate trace norm across all subgraphs.

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.7.1 Detailed Description

The graph module.

9.7.2 Function/Subroutine Documentation

9.7.2.1 subroutine, public [prg_graph_mod::prg_destroygraphpartitioning](#) (type ([graph_partitioning_t](#)), intent(inout) *gp*)

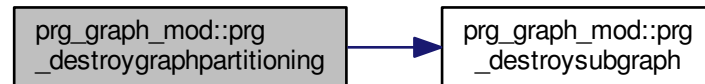
Destroy graph partitioning.

Parameters

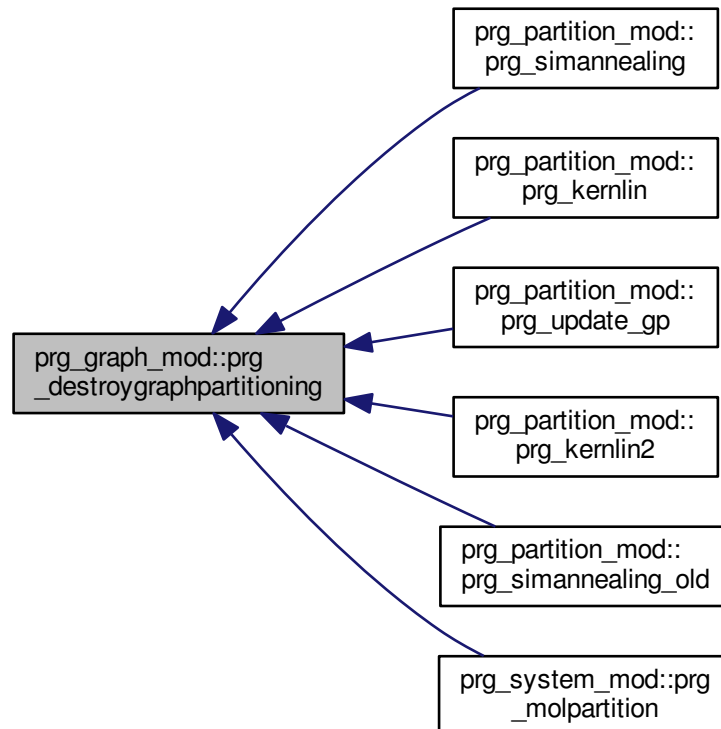
<code>sg</code>	Subgraph
-----------------	----------

Definition at line 263 of file `prg_graph_mod.F90`.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.2 subroutine, public `prg_graph_mod::prg_destroysubgraph (type (subgraph_t), intent(inout) sg)`

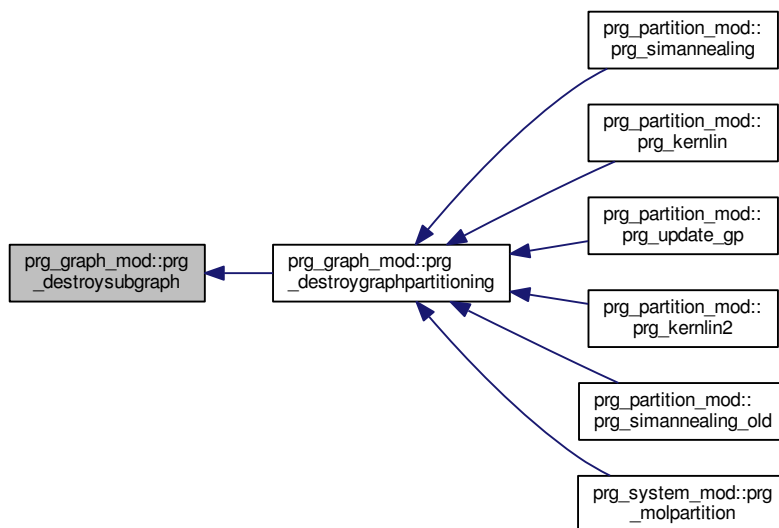
Destroy subgraph.

Parameters

<i>sg</i>	Subgraph
-----------	----------

Definition at line 159 of file prg_graph_mod.F90.

Here is the caller graph for this function:



9.7.2.3 subroutine, public `prg_graph_mod::prg_equalgrouppartition` (`type (graph_partitioning_t)`, intent(inout) *gp*, integer, dimension(2,ngroup), intent(in) *hindex*, integer, intent(in) *ngroup*, integer, intent(in) *nodesPerPart*, integer, intent(in) *nnodes*)

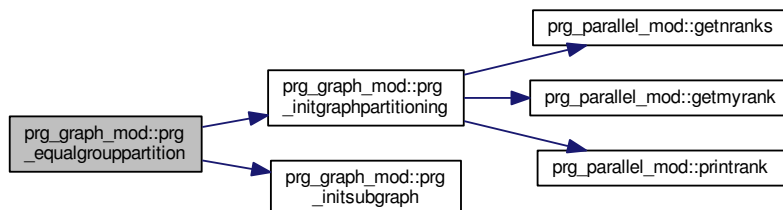
Create equal group graph partitions, based on number of atoms/groups.

Parameters

<i>gp</i>	Graph partitioning
<i>hindex</i>	Node indeces that represent ranges of atoms/groups
<i>ngroup</i>	Number of group nodes
<i>nodesPerPart</i>	Number of core nodes per partition
<i>nnodes</i>	Total nodes in Hamiltonian matrix

Definition at line 402 of file prg_graph_mod.F90.

Here is the call graph for this function:



9.7.2.4 subroutine, public `prg_graph_mod::prg_equalpartition (type (graph_partitioning_t), intent(inout) gp, integer, intent(in) nodesPerPart, integer, intent(in) nnodes)`

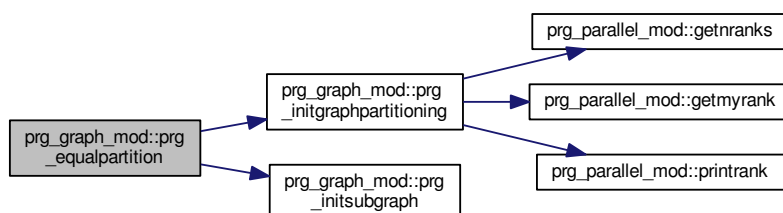
Create equal graph partitions, based on number of rows/orbitals.

Parameters

<i>gp</i>	Graph partitioning'
<i>nodesPerPart</i>	Number of core nodes per partition
<i>nnodes</i>	Total nodes in Hamiltonian matrix

Definition at line 355 of file `prg_graph_mod.F90`.

Here is the call graph for this function:



9.7.2.5 subroutine, public `prg_graph_mod::prg_filepartition (type (graph_partitioning_t), intent(inout) gp, character(len=*), intent(in) partFile)`

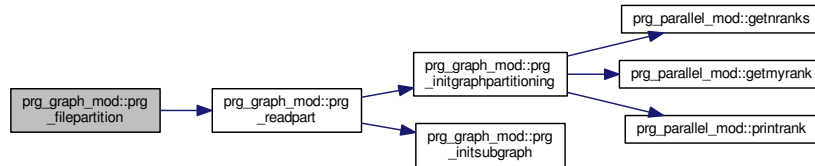
Read graph partitions from a file, based on number of rows/orbitals.

Parameters

<i>partFile</i>	File containing core nodes for each partition
<i>gp</i>	Graph partitioning

Definition at line 463 of file prg_graph_mod.F90.

Here is the call graph for this function:



9.7.2.6 subroutine, public prg_graph_mod::prg_fnormgraph (type(graph_partitioning_t), intent(inout) gp)

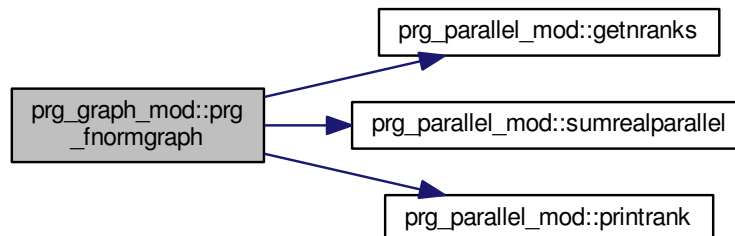
Accumulate trace norm across all subgraphs.

Parameters

<i>gp</i>	Graph partitioning
-----------	--------------------

Definition at line 516 of file prg_graph_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.7 subroutine, public prg_graph_mod::prg_initgraphpartitioning (type (graph_partitioning_t), intent(inout) *gp*, character(len=*), intent(in) *pname*, integer, intent(in) *np*, integer, intent(in) *nnodes*, integer, intent(in) *nnodes2*)

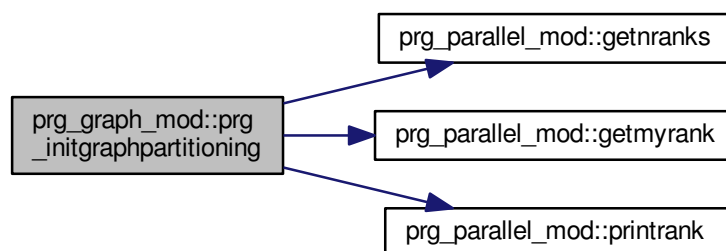
Initialize graph partitioning.

Parameters

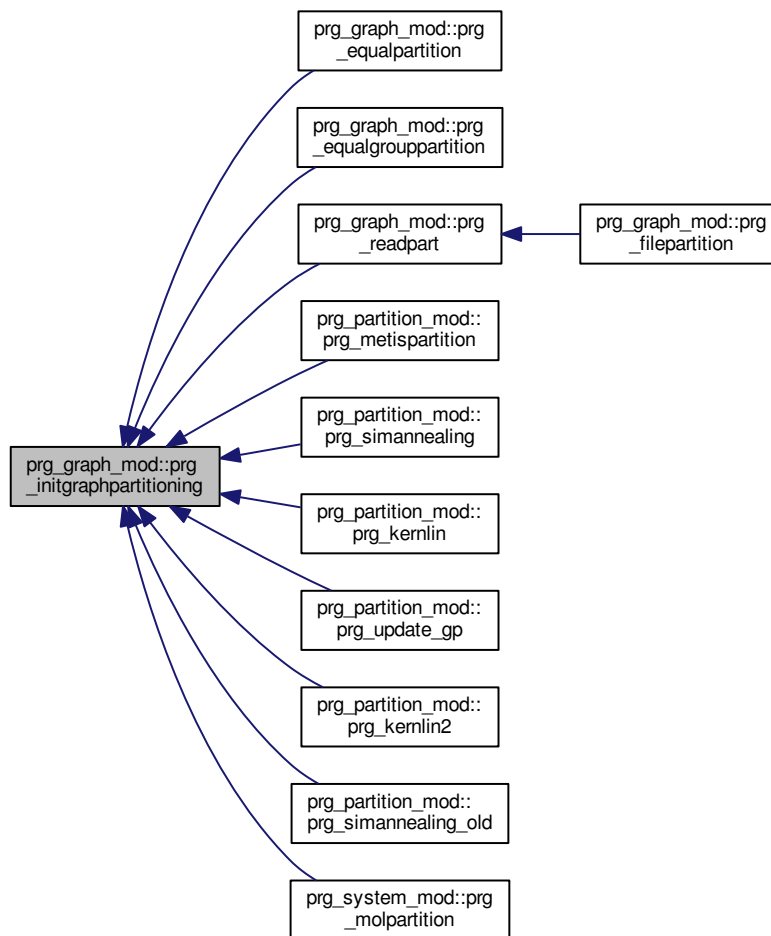
<i>gp</i>	Graph partitioning
<i>pname</i>	Partitioning name
<i>np</i>	Number of partitions
<i>nnodes</i>	Number of groups/nodes
<i>nnodes2</i>	Number of nodes

Definition at line 175 of file prg_graph_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.8 subroutine, public `prg_graph_mod::prg_initsubgraph (type (subgraph_t), intent(inout) sg, integer, intent(in) pnum, integer, intent(in) hsize)`

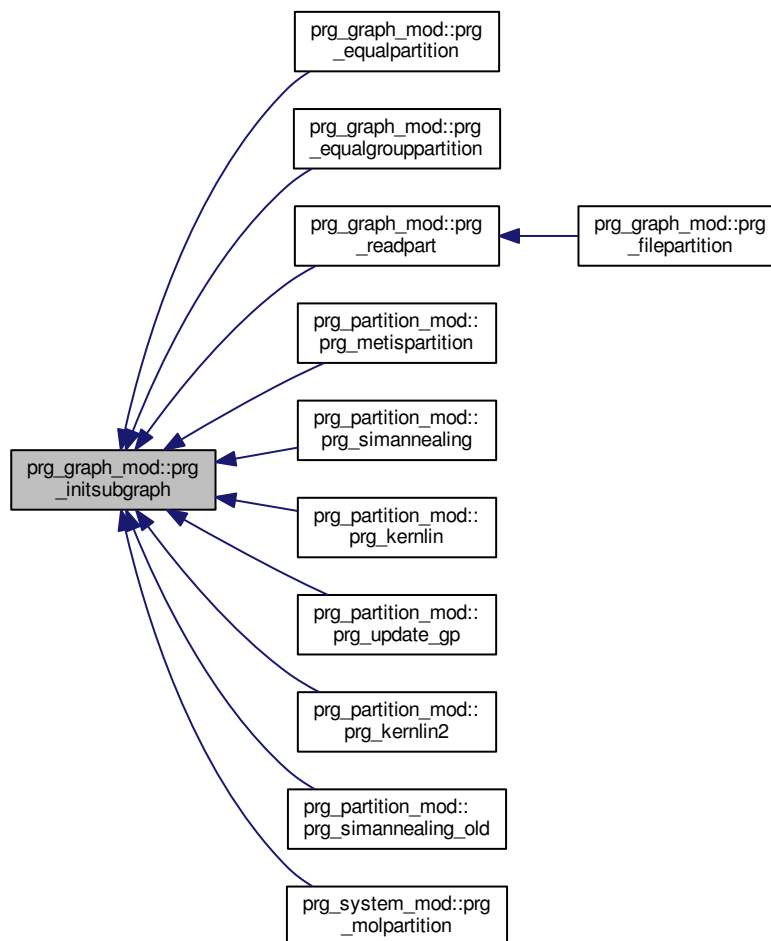
Initialize subgraph.

Parameters

<i>sg</i>	Subgraph
<i>pnum</i>	Part number
<i>hsize</i>	Size of full matrix

Definition at line 143 of file `prg_graph_mod.F90`.

Here is the caller graph for this function:



9.7.2.9 subroutine, public `prg_graph_mod::prg_printgraphpartitioning (type (graph_partitioning_t), intent(in) gp)`

Print graph partitioning structure data.

Parameters

<i>gp</i>	Graph partitioning
-----------	--------------------

Definition at line 292 of file `prg_graph_mod.F90`.

9.7.2.10 subroutine `prg_graph_mod::prg_readpart (type (graph_partitioning_t), intent(inout) gp, character(len=*) intent(in) partFile) [private]`

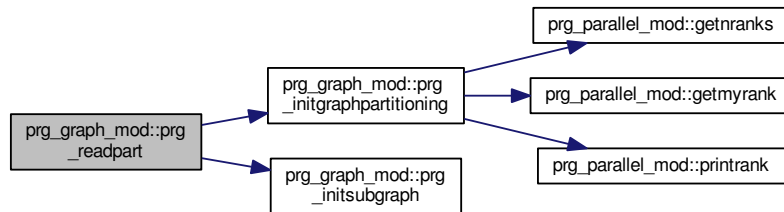
Read parts (core) from part file.

Parameters

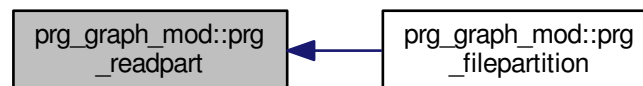
<i>gp</i>	Graph partitioning
<i>partFile</i>	Partition file

Definition at line 475 of file prg_graph_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.3 Variable Documentation

9.7.3.1 integer, parameter `prg_graph_mod::dp = kind(1.0d0)` `[private]`

Definition at line 16 of file prg_graph_mod.F90.

9.8 prg_graphsp2parser_mod Module Reference

Graph partitioning SP2 parser.

Data Types

- type [gsp2data_type](#)
General SP2 solver type.

Functions/Subroutines

- subroutine, public `prg_parse_gsp2` (gsp2data, filename)
The parser for SP2 solver.

Variables

- integer, parameter `dp = kind(1.0d0)`

9.8.1 Detailed Description

Graph partitioning SP2 parser.

This module is used to parse all the necessary input variables for graph-based SP2 electronic structure solver. Adding a new input keyword to the parser:

- If the variable is real, we have to increase `nkey_re`.
- Add the keyword (character type) in the `keyvector_re` vector.
- Add a default value (real type) in the `valvector_re`.
- Define a new variable and pass the value through `valvector_re(num)` where `num` is the position of the new keyword in the vector.

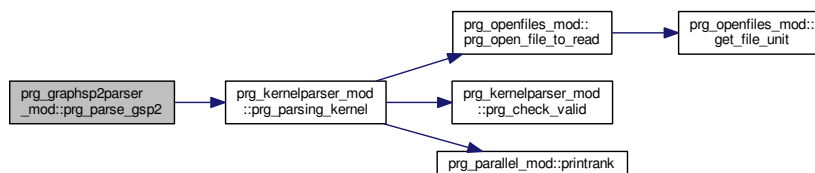
9.8.2 Function/Subroutine Documentation

9.8.2.1 subroutine, public `prg_graphsp2parser_mod::prg_parse_gsp2` (`type(gsp2data_type)`, intent(inout) `gsp2data`, character(len=*) `filename`)

The parser for SP2 solver.

Definition at line 62 of file `prg_graphsp2parser_mod.F90`.

Here is the call graph for this function:



9.8.3 Variable Documentation

9.8.3.1 integer, parameter `prg_graphsp2parser_mod::dp = kind(1.0d0)` [private]

Definition at line 22 of file `prg_graphsp2parser_mod.F90`.

9.9 prg_homolumo_mod Module Reference

The homolumo module.

Functions/Subroutines

- subroutine, public [prg_homolumogap](#) (vv, imax, pp, mineval, maxeval, ehomo, elumo, egap, verbose)
- subroutine, public [prg_sp2sequence](#) (pp, imax, mineval, maxeval, ehomo, elumo, errlimit, verbose)

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.9.1 Detailed Description

The homolumo module.

9.9.2 Function/Subroutine Documentation

9.9.2.1 subroutine, public `prg_homolumo_mod::prg_homolumogap` (`real(dp)`, `dimension(:)`, `intent(in)` `vv`, `integer`, `intent(in)` `imax`, `integer`, `dimension(:)`, `intent(in)` `pp`, `real(dp)`, `intent(in)` `mineval`, `real(dp)`, `intent(in)` `maxeval`, `real(dp)`, `intent(inout)` `ehomo`, `real(dp)`, `intent(inout)` `elumo`, `real(dp)`, `intent(inout)` `egap`, `integer`, `intent(in)`, optional `verbose`)

Definition at line 24 of file `prg_homolumo_mod.F90`.

9.9.2.2 subroutine, public `prg_homolumo_mod::prg_sp2sequence` (`integer`, `dimension(:)`, `intent(inout)` `pp`, `integer`, `intent(inout)` `imax`, `real(dp)`, `intent(in)` `mineval`, `real(dp)`, `intent(in)` `maxeval`, `real(dp)`, `intent(in)` `ehomo`, `real(dp)`, `intent(in)` `elumo`, `real(dp)`, `intent(in)` `errlimit`, `integer`, `intent(in)`, optional `verbose`)

Definition at line 94 of file `prg_homolumo_mod.F90`.

9.9.3 Variable Documentation

9.9.3.1 integer, parameter `prg_homolumo_mod::dp` = kind(1.0d0) [private]

Definition at line 14 of file `prg_homolumo_mod.F90`.

9.10 prg_implicit_fermi_mod Module Reference

Functions/Subroutines

- subroutine, public [prg_implicit_fermi](#) (h_bml, xi0_bml, p_bml, nsteps, nocc, mu, beta, osteps, occErrLimit, threshold)
Recursive Implicit Fermi Dirac.

Variables

- integer, parameter `dp = kind(1.0d0)`

9.10.1 Function/Subroutine Documentation

9.10.1.1 subroutine, public `prg_implicit_fermi_mod::prg_implicit_fermi (type(bml_matrix_t), intent(in) h_bml, type(bml_matrix_t), intent(inout) xi0_bml, type(bml_matrix_t), intent(inout) p_bml, integer, intent(in) nsteps, real(dp), intent(in) nocc, real(dp), intent(inout) mu, real(dp), intent(in) beta, integer, intent(in) osteps, real(dp), intent(in) occErrLimit, real(dp), intent(in) threshold)`

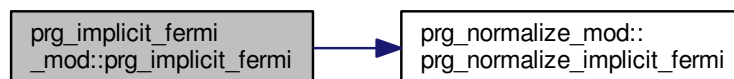
Recursive Implicit Fermi Dirac.

Parameters

<i>h_bml</i>	Input Hamiltonian matrix.
<i>xi0_bml</i>	Initial guess of first inverse.
<i>p_bml</i>	Output density matrix.
<i>nsteps</i>	Number of sp2 iterations.
<i>nocc</i>	Number of occupied states.
<i>mu</i>	Shifted chemical potential
<i>beta</i>	Input inverse temperature.
<i>osteps</i>	Outer loop steps to converge chemical potential
<i>occErrLimit</i>	Occupation error limit.
<i>threshold</i>	Threshold for multiplication.

Definition at line 39 of file `prg_implicit_fermi_mod.F90`.

Here is the call graph for this function:



9.10.2 Variable Documentation

9.10.2.1 integer, parameter `prg_implicit_fermi_mod::dp = kind(1.0d0)` `[private]`

Definition at line 20 of file `prg_implicit_fermi_mod.F90`.

9.11 prg_initmatrices_mod Module Reference

Initialization module.

Functions/Subroutines

- subroutine, public [prg_init_hsmat](#) (ham_bml, over_bml, bml_type, mdim, norb)
Initialize Hamiltonian and Overlap Matrix.
- subroutine, public [prg_init_pzmat](#) (rho_bml, zmat_bml, bml_type, mdim, norb)
Initialize Density matrix and Inverse square root Overlap.
- subroutine, public [prg_init_ortho](#) (orthoh_bml, orthop_bml, bml_type, mdim, norb)
Initialize The orthogonal versions of Hamiltonian and Density Matrix.

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.11.1 Detailed Description

Initialization module.

Routines in this module are used to prg_initialize several matrices that will be used in the code.

9.11.2 Function/Subroutine Documentation

9.11.2.1 subroutine, public prg_initmatrices_mod::prg_init_hsmat (type(bml_matrix_t), intent(inout) ham_bml, type(bml_matrix_t), intent(inout) over_bml, character(20) bml_type, integer, intent(inout) mdim, integer, intent(in) norb)

Initialize Hamiltonian and Overlap Matrix.

Allocation of the Hamiltonian and Overlap matrix into bml formats.

Parameters

<i>ham_bml</i>	Hamiltonian in bml format.
<i>over_bml</i>	Overlap in bml format.
<i>threshold</i>	Threshold value for matrix elements.
<i>mdim</i>	Max nonzero elements per row for every row see [1] .
<i>norb</i>	Total number of orbitals.

Definition at line 28 of file prg_initmatrices_mod.F90.

9.11.2.2 subroutine, public prg_initmatrices_mod::prg_init_ortho (type(bml_matrix_t), intent(inout) orthoh_bml, type(bml_matrix_t), intent(inout) orthop_bml, character(20) bml_type, integer, intent(inout) mdim, integer, intent(in) norb)

Initialize The orthogonal versions of Hamiltonian and Density Matrix.

Allocation of the orthogonal Hamiltonian and Density matrix into bml formats.

Parameters

<i>orthoh_bml</i>	Orthogonal Hamiltonian in bml format.
<i>orthop_bml</i>	Orthogonal Density Matrix in bml format.
<i>threshold</i>	Threshold value for matrix elements.
<i>mdim</i>	Max nonzero elements per row for every row see [1] .
<i>norb</i>	Total number of orbitals.

Definition at line 70 of file prg_initmatrices_mod.F90.

```
9.11.2.3  subroutine, public prg_initmatrices_mod::prg_init_pzmat ( type(bml_matrix_t), intent(inout) rho_bml,
                                type(bml_matrix_t), intent(inout) zmat_bml, character(20) bml_type, integer, intent(inout) mdim, integer, intent(in) norb
                                )
```

Initialize Density matrix and Inverse square root Overlap.

Allocation of the Density matrix and Inverse square root Overlap matrix into bml formats.

Parameters

<i>rho_bml</i>	Density matrix in bml format.
<i>zmat_bml</i>	Inverse square root Overlap in bml format.
<i>threshold</i>	Threshold value for matrix elements.
<i>mdim</i>	Max nonzero elements per row for every row see [1] .
<i>norb</i>	Total number of orbitals.

Definition at line 49 of file prg_initmatrices_mod.F90.

9.11.3 Variable Documentation

```
9.11.3.1  integer, parameter prg_initmatrices_mod::dp = kind(1.0d0)  [private]
```

Definition at line 14 of file prg_initmatrices_mod.F90.

9.12 prg_kernelparser_mod Module Reference

Some general parsing functions.

Functions/Subroutines

- subroutine, public [prg_parsing_kernel](#) (keyvector_char, valvector_char, keyvector_int, valvector_int, keyvector_re, valvector_re, keyvector_log, valvector_log, filename, startstop)
The general parsing function. It is used to vectorize a set of "keywords" "value" pairs as included in a general input file.
- subroutine [prg_check_valid](#) (invalidc)
Check for valid keywords (checks for an = sign)

Variables

- integer, parameter `dp` = kind(1.0d0)

9.12.1 Detailed Description

Some general parsing functions.

Author

C. F. A. Negre (cnegre@lanl.gov)

9.12.2 Function/Subroutine Documentation

9.12.2.1 subroutine `prg_kernelparser_mod::prg_check_valid` (`character(len=*)`, `intent(in) invalidc`) [private]

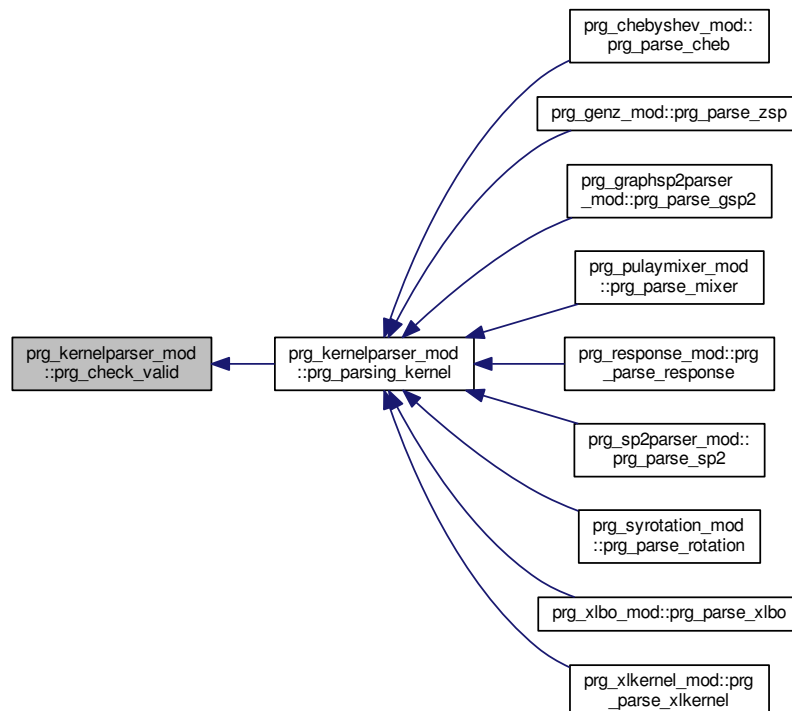
Check for valid keywords (checks for an = sign)

Parameters

<code>invalidc</code>	Keyword to check.
-----------------------	-------------------

Definition at line 396 of file `prg_kernelparser_mod.F90`.

Here is the caller graph for this function:



9.12.2.2 subroutine, public prg_kernelparser_mod::prg_parsing_kernel (character(50), dimension(:) *keyvector_char*, character(100), dimension(:) *valvector_char*, character(50), dimension(:) *keyvector_int*, integer, dimension(:) *valvector_int*, character(50), dimension(:) *keyvector_re*, real(dp), dimension(:) *valvector_re*, character(50), dimension(:) *keyvector_log*, logical, dimension(:) *valvector_log*, character(len=*) *filename*, character(len=*) , dimension(2), intent(in), optional *startstop*)

The general parsing function. It is used to vectorize a set of "keywords" "value" pairs as included in a general input file.

Note

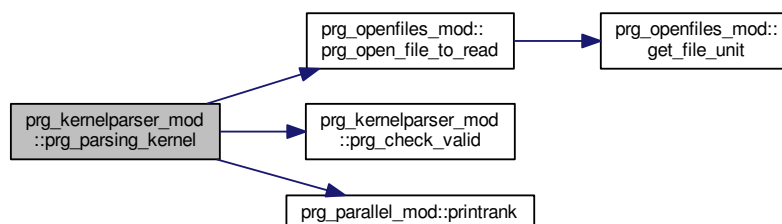
This parsing strategy can only parse a file of 500 lines by 500 words.

Warning

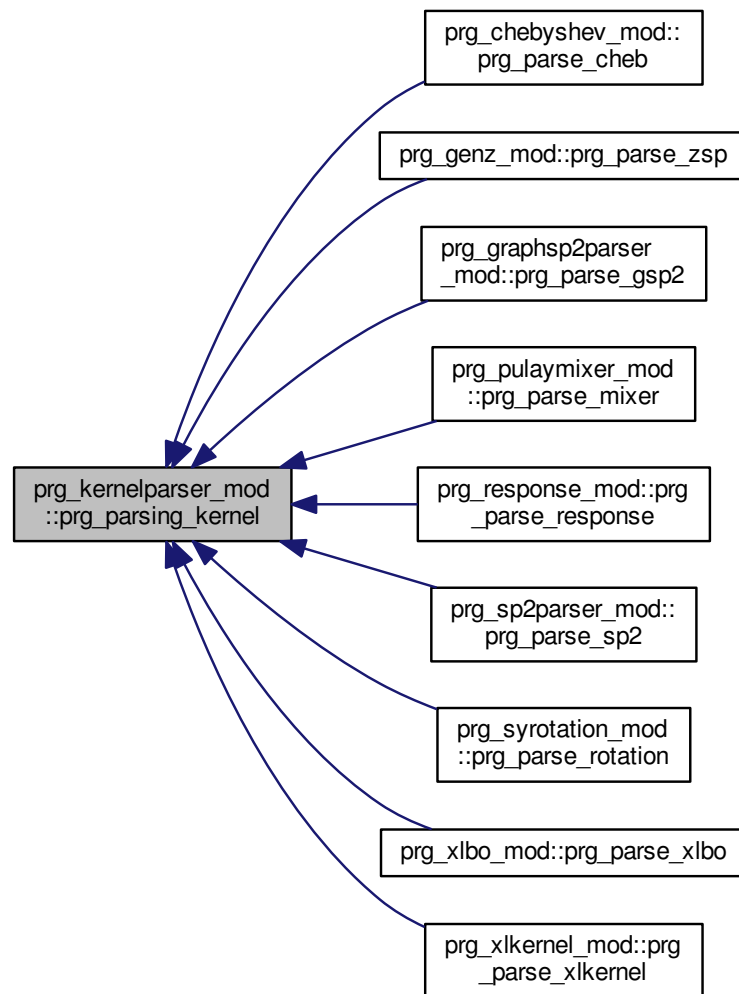
If the length of variable vect is changed, this could produce a segmentation fault.

Definition at line 33 of file prg_kernelparser_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.12.3 Variable Documentation

9.12.3.1 integer, parameter prg_kernelparser_mod::dp = kind(1.0d0) [private]

Definition at line 16 of file prg_kernelparser_mod.F90.

9.13 prg_nonortho_mod Module Reference

Module to prg_orthogonalize and prg_deorthogonalize any operator.

Functions/Subroutines

- subroutine, public `prg_orthogonalize` (`A_bml`, `zmat_bml`, `orthoA_bml`, `threshold`, `bml_type`, `verbose`)
This routine performs: $A_{ortho} = Z^\dagger A Z$.
- subroutine, public `prg_deorthogonalize` (`orthoA_bml`, `zmat_bml`, `a_bml`, `threshold`, `bml_type`, `verbose`)
This routine performs: $A = Z A_{ortho} Z^\dagger$.

Variables

- integer, parameter `dp` = kind(1.0d0)

9.13.1 Detailed Description

Module to `prg_orthogonalize` and `prg_deorthogonalize` any operator.

Typically the Hamiltonin needs to be `prg_orthogonalized`: $H_{ortho} = Z^\dagger H Z$

Also, if the density matrix was obtained from the `prg_orthogonalized` Hamiltonian, it can be `prg_deorthogonalized` as: $\rho = Z \rho_{ortho} Z^\dagger$

9.13.2 Function/Subroutine Documentation

- 9.13.2.1 subroutine, public `prg_nonortho_mod::prg_deorthogonalize` (`type(bml_matrix_t)`, `intent(in) orthoA_bml`, `type(bml_matrix_t)`, `intent(in) zmat_bml`, `type(bml_matrix_t)`, `intent(inout) a_bml`, `real(dp) threshold`, `character(len=*) bml_type`, `integer verbose`)

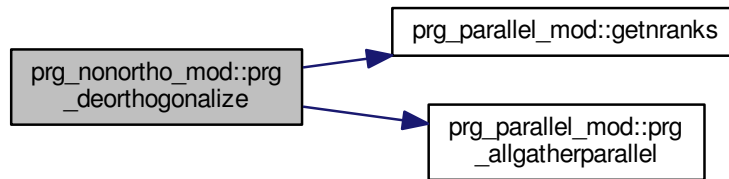
This routine performs: $A = Z A_{ortho} Z^\dagger$.

Parameters

<code>orthoA_bml</code>	Matrix to be <code>prg_deorthogonalized</code> .
<code>zmat_bml</code>	Congruence transform to be used.
<code>A_bml</code>	Matrix resulting from the <code>prg_deorthogonalized</code> in <code>bml</code> format.
<code>threshold</code>	Threshold value to be used in the matrix-matrix operations.
<code>bml_type</code>	<code>bml</code> format to be used.
<code>verbose</code>	Verbosity level.

Definition at line 82 of file `prg_nonortho_mod.F90`.

Here is the call graph for this function:



9.13.2.2 subroutine, public `prg_nonortho_mod::prg_orthogonalize` (`type(bml_matrix_t)`, intent(inout) `A_bml`, `type(bml_matrix_t)`, intent(inout) `zmat_bml`, `type(bml_matrix_t)`, intent(inout) `orthoA_bml`, `real(dp)`, intent(in) `threshold`, `character(len=*)`, intent(in) `bml_type`, `integer`, intent(in) `verbose`)

This routine performs: $A_{ortho} = Z^{\dagger} A Z$.

Parameters

<i>A_bml</i>	Matrix to be <code>prg_orthogonalized</code> in bml format.
<i>zmat_bml</i>	Congruence transform to be used.
<i>orthoA_bml</i>	Matrix resulting from the orthogonalization.
<i>threshold</i>	Threshold value to be used in the matrix-matrix operations.
<i>bml_type</i>	bml format to be used.
<i>verbose</i>	Verbosity level.

Definition at line 36 of file `prg_nonortho_mod.F90`.

9.13.3 Variable Documentation

9.13.3.1 `integer`, parameter `prg_nonortho_mod::dp = kind(1.0d0)` [`private`]

Definition at line 19 of file `prg_nonortho_mod.F90`.

9.14 prg_normalize_mod Module Reference

The `prg_normalize` module.

Functions/Subroutines

- subroutine, public [prg_normalize](#) (h_bml)
Normalize a Hamiltonian matrix prior to running the SP2 algorithm.
- subroutine, public [prg_normalize_fermi](#) (h_bml, h1, hN, mu)
Normalize a Hamiltonian matrix prior to running the truncated SP2 algorithm.
- subroutine, public [prg_normalize_implicit_fermi](#) (h_bml, cnst, mu)
Normalize a Hamiltonian matrix prior to running the implicit fermi dirac algorithm.
- subroutine, public [prg_gershgorinreduction](#) (gp)
Determine gershgorin bounds across all parts, local and distributed.
- subroutine, public [prg_normalize_chen](#) (h_bml, mu, emin, emax, alpha, scaledmu)
Normalize a Hamiltonian matrix prior to running the Chebyshev algorithm.

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.14.1 Detailed Description

The prg_normalize module.

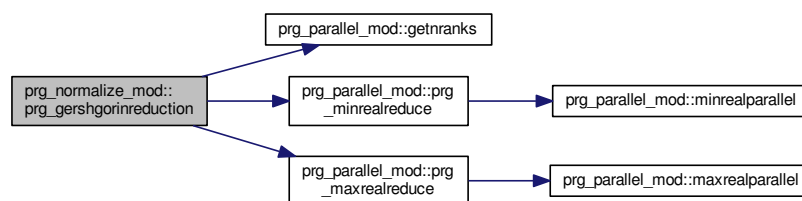
9.14.2 Function/Subroutine Documentation

9.14.2.1 subroutine, public `prg_normalize_mod::prg_gershgorinreduction (type(graph_partitioning_t), intent(inout) gp)`

Determine gershgorin bounds across all parts, local and distributed.

Definition at line 101 of file prg_normalize_mod.F90.

Here is the call graph for this function:



9.14.2.2 subroutine, public `prg_normalize_mod::prg_normalize (type(bml_matrix_t), intent(inout) h_bml)`

Normalize a Hamiltonian matrix prior to running the SP2 algorithm.

$$X0 = (e_{\max} * I - H) / (e_{\max} - e_{\min})$$

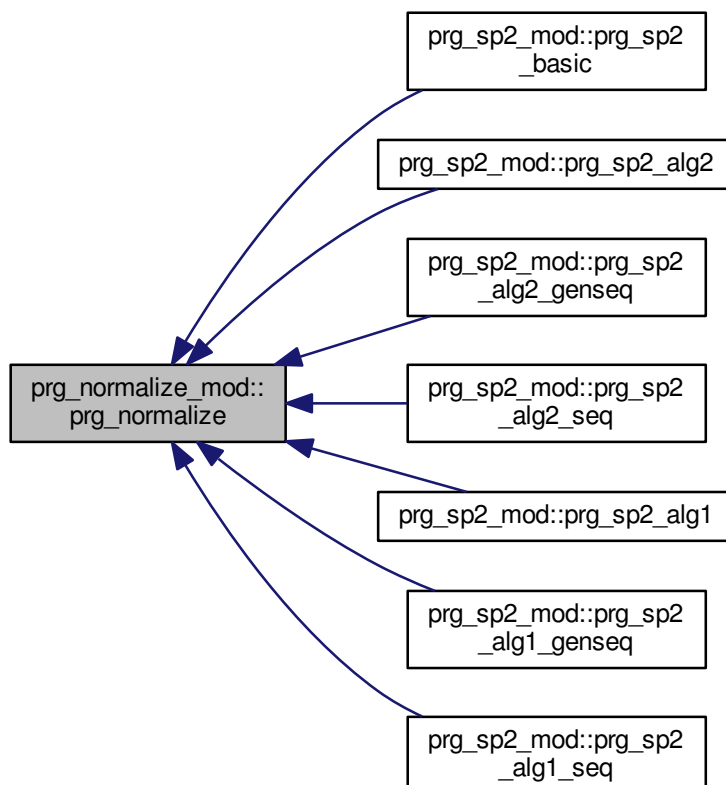
where e_{\max} and e_{\min} are obtained using the Gershgorin circle theorem.

Parameters

<i>h_bml</i>	Input/Output Hamiltonian matrix
--------------	---------------------------------

Definition at line 33 of file prg_normalize_mod.F90.

Here is the caller graph for this function:



```

9.14.2.3  subroutine, public prg_normalize_mod::prg_normalize_cheb ( type(bml_matrix_t), intent(inout) h_bml, real(dp),
intent(in) mu, real(dp), intent(in) emin, real(dp), intent(in) emax, real(dp), intent(inout) alpha, real(dp), intent(inout)
scaledmu )

```

Normalize a Hamiltonian matrix prior to running the Chebyshev algorithm.

$$X0 = 2*(H - e_min*I) / (e_max - e_min) - I$$

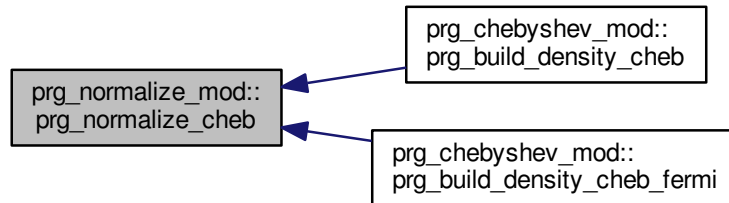
where `e_max` and `e_min` are obtained sing the Gershgorin circle theorem.

Parameters

<i>h_bml</i>	Input/Output Hamiltonian matrix
--------------	---------------------------------

Definition at line 130 of file prg_normalize_mod.F90.

Here is the caller graph for this function:



9.14.2.4 subroutine, public prg_normalize_mod::prg_normalize_fermi (type(bml_matrix_t), intent(inout) *h_bml*, real(dp), intent(in) *h1*, real(dp), intent(in) *hN*, real(dp), intent(in) *mu*)

Normalize a Hamiltonian matrix prior to running the truncated SP2 algorithm.

$$X0 = ((hN - \mu) * I - H) / (hN - h1) \text{ or } X0 = (hN * I - H0 - \mu * I) / (hN - h1)$$

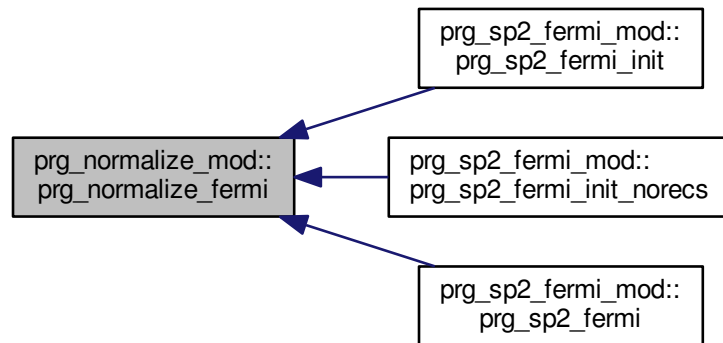
where *h1* and *hN* are scaled Gershgorin bounds.

Parameters

<i>H_bml</i>	Hamiltonian matrix
<i>h1</i>	Scaled minimum Gershgorin bound.
<i>hN</i>	Scaled maximum Gershgorin bound.
<i>mu</i>	Chemical potential

Definition at line 63 of file prg_normalize_mod.F90.

Here is the caller graph for this function:



9.14.2.5 subroutine, public prg_normalize_mod::prg_normalize_implicit_fermi (type(bml_matrix_t), intent(inout) *h_bml*, real(dp), intent(in) *cnst*, real(dp), intent(in) *mu*)

Normalize a Hamiltonian matrix prior to running the implicit fermi dirac algorithm.

$X0 = 0.5 * I - cnst * (H0 - \mu_0 * I)$ or $X0 = (0.5 + cnst * \mu_0) * I - cnst * H0$

Parameters

<i>H_bml</i>	Hamiltonian matrix
<i>cnst</i>	Constant based on beta and steps
<i>mu</i>	Chemical potential

Definition at line 87 of file prg_normalize_mod.F90.

Here is the caller graph for this function:



9.14.3 Variable Documentation

9.14.3.1 integer, parameter prg_normalize_mod::dp = kind(1.0d0) [private]

Definition at line 15 of file prg_normalize_mod.F90.

9.15 prg_openfiles_mod Module Reference

Module to handle input output files for the PROGRESS lib.

Functions/Subroutines

- integer function, public [get_file_unit](#) (io_max)
Returns a unit number that is not in use.
- subroutine, public [prg_open_file](#) (io, name)
Opens a file to write.
- subroutine, public [prg_open_file_to_read](#) (io, name)
Opens a file to read.

9.15.1 Detailed Description

Module to handle input output files for the PROGRESS lib.

9.15.2 Function/Subroutine Documentation

9.15.2.1 integer function, public prg_openfiles_mod::get_file_unit (integer io_max)

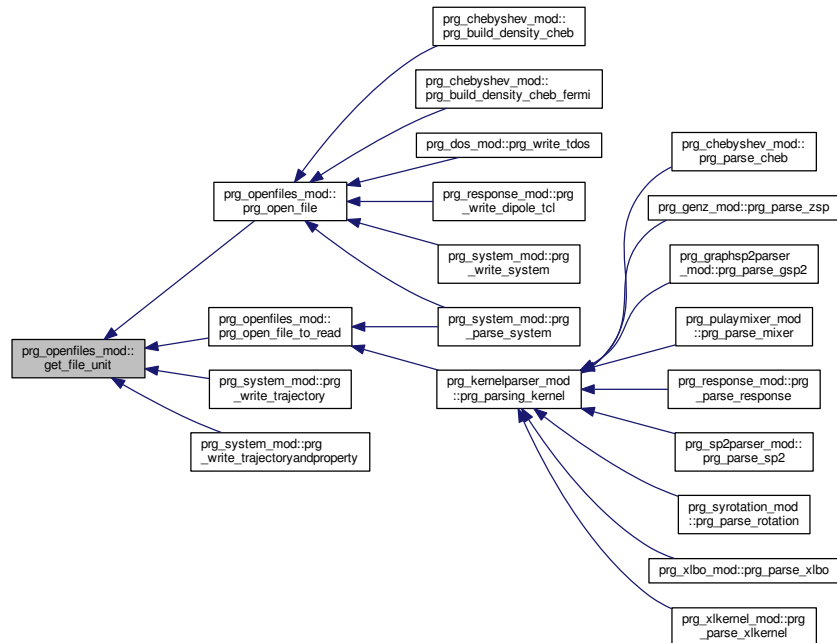
Returns a unit number that is not in use.

Parameters

<i>io_max</i>	Maximum units to search.
<i>get_file_unit</i>	Unit return to use for the file.

Definition at line 19 of file prg_openfiles_mod.F90.

Here is the caller graph for this function:



9.15.2.2 subroutine, public prg_openfiles_mod::prg_open_file (integer *io*, character(len=*) *name*)

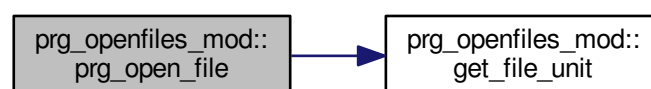
Opens a file to write.

Parameters

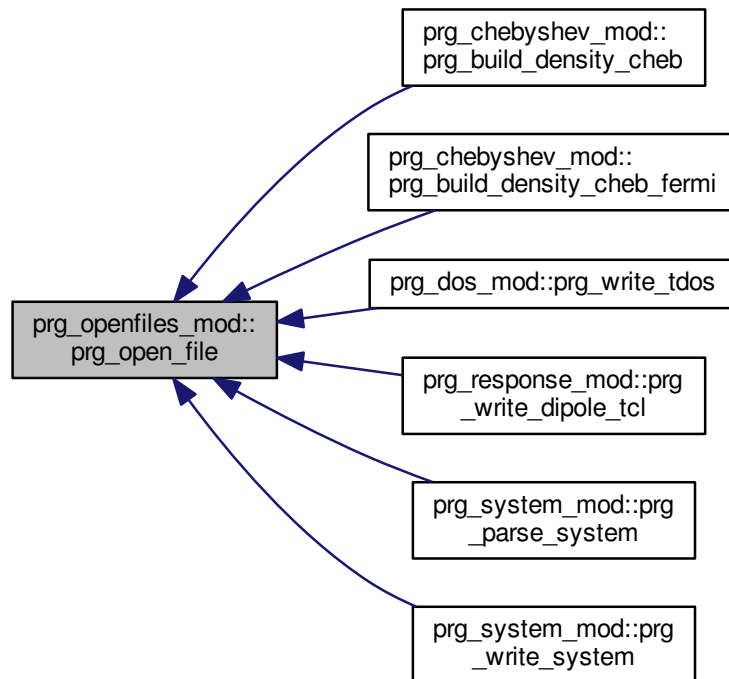
<i>io</i>	Unit for the file.
<i>name</i>	Name of the file.

Definition at line 38 of file prg_openfiles_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.15.2.3 subroutine, public `prg_openfiles_mod::prg_open_file_to_read (integer io, character(len=*) name)`

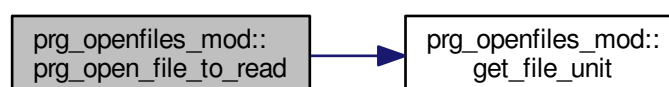
Opens a file to read.

Parameters

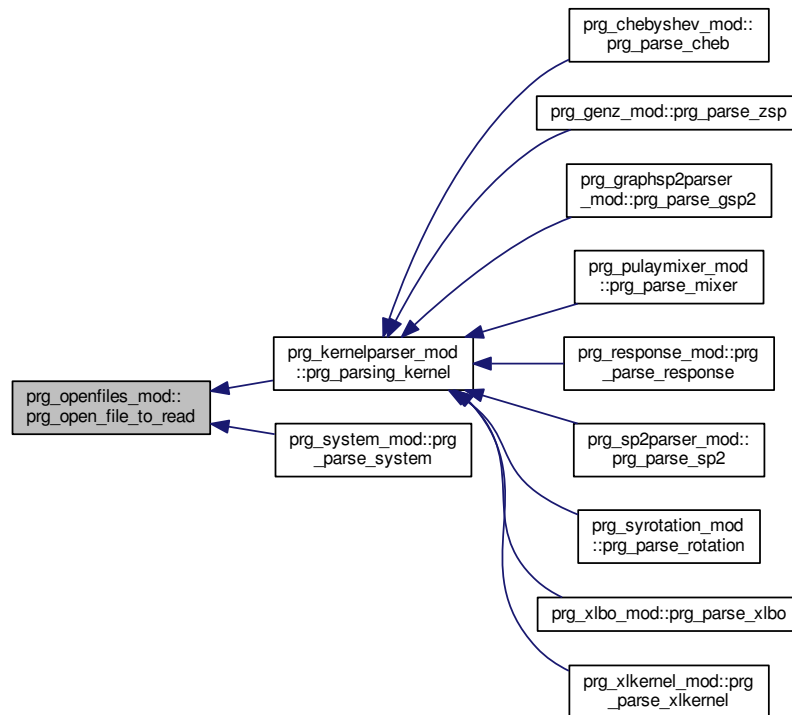
<i>io</i>	Unit for the file.
<i>name</i>	Name of the file.

Definition at line 54 of file `prg_openfiles_mod.F90`.

Here is the call graph for this function:



Here is the caller graph for this function:



9.16 prg_parallel_mod Module Reference

The parallel module.

Data Types

- type [rankreducedata_t](#)
Data structure for rection over MPI ranks.

Functions/Subroutines

- integer function, public [getnranks](#) ()
- integer function, public [getmyrank](#) ()
- integer function, public [printrank](#) ()
- subroutine, public [prg_initparallel](#) ()
- subroutine, public [prg_shutdownparallel](#) ()
- integer function [saverequest](#) (irequest)
- subroutine, public [prg_barrierparallel](#) ()
- subroutine, public [sendreceiveparallel](#) (sendBuf, sendLen, dest, recvBuf, recvLen, source, nreceived)
- subroutine, public [isendparallel](#) (sendBuf, sendLen, dest)
- subroutine, public [sendparallel](#) (sendBuf, sendLen, dest)

- subroutine, public [prg_iprg_recvparallel](#) (recvBuf, recvLen, rind)
- subroutine, public [prg_recvparallel](#) (recvBuf, recvLen)
- subroutine, public [sumintparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [sumrealparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [maxintparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [maxrealparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [minintparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [minrealparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [prg_minrealreduce](#) (rvalue)
- subroutine, public [prg_maxrealreduce](#) (rvalue)
- subroutine, public [prg_maxintreduce2](#) (value1, value2)
- subroutine, public [prg_sumintreduce2](#) (value1, value2)
- subroutine, public [prg_sumrealreduce](#) (value1)
- subroutine, public [prg_sumrealreduce2](#) (value1, value2)
- subroutine, public [prg_sumrealreduce3](#) (value1, value2, value3)
- subroutine, public [prg_sumrealreducen](#) (valueVec, N)
- subroutine, public [prg_sumintreducen](#) (valueVec, N)
- subroutine, public [minrankrealparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [maxrankrealparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [prg_bcastparallel](#) (buf, blen, root)
- subroutine, public [allgatherrealparallel](#) (sendBuf, sendLen, recvBuf, recvLen)
- subroutine, public [allgatherintparallel](#) (sendBuf, sendLen, recvBuf, recvLen)
- subroutine, public [allgathervrealparallel](#) (sendBuf, sendLen, recvBuf, recvLen, recvDispl)
- subroutine, public [allgathervintparallel](#) (sendBuf, sendLen, recvBuf, recvLen, recvDispl)
- subroutine, public [prg_allsumrealreduceparallel](#) (buf, buflen)
- subroutine, public [prg_allsumintreduceparallel](#) (buf, buflen)
- subroutine, public [prg_allgatherparallel](#) (a)
- subroutine, public [prg_wait](#) ()

Variables

- integer, parameter [dp](#) = kind(1.0d0)
- integer [myrank](#)
- integer [nranks](#)
- integer [ierr](#)
- integer [reqcount](#)
- integer, dimension(:), allocatable [requestlist](#)
- integer, dimension(:), allocatable [rused](#)

9.16.1 Detailed Description

The parallel module.

9.16.2 Function/Subroutine Documentation

- 9.16.2.1 subroutine, public [prg_parallel_mod::allgatherintparallel](#) (integer, dimension(*), intent(in) *sendBuf*, integer, intent(in) *sendLen*, integer, dimension(*), intent(out) *recvBuf*, integer, intent(in) *recvLen*)

Definition at line 660 of file [prg_parallel_mod.F90](#).

9.16.2.2 subroutine, public prg_parallel_mod::allgatherrealparallel (real(dp), dimension(*), intent(in) *sendBuf*, integer, intent(in) *sendLen*, real(dp), dimension(*), intent(out) *recvBuf*, integer, intent(in) *recvLen*)

Definition at line 644 of file prg_parallel_mod.F90.

9.16.2.3 subroutine, public prg_parallel_mod::allgathervintparallel (integer, dimension(*), intent(in) *sendBuf*, integer, intent(in) *sendLen*, integer, dimension(*), intent(out) *recvBuf*, integer, dimension(*), intent(in) *recvLen*, integer, dimension(*), intent(in) *recvDispl*)

Definition at line 696 of file prg_parallel_mod.F90.

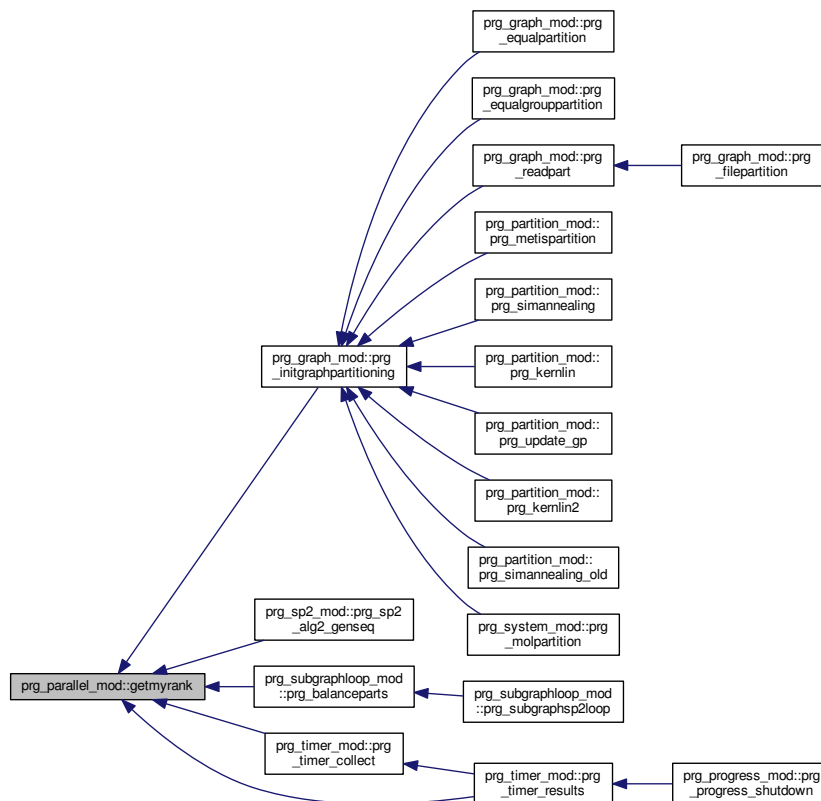
9.16.2.4 subroutine, public prg_parallel_mod::allgathervrealparallel (real(dp), dimension(*), intent(in) *sendBuf*, integer, intent(in) *sendLen*, real(dp), dimension(*), intent(out) *recvBuf*, integer, dimension(*), intent(in) *recvLen*, integer, dimension(*), intent(in) *recvDispl*)

Definition at line 676 of file prg_parallel_mod.F90.

9.16.2.5 integer function, public prg_parallel_mod::getmyrank ()

Definition at line 99 of file prg_parallel_mod.F90.

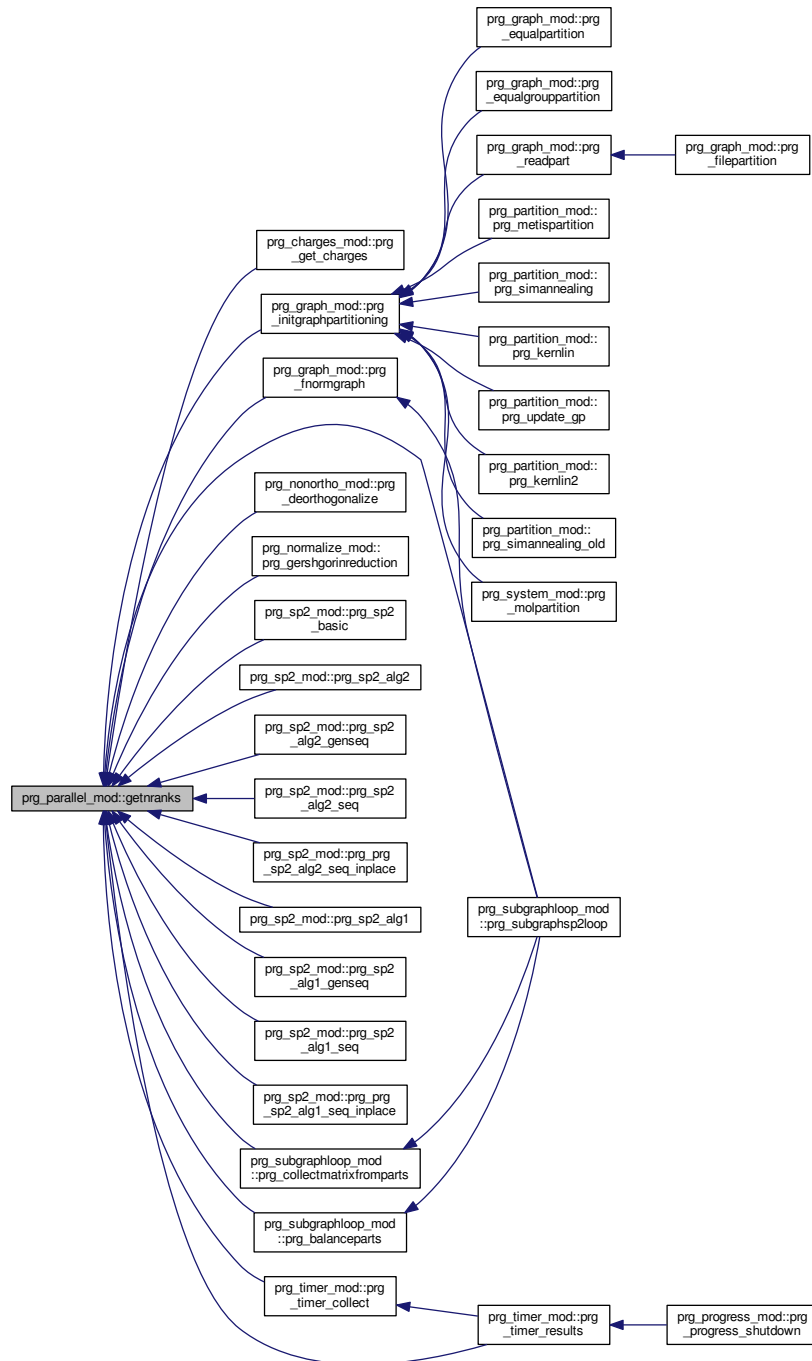
Here is the caller graph for this function:



9.16.2.6 integer function, public prg_parallel_mod::getnranks ()

Definition at line 88 of file prg_parallel_mod.F90.

Here is the caller graph for this function:



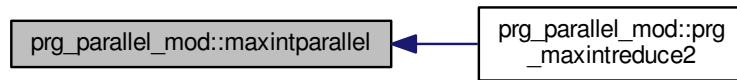
9.16.2.7 subroutine, public prg_parallel_mod::isendparallel (real(dp), dimension(*), intent(in) sendBuf, integer, intent(in) sendLen, integer, intent(in) dest)

Definition at line 230 of file prg_parallel_mod.F90.

9.16.2.8 subroutine, public prg_parallel_mod::maxintparallel (integer, dimension(*), intent(in) *sendBuf*, integer, dimension(*), intent(out) *recvBuf*, integer, intent(in) *icount*)

Definition at line 337 of file prg_parallel_mod.F90.

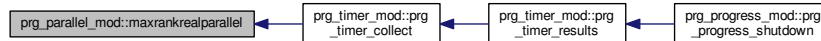
Here is the caller graph for this function:



9.16.2.9 subroutine, public prg_parallel_mod::maxrankrealparallel (type(rankreducedata_t), dimension(*), intent(in) *sendBuf*, type(rankreducedata_t), dimension(*), intent(out) *recvBuf*, integer, intent(in) *icount*)

Definition at line 607 of file prg_parallel_mod.F90.

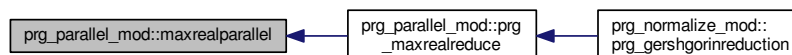
Here is the caller graph for this function:



9.16.2.10 subroutine, public prg_parallel_mod::maxrealparallel (real(dp), dimension(*), intent(in) *sendBuf*, real(dp), dimension(*), intent(out) *recvBuf*, integer, intent(in) *icount*)

Definition at line 358 of file prg_parallel_mod.F90.

Here is the caller graph for this function:



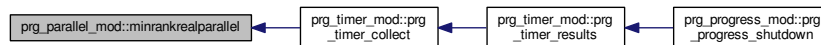
9.16.2.11 subroutine, public prg_parallel_mod::minintparallel (integer, dimension(*), intent(in) *sendBuf*, integer, dimension(*), intent(out) *recvBuf*, integer, intent(in) *icount*)

Definition at line 379 of file prg_parallel_mod.F90.

9.16.2.12 subroutine, public prg_parallel_mod::minrankrealparallel (type(rankreducedata_t), dimension(*), intent(in) *sendBuf*, type(rankreducedata_t), dimension(*), intent(out) *recvBuf*, integer, intent(in) *icount*)

Definition at line 584 of file prg_parallel_mod.F90.

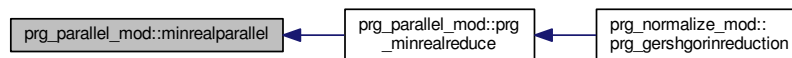
Here is the caller graph for this function:



9.16.2.13 subroutine, public prg_parallel_mod::minrealparallel (real(dp), dimension(*), intent(in) *sendBuf*, real(dp), dimension(*), intent(out) *recvBuf*, integer, intent(in) *icount*)

Definition at line 400 of file prg_parallel_mod.F90.

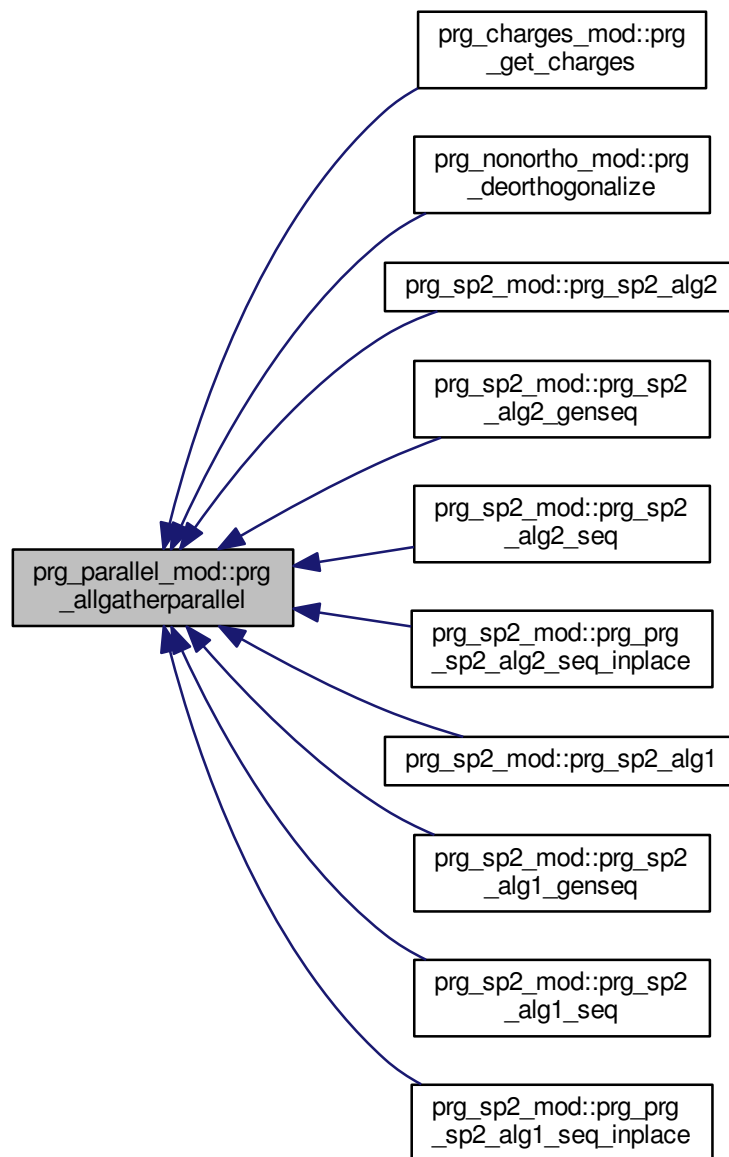
Here is the caller graph for this function:



9.16.2.14 subroutine, public prg_parallel_mod::prg_allgatherparallel (type(bml_matrix_t), intent(inout) *a*)

Definition at line 744 of file prg_parallel_mod.F90.

Here is the caller graph for this function:



9.16.2.15 subroutine, public prg_parallel_mod::prg_allsumintreduceparallel (integer, dimension(*), intent(inout) *buf*, integer, intent(in) *buflen*)

Definition at line 729 of file prg_parallel_mod.F90.

```
9.16.2.16 subroutine, public prg_parallel_mod::prg_allsumrealreduceparallel ( real(dp), dimension(*), intent(inout) buf, integer,
intent(in) buflen )
```

Definition at line 714 of file prg_parallel_mod.F90.

9.16.2.17 subroutine, public prg_parallel_mod::prg_barrierparallel ()

Definition at line 196 of file prg_parallel_mod.F90.

9.16.2.18 subroutine, public prg_parallel_mod::prg_bcastparallel (character, dimension(*), intent(in) *buf*, integer, intent(in) *blen*, integer, intent(in) *root*)

Definition at line 630 of file prg_parallel_mod.F90.

9.16.2.19 subroutine, public prg_parallel_mod::prg_initparallel ()

Definition at line 127 of file prg_parallel_mod.F90.

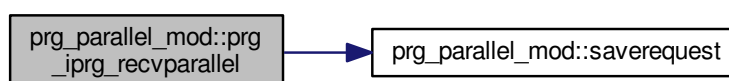
Here is the caller graph for this function:



9.16.2.20 subroutine, public prg_parallel_mod::prg_iprg_recvparallel (real(dp), dimension(*) *recvBuf*, integer, intent(in) *recvLen*, integer *rind*)

Definition at line 261 of file prg_parallel_mod.F90.

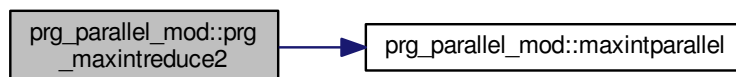
Here is the call graph for this function:



9.16.2.21 subroutine, public prg_parallel_mod::prg_maxintreduce2 (integer, intent(inout) *value1*, integer, intent(inout) *value2*)

Definition at line 453 of file prg_parallel_mod.F90.

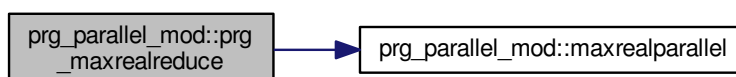
Here is the call graph for this function:



9.16.2.22 subroutine, public prg_parallel_mod::prg_maxrealreduce (real(dp), intent(inout) *rvalue*)

Definition at line 437 of file prg_parallel_mod.F90.

Here is the call graph for this function:



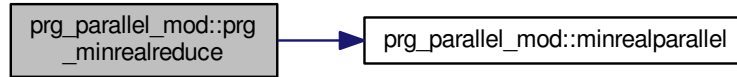
Here is the caller graph for this function:



9.16.2.23 subroutine, public prg_parallel_mod::prg_minrealreduce (real(dp), intent(inout) *rvalue*)

Definition at line 421 of file prg_parallel_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.16.2.24 subroutine, public prg_parallel_mod::prg_recvparallel (real(dp), dimension(*) *recvBuf*, integer, intent(in) *recvLen*)

Definition at line 279 of file prg_parallel_mod.F90.

9.16.2.25 subroutine, public prg_parallel_mod::prg_shutdownparallel ()

Definition at line 154 of file prg_parallel_mod.F90.

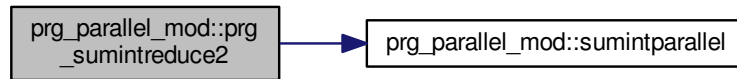
Here is the caller graph for this function:



9.16.2.26 subroutine, public prg_parallel_mod::prg_sumintreduce2 (integer, intent(inout) *value1*, integer, intent(inout) *value2*)

Definition at line 471 of file prg_parallel_mod.F90.

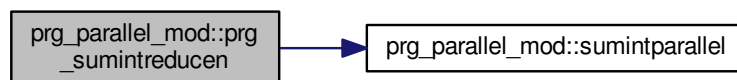
Here is the call graph for this function:



9.16.2.27 subroutine, public prg_parallel_mod::prg_sumintreducen (integer, dimension(n), intent(inout) *valueVec*, integer, intent(in) *N*)

Definition at line 564 of file prg_parallel_mod.F90.

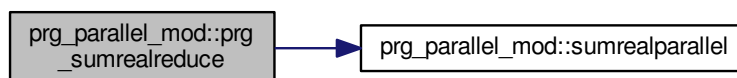
Here is the call graph for this function:



9.16.2.28 subroutine, public prg_parallel_mod::prg_sumrealreduce (real(dp), intent(inout) *value1*)

Definition at line 489 of file prg_parallel_mod.F90.

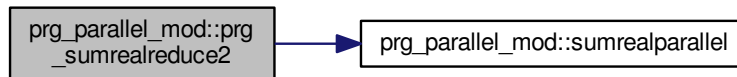
Here is the call graph for this function:



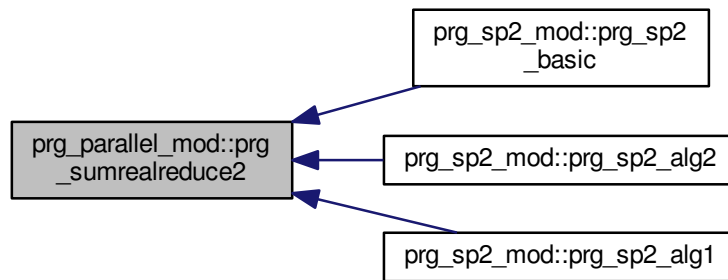
9.16.2.29 subroutine, public prg_parallel_mod::prg_sumrealreduce2 (real(dp), intent(inout) *value1*, real(dp), intent(inout) *value2*)

Definition at line 505 of file prg_parallel_mod.F90.

Here is the call graph for this function:



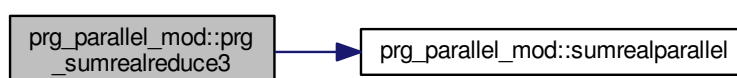
Here is the caller graph for this function:



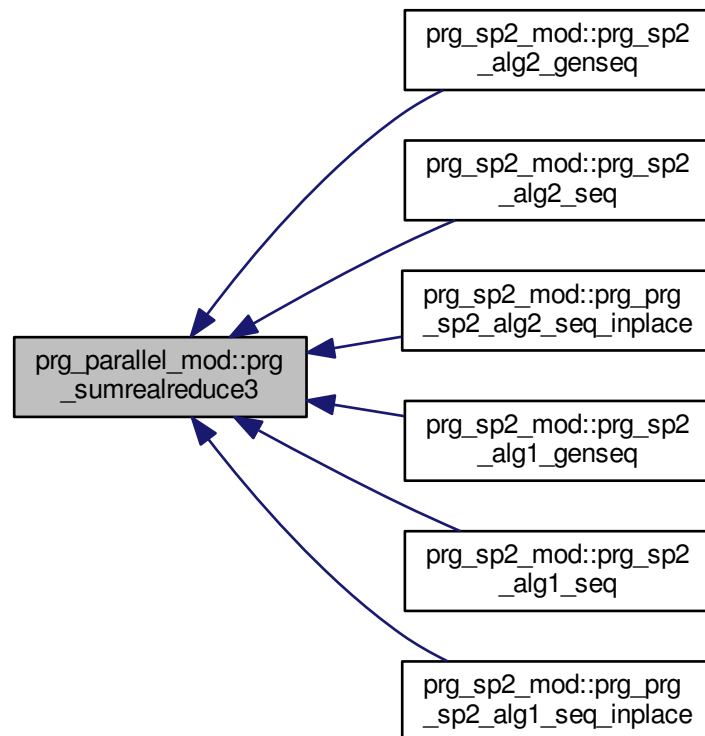
9.16.2.30 subroutine, public prg_parallel_mod::prg_sumrealreduce3 (real(dp), intent(inout) *value1*, real(dp), intent(inout) *value2*, real(dp), intent(inout) *value3*)

Definition at line 523 of file prg_parallel_mod.F90.

Here is the call graph for this function:



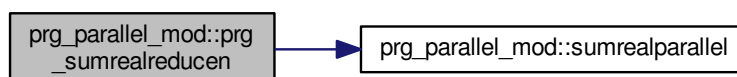
Here is the caller graph for this function:



9.16.2.31 subroutine, public `prg_parallel_mod::prg_sumrealreducen` (`real(dp)`, `dimension(n)`, `intent(inout) valueVec`, `integer`, `intent(in) N`)

Definition at line 543 of file `prg_parallel_mod.F90`.

Here is the call graph for this function:



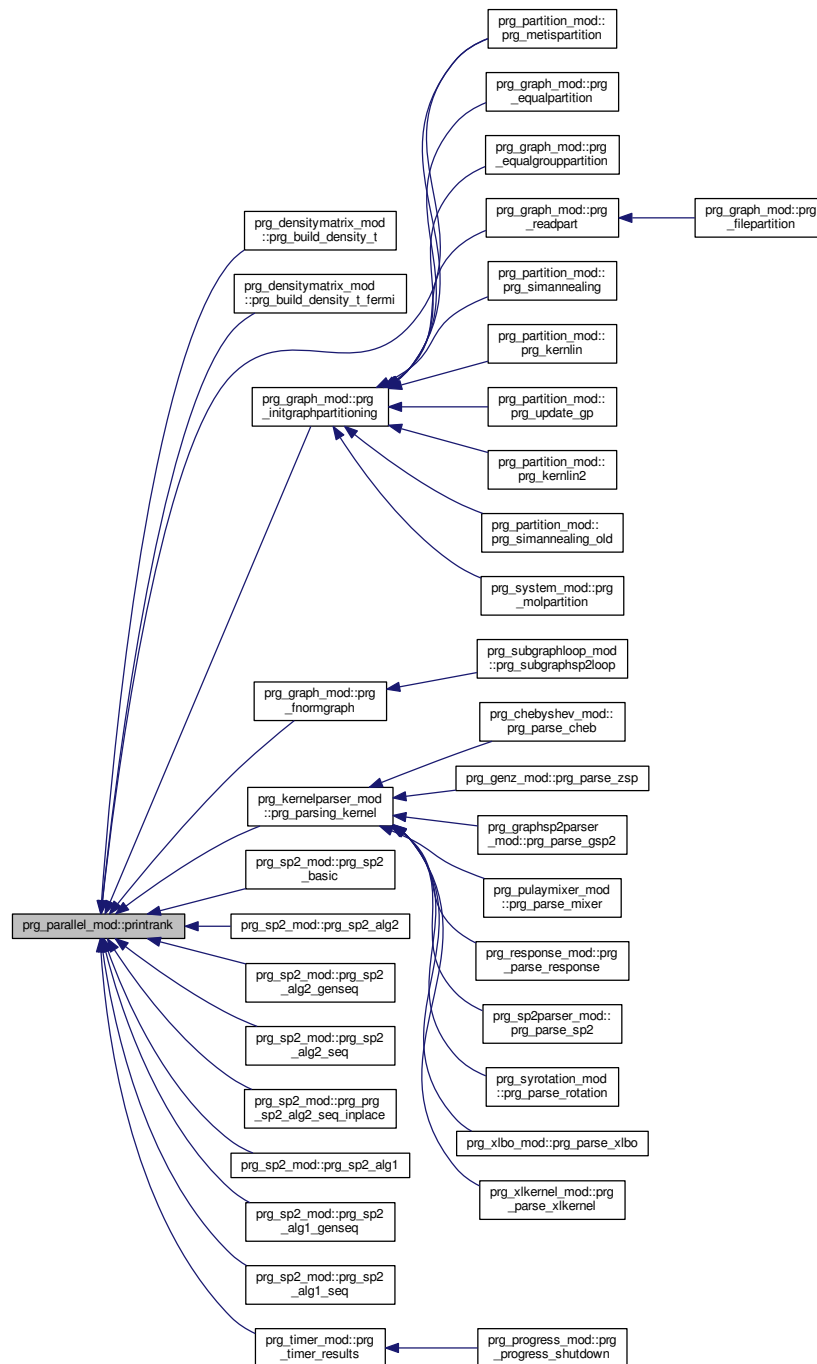
9.16.2.32 subroutine, public `prg_parallel_mod::prg_wait` ()

Definition at line 758 of file `prg_parallel_mod.F90`.

9.16.2.33 integer function, public prg_parallel_mod::printrank ()

Definition at line 111 of file prg_parallel_mod.F90.

Here is the caller graph for this function:



9.16.2.34 integer function prg_parallel_mod::saverequest (integer, intent(in) irequest) [private]

Definition at line 170 of file prg_parallel_mod.F90.

Here is the caller graph for this function:



9.16.2.35 subroutine, public `prg_parallel_mod::sendparallel (real(dp), dimension(*), intent(in) sendBuf, integer, intent(in) sendLen, integer, intent(in) dest)`

Definition at line 246 of file `prg_parallel_mod.F90`.

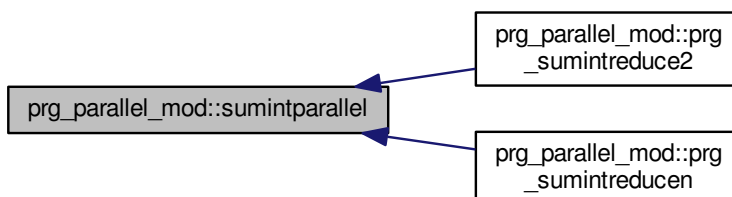
9.16.2.36 subroutine, public `prg_parallel_mod::sendreceiveparallel (real(dp), dimension(*), intent(in) sendBuf, integer, intent(in) sendLen, integer, intent(in) dest, real(dp), dimension(*), intent(out) recvBuf, integer, intent(in) recvLen, integer, intent(in) source, integer, intent(out) nreceived)`

Definition at line 207 of file `prg_parallel_mod.F90`.

9.16.2.37 subroutine, public `prg_parallel_mod::sumintparallel (integer, dimension(*), intent(in) sendBuf, integer, dimension(*) recvBuf, integer, intent(in) icount)`

Definition at line 295 of file `prg_parallel_mod.F90`.

Here is the caller graph for this function:



9.16.3.5 integer prg_parallel_mod::reqcount [private]

Definition at line 29 of file prg_parallel_mod.F90.

9.16.3.6 integer, dimension(:), allocatable prg_parallel_mod::requestlist [private]

Definition at line 30 of file prg_parallel_mod.F90.

9.16.3.7 integer, dimension(:), allocatable prg_parallel_mod::rused [private]

Definition at line 30 of file prg_parallel_mod.F90.

9.17 prg_partition_mod Module Reference

The partition module.

Functions/Subroutines

- subroutine, public [prg_metispartition](#) (gp, ngroups, nnodes, xadj, adjncy, nparts, part, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)
Create graph partitions minizing number of cut edges.
- subroutine, public [prg_costpartition](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)
Compute cost of a partition.
- subroutine, public [update_prg_costpartition](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, node, new_part)
Update cost of partition and the different parameters node is moves into new_part For each neighbor of node, the following cases hold: Case 1: neighbor is in old_part Case 2: neighbor is in new_part Case 3: neighbor is neither in old_ or new_part.
- subroutine [prg_accept_prob](#) (it, prg_delta, r)
Compute acceptance probability for simulated annealing.
- subroutine [prg_costindex](#) (cost, sumCubes, maxCH, smooth_maxCH, obj_fun)
Choose objective function to work with.
- subroutine [prg_rand_node](#) (gp, node, seed)
Pick a random node.
- subroutine, public [prg_simannealing](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, niter, seed)
Graph partitioning based on Simulated Annealing.
- subroutine, public [prg_kernlin](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, nconverg, seed)
Graph partitioning based on inspired by Kernighan-Lin Review METiS manual for description of k-way implementation of KL Pick a core together with its halos Place free vertices on a priority queue with (key, value) =(prg_delta, best_part), with prg_delta = change in obj_value Dequeue and allow hill climbing.
- subroutine, public [prg_update_gp](#) (gp, partNumber, core_count)
- subroutine [prg_rand_shuffle](#) (array, seed)
Randomly shuffle array.
- subroutine, public [prg_check_arrays](#) (gp, core_count, CH_count, Halo_count)

Error checking Checking that core_count, CH_count, Halo_count match.

- subroutine, public [prg_kernlin_queue](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)

Greedy algorithm. At each step it chooses the (vertex, new_part) pair with highest gain Currently implementation is very slow.

- subroutine [prg_find_best_move](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, best_node, best_part)

For kernlin_queue to find (vertex, new_part) pair with highest gain.

- subroutine, public [prg_kernlin2](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)
- subroutine [prg_get_largest_hedge_in_part](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, search_part, largest_Hedge)
- subroutine, public [prg_simannealing_old](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, niter, seed)

Variables

- integer, parameter [dp](#) = kind(1.0d0)
- integer, parameter [metis_index_kind](#) = METIS_INDEX_KIND
From /usr/include/metis.h.
- integer, parameter [metis_real_kind](#) = kind(METIS_REAL_KIND)
From /usr/include/metis.h.

9.17.1 Detailed Description

The partition module.

Contains different partitioning algorithms such as Metis, Simulated Annealing etc. Also contains optimization routines to improve upon existing partitioning, such as simulated annealing, etc.

9.17.2 Function/Subroutine Documentation

- 9.17.2.1 subroutine [prg_partition_mod::prg_accept_prob](#) (integer, intent(in) *it*, real(dp), intent(in) *prg_delta*, real, intent(inout) *r*) [private]

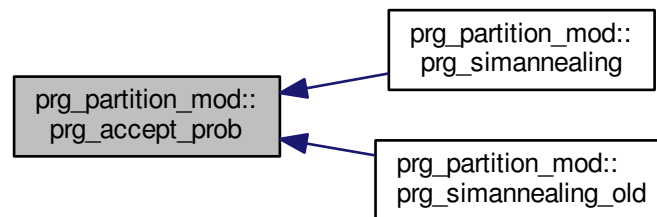
Compute acceptance probability for simulated annealing.

Parameters

<i>it</i>	iteration
<i>prg_delta</i>	(new_obj_value - old_obj_value)
<i>r</i>	acceptance probability

Definition at line 489 of file prg_partition_mod.F90.

Here is the caller graph for this function:



9.17.2.2 subroutine, public prg_partition_mod::prg_check_arrays (type (graph_partitioning_t), intent(inout) *gp*, integer, dimension(:), intent(inout), allocatable *core_count*, integer, dimension(:), intent(inout), allocatable *CH_count*, integer, dimension(:,:), intent(inout), allocatable *Halo_count*)

Error checking Checking that core_count, CH_count, Halo_count match.

Definition at line 1146 of file prg_partition_mod.F90.

9.17.2.3 subroutine prg_partition_mod::prg_costindex (real(dp), intent(inout) *cost*, real(dp), intent(inout) *sumCubes*, real(dp), intent(inout) *maxCH*, real(dp), intent(inout) *smooth_maxCH*, integer, intent(inout) *obj_fun*) [private]

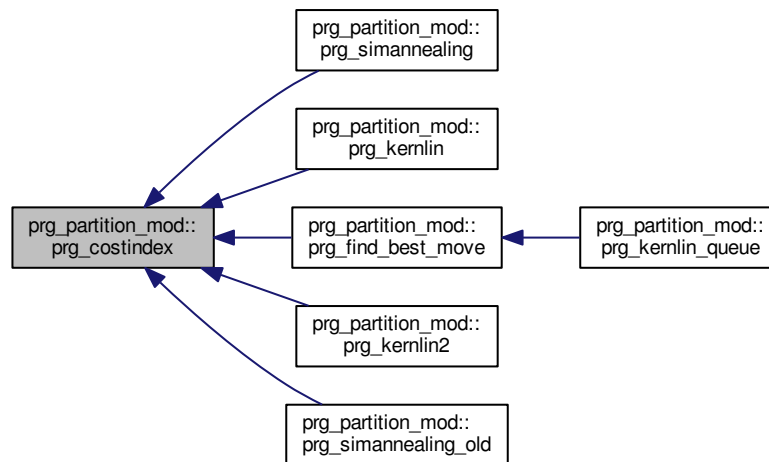
Choose objective function to work with.

Parameters

<i>cost</i>	output according to chosen obj_fun
<i>sumCubes</i>	Sum of cubes obj value
<i>maxCH</i>	maximum core-halo part size ojective value
<i>obj_fun</i>	0=sumcubes, 1=maxCH

Definition at line 507 of file prg_partition_mod.F90.

Here is the caller graph for this function:



9.17.2.4 subroutine, public prg_partition_mod::prg_costpartition (type (graph_partitioning_t), intent(inout) *gp*, integer, dimension(:), intent(inout), allocatable *xadj*, integer, dimension(:), intent(inout), allocatable *adjncy*, integer, dimension(:), intent(in), allocatable *partNumber*, integer, dimension(:), intent(inout), allocatable *core_count*, integer, dimension(:), intent(inout), allocatable *CH_count*, integer, dimension(:, :), intent(inout), allocatable *Halo_count*, real(dp), intent(inout) *sumCubes*, real(dp), intent(inout) *maxCH*, real(dp), intent(inout) *smooth_maxCH*, real(dp), intent(inout) *pnorm*)

Compute cost of a partition.

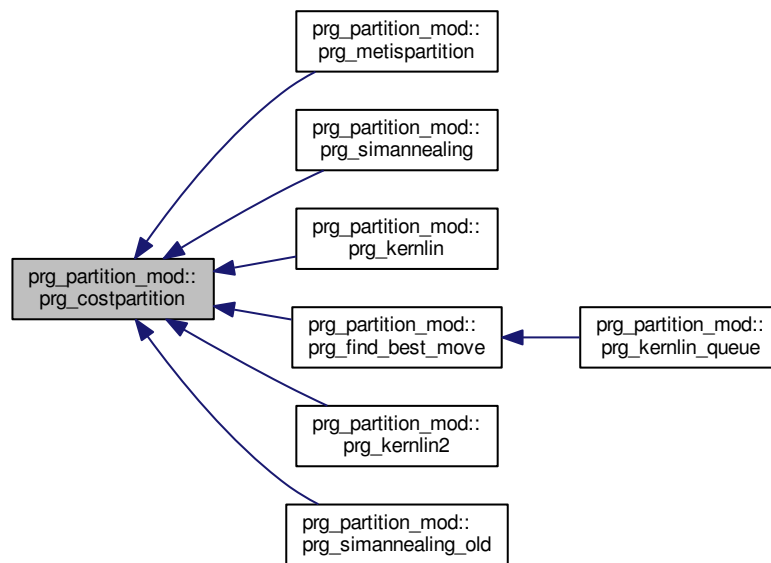
Parameters

<i>gp</i>	Graph partitioning
<i>xadj</i>	CSR array of graph nodes
<i>adjncy</i>	CSR array of graph neighbors
<i>nparts</i>	Number of Parts
<i>partNumber</i>	Partition vector
<i>core_count</i>	Array: number of core vertices in each part
<i>CH_count</i>	Array: number of core+halo vertices in each part
<i>Halo_count</i>	2D Array of size nparts by totalNodes: Halo_count(i,j) = k, node j is a halo of part i with k connections
<i>sumCubes</i>	Sum of cubes objective value
<i>maxCh</i>	maximum core-halo part size objective value

prg_initialize

Definition at line 327 of file prg_partition_mod.F90.

Here is the caller graph for this function:

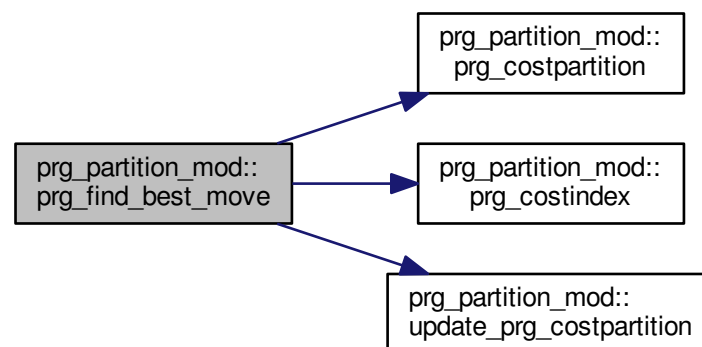


9.17.2.5 subroutine `prg_partition_mod::prg_find_best_move` (type(`graph_partitioning_t`), intent(inout) *gp*, integer, dimension(:), intent(inout), allocatable *xadj*, integer, dimension(:), intent(inout), allocatable *adjncy*, integer, dimension(:), intent(inout), allocatable *partNumber*, integer, dimension(:), intent(inout), allocatable *core_count*, integer, dimension(:), intent(inout), allocatable *CH_count*, integer, dimension(:,:), intent(inout), allocatable *Halo_count*, real(dp), intent(inout) *sumCubes*, real(dp), intent(inout) *maxCH*, real(dp), intent(inout) *smooth_maxCH*, real(dp), intent(inout) *pnorm*, integer, intent(inout) *best_node*, integer, intent(inout) *best_part*) [private]

For `kerlin_queue` to find (vertex, new_part) pair with highest gain.

Definition at line 1209 of file `prg_partition_mod.F90`.

Here is the call graph for this function:



Here is the caller graph for this function:

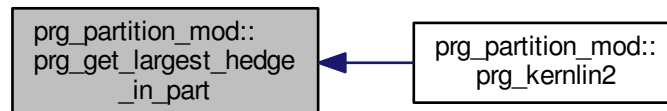


9.17.2.6 subroutine `prg_partition_mod::prg_get_largest_hedge_in_part` (type (`graph_partitioning_t`), intent(inout) `gp`, integer, dimension(:), intent(inout), allocatable `xadj`, integer, dimension(:), intent(inout), allocatable `adjncy`, integer, dimension(:), intent(inout), allocatable `partNumber`, integer, dimension(:), intent(inout), allocatable `core_count`, integer, dimension(:), intent(inout), allocatable `CH_count`, integer, dimension(:, :), intent(inout), allocatable `Halo_count`, real(`dp`), intent(inout) `sumCubes`, real(`dp`), intent(inout) `maxCH`, real(`dp`), intent(inout) `smooth_maxCH`, real(`dp`), intent(inout) `pnorm`, integer, intent(inout) `search_part`, integer, intent(inout) `largest_Hedge`) [private]

i can be viewed as a hyperedge for all hyperedges in `search_part`, pick the one with largest size

Definition at line 1420 of file `prg_partition_mod.F90`.

Here is the caller graph for this function:



9.17.2.7 subroutine, public `prg_partition_mod::prg_kernlin` (type (`graph_partitioning_t`), intent(inout) `gp`, integer, dimension(:), intent(inout), allocatable `xadj`, integer, dimension(:), intent(inout), allocatable `adjncy`, integer, dimension(:), intent(inout), allocatable `partNumber`, integer, dimension(:), intent(inout), allocatable `core_count`, integer, dimension(:), intent(inout), allocatable `CH_count`, integer, dimension(:, :), intent(inout), allocatable `Halo_count`, real(`dp`), intent(inout) `sumCubes`, real(`dp`), intent(inout) `maxCH`, real(`dp`), intent(inout) `smooth_maxCH`, real(`dp`), intent(inout) `pnorm`, integer, intent(in) `nconverg`, integer, intent(inout) `seed`)

Graph partitioning based on inspired by Kernighan-Lin Review METIS manual for description of k-way implementation of KL Pick a core together with its halos Place free vertices on a priority queue with (key, value) =(prg_delta, best_part), with prg_delta = change in obj_value Dequeue and allow hill climbing.

Parameters

<code>gp</code>	Graph partitioning
-----------------	--------------------

Parameters

<i>xadj</i>	CSR array of graph nodes
<i>adjncy</i>	CSR array of graph neighbors
<i>nparts</i>	Number of Parts
<i>partNumber</i>	Partition vector
<i>core_count</i>	Array: number of core vertices in each part
<i>CH_count</i>	Array: number of core+halo vertices in each part
<i>Halo_count</i>	2D Array of size nparts by totalNodes: $Halo_count(i,j) = k$, node j is a halo of part i with k connections
<i>sumCubes</i>	Sum of cubes objective value
<i>maxCh</i>	maximum core-halo part size objective value
<i>nconverg</i>	number of before convergence
<i>seed</i>	random number generator seed

Allocate arrays

Initialize variables

Initialize array of nodes

Randomize nodes

Compute current cost of partition

Choose objective function to minimize

iterate over the columns of the matrix, ie the hyperedges

KL iteration

let min_part be the smallest CH_part

Try and move free nodes to min_part

lock vertices (climb_counter) vertices have been accepted need to lock (climb_counter) vertices Last vertex to be moved is node_backup(climb_counter)

reset

If all vertices locked, go to next iteration

If empty parts exist, place a vertex in max_part there

Place j and it's neighbors that are in the max part into the empty part

Check Convergence

Check empty part exist move nodes from maxpart to empty part

move it neighbor in the max parts to the newpart

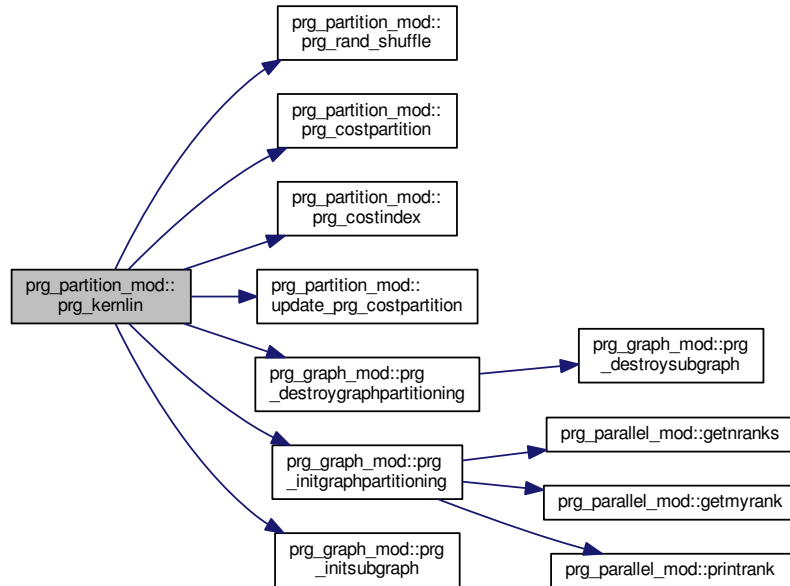
Update graph structure

Allocate subgraph structure

Assign node ids to sgraph

Definition at line 758 of file prg_partition_mod.F90.

Here is the call graph for this function:



9.17.2.8 subroutine, public prg_partition_mod::prg_kernlin2 (type (graph_partitioning_t), intent(inout) gp, integer, dimension(:), intent(inout), allocatable xadj, integer, dimension(:), intent(inout), allocatable adjncy, integer, dimension(:), intent(inout), allocatable partNumber, integer, dimension(:), intent(inout), allocatable core_count, integer, dimension(:), intent(inout), allocatable CH_count, integer, dimension(:,:), intent(inout), allocatable Halo_count, real(dp), intent(inout) sumCubes, real(dp), intent(inout) maxCH, real(dp), intent(inout) smooth_maxCH, real(dp), intent(inout) pnorm)

Allocate arrays

Pick hyperedge with largest size or random hyperedge with probability 0.5 We will change it to pick hyperedge with highest priority, where priority will be defined according to different metrics

Find part with smallest size (should be included in update_prg_costPartition)

if current part is max, move to min_part then move subsets (neighbors)

Move hyperedges to minCH part

Try and move intersecting hyperedges

Move k number of vertices. k should be small i.e k <=20, k set in prg_Kernlin_queue Only use this for small systems

Check empty part exist move nodes from maxpart to empty part

move it neighbor in the max parts to the newpart

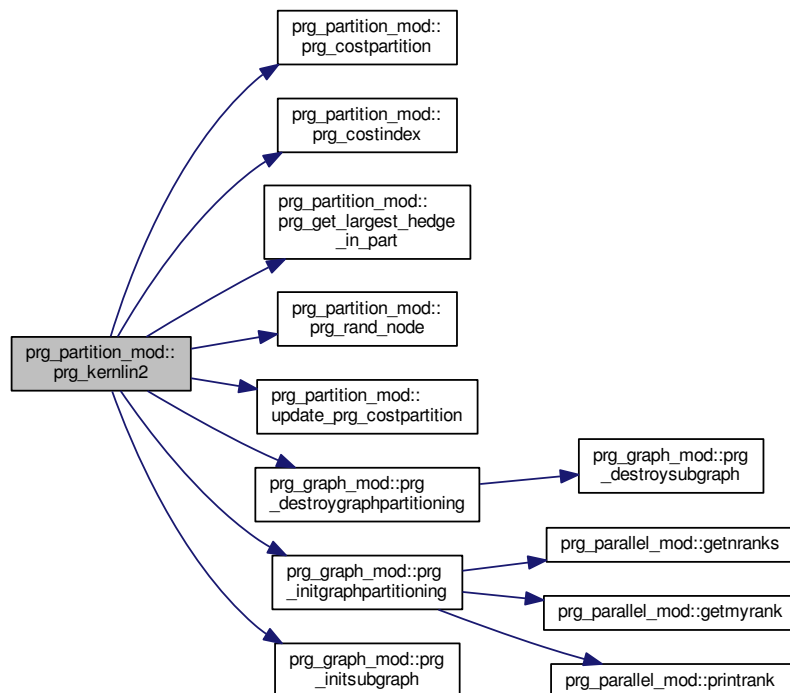
Update graph structure

Allocate subgraph structure

Assign node ids to sgraph

Definition at line 1257 of file prg_partition_mod.F90.

Here is the call graph for this function:

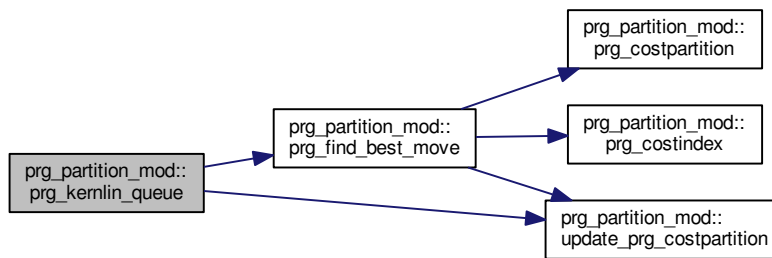


9.17.2.9 subroutine, public `prg_partition_mod::prg_kernlin_queue` (type (`graph_partitioning_t`), intent(inout) `gp`, integer, dimension(:), intent(inout), allocatable `xadj`, integer, dimension(:), intent(inout), allocatable `adjncy`, integer, dimension(:), intent(inout), allocatable `partNumber`, integer, dimension(:), intent(inout), allocatable `core_count`, integer, dimension(:), intent(inout), allocatable `CH_count`, integer, dimension(:, :), intent(inout), allocatable `Halo_count`, real(dp), intent(inout) `sumCubes`, real(dp), intent(inout) `maxCH`, real(dp), intent(inout) `smooth_maxCH`, real(dp), intent(inout) `pnorm`)

Greedy algorithm. At each step it chooses the (vertex, new_part) pair with highest gain Currently implementation is very slow.

Definition at line 1173 of file prg_partition_mod.F90.

Here is the call graph for this function:



9.17.2.10 subroutine, public `prg_partition_mod::prg_metispartition` (type (`graph_partitioning_t`), intent(inout) `gp`, integer, intent(in) `ngroups`, integer, intent(in) `nnodes`, integer, dimension(:), intent(inout), allocatable `xadj`, integer, dimension(:), intent(inout), allocatable `adjncy`, integer, intent(inout) `nparts`, integer, dimension(:), intent(inout), allocatable `part`, integer, dimension(:), intent(inout), allocatable `core_count`, integer, dimension(:), intent(inout), allocatable `CH_count`, integer, dimension(:, :), intent(inout), allocatable `Halo_count`, real(dp), intent(inout) `sumCubes`, real(dp), intent(inout) `maxCH`, real(dp), intent(inout) `smooth_maxCH`, real(dp), intent(inout) `pnorm`)

Create graph partitions minizing number of cut edges.

Parameters

<i>gp</i>	Graph partitioning'
<i>ngroups</i>	Number of groups/nodes
<i>nnodes</i>	Number of nodes
<i>xadj</i>	CSR array of graph nodes
<i>adjncy</i>	CSR array of graph neighbors
<i>nparts</i>	Number of Parts
<i>part</i>	Partition vector
<i>core_count</i>	Array: number of core vertices in each part
<i>CH_count</i>	Array: number of core+halo vertices in each part
<i>Halo_count</i>	2D Array of size nparts by totalNodes: <code>Halo_count(i,j) = k</code> , node <code>j</code> is a halo of part <code>i</code> with <code>k</code> connections
<i>sumCubes</i>	Sum of cubes objective value
<i>maxCh</i>	maximum core-halo part size obective value

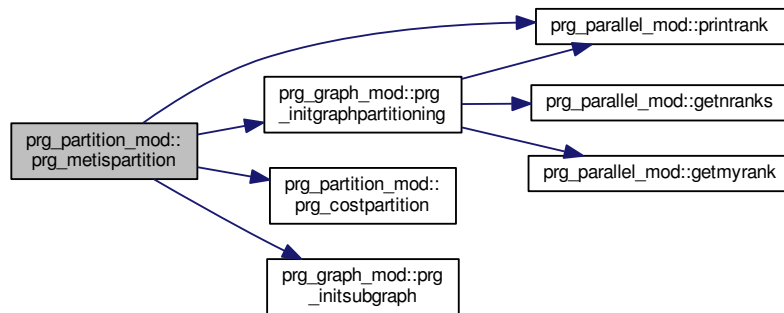
`prg_initialize`

Partition graph into nparts'

Compute cost of partition

Definition at line 217 of file `prg_partition_mod.F90`.

Here is the call graph for this function:



9.17.2.11 subroutine `prg_partition_mod::prg_rand_node` (type (graph_partitioning_t), intent(inout) *gp*, integer, intent(inout) *node*, integer, intent(inout) *seed*) [private]

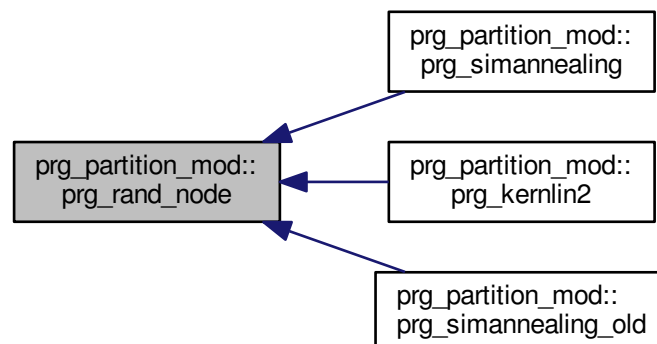
Pick a random node.

Parameters

<i>gp</i>	graph partitioning structure
<i>node</i>	output node
<i>seed</i>	random seed

Definition at line 527 of file `prg_partition_mod.F90`.

Here is the caller graph for this function:



9.17.2.12 subroutine `prg_partition_mod::prg_rand_shuffle` (integer, dimension(:), intent(inout) *array*, integer, intent(inout) *seed*) [private]

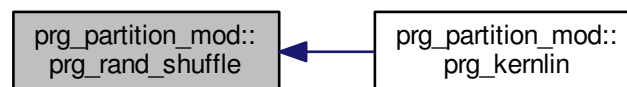
Randomly shuffle array.

Random seed

Shuffle array

Definition at line 1123 of file `prg_partition_mod.F90`.

Here is the caller graph for this function:



9.17.2.13 subroutine, public `prg_partition_mod::prg_simannealing` (type (graph_partitioning_t), intent(inout) *gp*, integer, dimension(:), intent(inout), allocatable *xadj*, integer, dimension(:), intent(inout), allocatable *adjncy*, integer, dimension(:), intent(inout), allocatable *partNumber*, integer, dimension(:), intent(inout), allocatable *core_count*, integer, dimension(:), intent(inout), allocatable *CH_count*, integer, dimension(:, :), intent(inout), allocatable *Halo_count*, real(dp), intent(inout) *sumCubes*, real(dp), intent(inout) *maxCH*, real(dp), intent(inout) *smooth_maxCH*, real(dp), intent(inout) *pnorm*, integer, intent(in) *niter*, integer, intent(inout) *seed*)

Graph partitioning based on Simulated Annealing.

Parameters

<i>gp</i>	Graph partitioning
<i>xadj</i>	CSR array of graph nodes
<i>adjncy</i>	CSR array of graph neighbors
<i>nparts</i>	Number of Parts
<i>partNumber</i>	Partition vector
<i>core_count</i>	Array: number of core vertices in each part
<i>CH_count</i>	Array: number of core+halo vertices in each part
<i>Halo_count</i>	2D Array of size nparts by totalNodes: $Halo_count(i,j) = k$, node <i>j</i> is a halo of part <i>i</i> with <i>k</i> connections
<i>sumCubes</i>	Sum of cubes objective value
<i>maxCh</i>	maximum core-halo part size objective value
<i>niter</i>	Number of iterations
<i>seed</i>	Random seed

Compute current cost of partition

Choose objective function to minimize

Perform SA

Find part with smallest size (should be included in update_prg_costPartition

if part(node) == max_ch_part, try to move node and its neighbors to min_ch_part else move neighbors to part(node)

Check empty part exist move nodes from maxpart to empty part

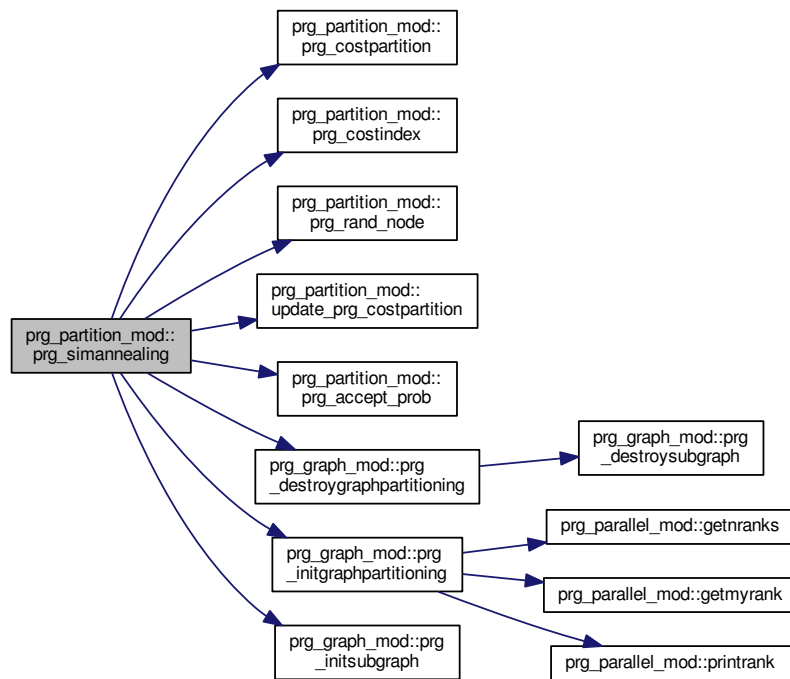
move it neighbor in the max parts to the newpart

Update graph structure

For debugging

Definition at line 552 of file prg_partition_mod.F90.

Here is the call graph for this function:



9.17.2.14 subroutine, public `prg_partition_mod::prg_simannealing_old` (type (graph_partitioning_t), intent(inout) *gp*, integer, dimension(:), intent(inout), allocatable *xadj*, integer, dimension(:), intent(inout), allocatable *adjncy*, integer, dimension(:), intent(inout), allocatable *partNumber*, integer, dimension(:), intent(inout), allocatable *core_count*, integer, dimension(:), intent(inout), allocatable *CH_count*, integer, dimension(:, :), intent(inout), allocatable *Halo_count*, real(dp), intent(inout) *sumCubes*, real(dp), intent(inout) *maxCH*, real(dp), intent(inout) *smooth_maxCH*, real(dp), intent(inout) *pnorm*, integer, intent(in) *niter*, integer, intent(inout) *seed*)

Compute current cost of partition

Choose objective function to minimize

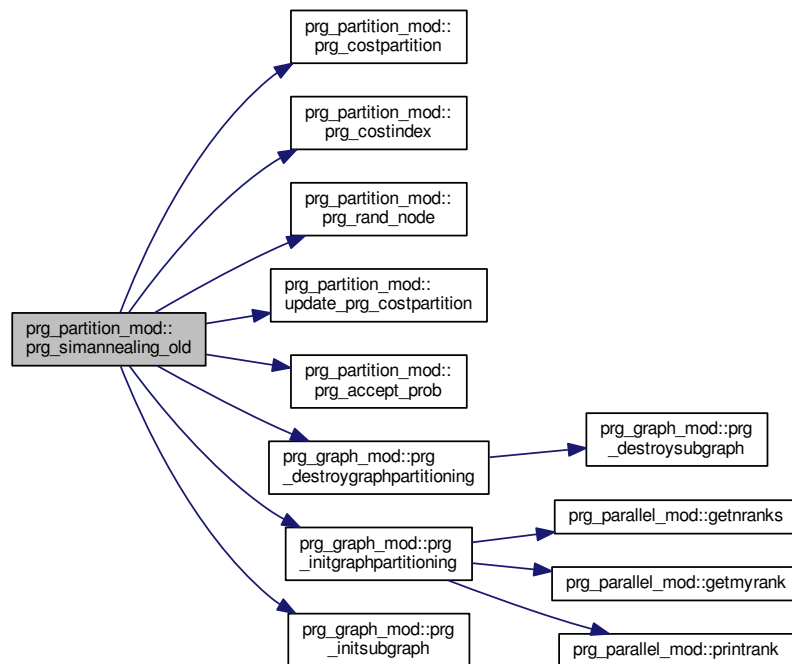
Perform SA

Update graph structure

For debugging

Definition at line 1453 of file prg_partition_mod.F90.

Here is the call graph for this function:



9.17.2.15 subroutine, public `prg_partition_mod::prg_update_gp (type (graph_partitioning_t), intent(inout) gp, integer, dimension(:), intent(inout), allocatable partNumber, integer, dimension(:), intent(inout), allocatable core_count)`

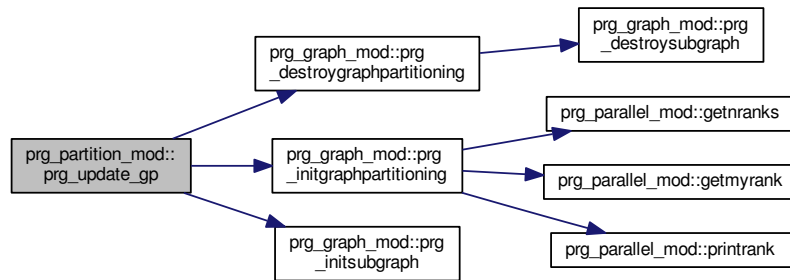
Update graph structure

Allocate subgraph structure

Assign node ids to sgraph

Definition at line 1082 of file prg_partition_mod.F90.

Here is the call graph for this function:



9.17.2.16 subroutine, public prg_partition_mod::update_prg_costpartition (type (graph_partitioning_t), intent(inout) *gp*, integer, dimension(:), intent(inout), allocatable *xadj*, integer, dimension(:), intent(inout), allocatable *adjncy*, integer, dimension(:), intent(inout), allocatable *partNumber*, integer, dimension(:), intent(inout), allocatable *core_count*, integer, dimension(:), intent(inout), allocatable *CH_count*, integer, dimension(:,,:), intent(inout), allocatable *Halo_count*, real(dp), intent(inout) *sumCubes*, real(dp), intent(inout) *maxCH*, real(dp), intent(inout) *smooth_maxCH*, real(dp), intent(inout) *pnorm*, integer, intent(in) *node*, integer, intent(in) *new_part*)

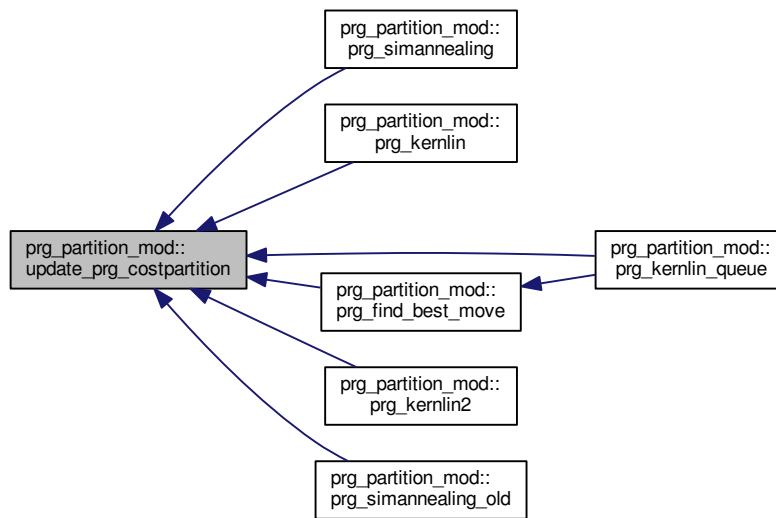
Update cost of partition and the different parameters node is moves into new_part For each neighbor of node, the following cases hold: Case 1: neighbor is in old_part Case 2: neighbor is in new_part Case 3: neighbor is neither in old_ or new_part.

Parameters

<i>gp</i>	Graph partitioning
<i>xadj</i>	CSR array of graph nodes
<i>adjncy</i>	CSR array of 1043365660.0000000graph neighbors
<i>nparts</i>	Number of Parts
<i>partNumber</i>	Partition vector
<i>core_count</i>	Array: number of core vertices in each part
<i>CH_count</i>	Array: number of core+halo vertices in each part
<i>Halo_count</i>	2D Array of size nparts by totalNodes: Halo_count(i,j) = k, node j is a halo of part i with k connections
<i>sumCubes</i>	Sum of cubes objective value
<i>maxCh</i>	maximum core-halo part size oboective value
<i>node</i>	Vertex that has moved to new_part
<i>new_part</i>	new part that node has moved to

Definition at line 401 of file prg_partition_mod.F90.

Here is the caller graph for this function:



9.17.3 Variable Documentation

9.17.3.1 integer, parameter `prg_partition_mod::dp = kind(1.0d0)` `[private]`

Definition at line 18 of file `prg_partition_mod.F90`.

9.17.3.2 integer, parameter `prg_partition_mod::metis_index_kind = METIS_INDEX_KIND` `[private]`

From `/usr/include/metis.h`.

`IDXTYPEWIDTH = 32 -> metis_index_kind = 4` `IDXTYPEWIDTH = 64 -> metis_index_kind = 8`

Definition at line 24 of file `prg_partition_mod.F90`.

9.17.3.3 integer, parameter `prg_partition_mod::metis_real_kind = kind(METIS_REAL_KIND)` `[private]`

From `/usr/include/metis.h`.

`REALTYPEWIDTH = 32 -> metis_real_kind = kind(0e0)` `REALTYPEWIDTH = 64 -> metis_real_kind = kind(0d0)`

Definition at line 30 of file `prg_partition_mod.F90`.

9.18 prg_progress_mod Module Reference

The progress module.

Functions/Subroutines

- subroutine, public [prg_progress_init](#) ()
Initialize progress.
- subroutine, public [prg_progress_shutdown](#) ()
Shutdown progress.

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.18.1 Detailed Description

The progress module.

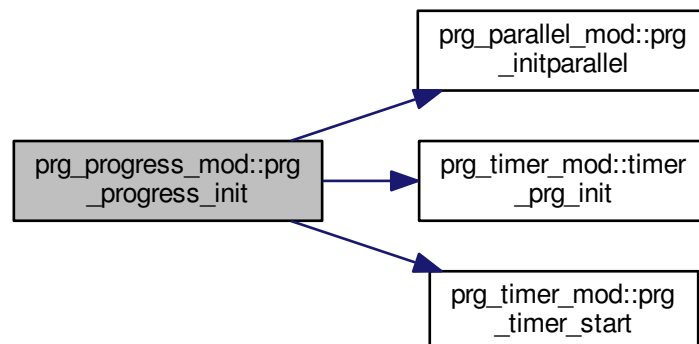
9.18.2 Function/Subroutine Documentation

9.18.2.1 subroutine, public `prg_progress_mod::prg_progress_init` ()

Initialize progress.

Definition at line 25 of file `prg_progress_mod.F90`.

Here is the call graph for this function:

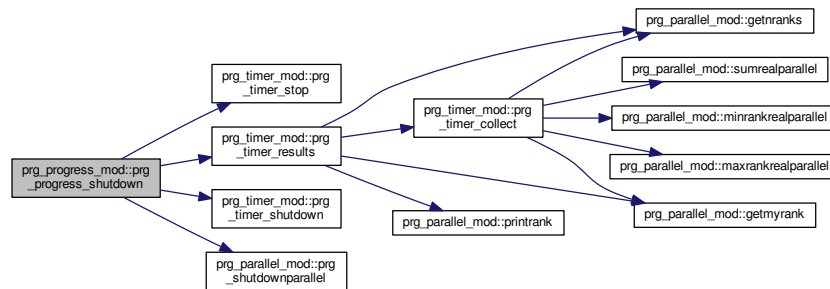


9.18.2.2 subroutine, public prg_progress_mod::prg_progress_shutdown ()

Shutdown progress.

Definition at line 37 of file prg_progress_mod.F90.

Here is the call graph for this function:



9.18.3 Variable Documentation

9.18.3.1 integer, parameter prg_progress_mod::dp = kind(1.0d0) [private]

Definition at line 16 of file prg_progress_mod.F90.

9.19 prg_ptable_mod Module Reference

Periodic table of elements.

Functions/Subroutines

- integer function, public [element_atomic_number](#) (symbol)
- integer function [element_atomic_number_upper](#) (symbol)

Variables

- integer, parameter [nz](#) = 103
- integer, parameter, private [dp](#) = kind(1.0d0)
- character(2), dimension([nz](#)), parameter [element_symbol](#) = [character(2) :: "H", "He", "Li", "Be", "B", "C", "N", "O", "F", "Ne", "Na", "Mg", "Al", "Si", "P", "S", "Cl", "Ar", "K", "Ca", "Sc", "Ti", "V", "Cr", "Mn", "Fe", "Co", "Ni", "Cu", "Zn", "Ga", "Ge", "As", "Se", "Br", "Kr", "Rb", "Sr", "Y", "Zr", "Nb", "Mo", "Tc", "Ru", "Rh", "Pd", "Ag", "Cd", "In", "Sn", "Sb", "Te", "I", "Xe", "Cs", "Ba", "La", "Ce", "Pr", "Nd", "Pm", "Sm", "Eu", "Gd", "Tb", "Dy", "Ho", "Er", "Tm", "Yb", "Lu", "Hf", "Ta", "W", "Re", "Os", "Ir", "Pt", "Au", "Hg", "Tl", "Pb", "Bi", "Po", "At", "Rn", "Fr", "Ra", "Ac", "Th", "Pa", "U", "Np", "Pu", "Am", "Cm", "Bk", "Cf", "Es", "Fm", "Md", "No", "Lr"]

Element symbol.

- `character(2)`, `dimension(nz)`, parameter `element_symbol_upper` = [`character(2)` :: "H", "HE", "LI", "BE", "B", "C", "N", "O", "F", "NE", "NA", "MG", "AL", "SI", "P", "S", "CL", "AR", "K", "CA", "SC", "TI", "V", "CR", "MN", "FE", "CO", "NI", "CU", "ZN", "GA", "GE", "AS", "SE", "BR", "KR", "RB", "SR", "Y", "ZR", "NB", "MO", "TC", "RU", "RH", "PD", "AG", "CD", "IN", "SN", "SB", "TE", "I", "XE", "CS", "BA", "LA", "CE", "PR", "ND", "PM", "SM", "EU", "GD", "TB", "DY", "HO", "ER", "TM", "YB", "LU", "HF", "TA", "W", "RE", "OS", "IR", "PT", "AU", "HG", "TL", "PB", "BI", "PO", "AT", "RN", "FR", "RA", "AC", "TH", "PA", "U", "NP", "PU", "AM", "CM", "BK", "CF", "ES", "FM", "MD", "NO", "LR"]

Element symbol upper.

- `character(20)`, `dimension(nz)`, parameter `element_name` = [`character(20)` :: "Hydrogen", "Helium", "Lithium", "Beryllium", "Boron", "Carbon", "Nitrogen", "Oxygen", "Fluorine", "Neon", "Sodium", "Magnesium", "Aluminium", "Silicon", "Phosphorus", "Sulfur", "Chlorine", "Argon", "Potassium", "Calcium", "Scandium", "Titanium", "Vanadium", "Chromium", "Manganese", "Iron", "Cobalt", "Nickel", "Copper", "Zinc", "Gallium", "Germanium", "Arsenic", "Selenium", "Bromine", "Krypton", "Rubidium", "Strontium", "Yttrium", "Zirconium", "Niobium", "Molybdenum", "Technetium", "Ruthenium", "Rhodium", "Palladium", "Silver", "Cadmium", "Indium", "Tin", "Antimony", "Tellurium", "Iodine", "Xenon", "Caesium", "Barium", "Lanthanum", "Cerium", "Praseodymium", "Neodymium", "Promethium", "Samarium", "Europium", "Gadolinium", "Terbium", "Dysprosium", "Holmium", "Erbium", "Thulium", "Ytterbium", "Lutetium", "Hafnium", "Tantalum", "Tungsten", "Rhenium", "Osmium", "Iridium", "Platinum", "Gold", "Mercury", "Thallium", "Lead", "Bismuth", "Polonium", "Astatine", "Radon", "Francium", "Radium", "Actinium", "Thorium", "Protactinium", "Uranium", "Neptunium", "Plutonium", "Americium", "Curium", "Berkelium", "Californium", "Einsteinium", "Fermium", "Mendelevium", "Nobelium", "Lawrencium"]

Element name.

- `real(dp)`, `dimension(nz)`, parameter `element_mass` = (/ 1.007825032, 4.002603254, 7.01600455, 9.0121822, 11.0093054, 12.0, 14.003074005, 15.99491462, 18.99840322, 19.992440175, 22.989769281, 23.↵ 9850417, 26.98153863, 27.976926532, 30.97376163, 31.972071, 34.96885268, 39.962383123, 38.↵ 96370668, 39.96259098, 44.9559119, 47.9479463, 50.9439595, 51.9405075, 54.9380451, 55.9349375, 58.933195, 57.9353429, 62.9295975, 63.929142, 68.925573, 73.921177, 74.921596, 79.916521, 78.↵ 918337, 83.911507, 84.911789, 87.905612, 88.905848, 89.904704, 92.906378, 97.905408, 97.907216, 101.904349, 102.905504, 105.903486, 106.905097, 113.903358, 114.903878, 119.902194, 120.↵ 903815, 129.906224, 126.904473, 131.904153, 132.905451, 137.905247, 138.906353, 139.905438, 140.907652, 141.907723, 144.912749, 151.919732, 152.92123, 157.924103, 158.925346, 163.929174, 164.930322, 165.930293, 168.934213, 173.938862, 174.940771, 179.94655, 180.947995, 183.950931, 186.955753, 191.96148, 192.962926, 194.964791, 196.966568, 201.970643, 204.974427, 207.976652, 208.980398, 208.98243, 209.987148, 222.017577, 223.019735, 226.025409, 227.027752, 232.038055, 231.035884, 238.050788, 237.048173, 244.064204, 243.061381, 247.070354, 247.070307, 251.079587, 252.08298, 257.095105, 258.098431, 259.10103, 262.10963 /)

Element mass in atomic mass units (1.66 x 10⁻²⁷ kg)

- `real(dp)`, `dimension(nz)`, parameter `element_vdwr` = (/ 1.1, 1.4, 1.81, 1.53, 1.92, 1.7, 1.55, 1.52, 1.47, 1.54, 2.27, 1.73, 1.84, 2.1, 1.8, 1.8, 1.75, 1.88, 2.75, 2.31, 2.3, 2.15, 2.05, 2.05, 2.05, 2.05, 2.0, 2.0, 2.0, 2.1, 1.87, 2.11, 1.85, 1.9, 1.83, 2.02, 3.03, 2.49, 2.4, 2.3, 2.15, 2.1, 2.05, 2.05, 2.0, 2.05, 2.1, 2.2, 2.2, 1.93, 2.17, 2.06, 1.98, 2.16, 3.43, 2.68, 2.5, 2.48, 2.47, 2.45, 2.43, 2.42, 2.4, 2.38, 2.37, 2.35, 2.33, 2.32, 2.3, 2.28, 2.27, 2.25, 2.2, 2.1, 2.05, 2.0, 2.0, 2.05, 2.1, 2.05, 1.96, 2.02, 2.07, 1.97, 2.02, 2.2, 3.48, 2.83, 2.0, 2.4, 2.0, 2.3, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0 /)

van der Waals radius (in Angstroms)

- `real(dp)`, `dimension(nz)`, parameter `element_covr` = (/ 0.31, 0.28, 1.28, 0.96, 0.84, 0.76, 0.71, 0.66, 0.57, 0.58, 1.66, 1.41, 1.21, 1.11, 1.07, 1.05, 1.02, 1.06, 2.03, 1.76, 1.7, 1.6, 1.53, 1.39, 1.39, 1.32, 1.26, 1.24, 1.32, 1.22, 1.22, 1.2, 1.19, 1.2, 1.2, 1.16, 2.2, 1.95, 1.9, 1.75, 1.64, 1.54, 1.47, 1.46, 1.42, 1.39, 1.45, 1.44, 1.42, 1.39, 1.39, 1.38, 1.39, 1.4, 2.44, 2.15, 2.07, 2.04, 2.03, 2.01, 1.99, 1.98, 1.98, 1.96, 1.94, 1.92, 1.92, 1.89, 1.9, 1.87, 1.87, 1.75, 1.7, 1.62, 1.51, 1.44, 1.41, 1.36, 1.36, 1.32, 1.45, 1.46, 1.48, 1.4, 1.5, 1.5, 2.6, 2.21, 2.15, 2.06, 2.0, 1.96, 1.9, 1.87, 1.8, 1.69, 1.6, 1.6, 1.6, 1.6, 1.6, 1.6, 1.6 /)

Covalent radius (in Angstroms)

- `real(dp)`, `dimension(nz)`, parameter `element_ip` = (/ 13.5984, 24.5874, 5.3917, 9.3227, 8.298, 11.2603, 14.5341, 13.6181, 17.4228, 21.5645, 5.1391, 7.6462, 5.9858, 8.1517, 10.4867, 10.36, 12.9676, 15.7596, 4.3407, 6.1132, 6.5615, 6.8281, 6.7462, 6.7665, 7.434, 7.9024, 7.881, 7.6398, 7.7264, 9.3942, 5.9993, 7.8994, 9.7886, 9.7524, 11.8138, 13.9996, 4.1771, 5.6949, 6.2173, 6.6339, 6.7589

, 7.0924 , 7.28 , 7.3605 , 7.4589 , 8.3369 , 7.5762 , 8.9938 , 5.7864 , 7.3439 , 8.6084 , 9.0096 , 10.4513 , 12.1298 , 3.8939 , 5.2117 , 5.5769 , 5.5387 , 5.473 , 5.525 , 5.582 , 5.6437 , 5.6704 , 6.1498 , 5.8638 , 5.9389 , 6.0215 , 6.1077 , 6.1843 , 6.2542 , 5.4259 , 6.8251 , 7.5496 , 7.864 , 7.8335 , 8.4382 , 8.967 , 8.9588 , 9.2255 , 10.4375 , 6.1082 , 7.4167 , 7.2855 , 8.414 , 0.0 , 10.7485 , 4.0727 , 5.2784 , 5.17 , 6.3067 , 5.89 , 6.1941 , 6.2657 , 6.026 , 5.9738 , 5.9914 , 6.1979 , 6.2817 , 6.42 , 6.5 , 6.58 , 6.65 , 4.9 /)

Ionization energy (in eV)

- real(dp), dimension(nz), parameter `element_ea` = (/ 0.75420375 , 0.0 , 0.618049 , 0.0 , 0.279723 , 1.262118 , -0.07 , 1.461112 , 3.4011887 , 0.0 , 0.547926 , 0.0 , 0.43283 , 1.389521 , 0.7465 , 2.0771029 , 3.612724 , 0.0 , 0.501459 , 0.02455 , 0.188 , 0.084 , 0.525 , 0.67584 , 0.0 , 0.151 , 0.6633 , 1.15716 , 1.23578 , 0.0 , 0.41 , 1.232712 , 0.814 , 2.02067 , 3.363588 , 0.0 , 0.485916 , 0.05206 , 0.307 , 0.426 , 0.893 , 0.7472 , 0.55 , 1.04638 , 1.14289 , 0.56214 , 1.30447 , 0.0 , 0.404 , 1.112066 , 1.047401 , 1.970875 , 3.059038 , 0.0 , 0.471626 , 0.14462 , 0.47 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.0 , 0.322 , 0.815 , 0.15 , 1.0778 , 1.56436 , 2.1251 , 2.30861 , 0.0 , 0.377 , 0.364 , 0.942363 , 1.9 , 2.8 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 /)

Electron affprg_inity (in eV)

- real(dp), dimension(nz), parameter `atom_en` = (/ 2.2 , 0.0 , 0.98 , 1.57 , 2.04 , 2.55 , 3.04 , 3.44 , 3.98 , 0.0 , 0.93 , 1.31 , 1.61 , 1.9 , 2.19 , 2.58 , 3.16 , 0.0 , 0.82 , 1.0 , 1.36 , 1.54 , 1.63 , 1.66 , 1.55 , 1.83 , 1.88 , 1.91 , 1.9 , 1.65 , 1.81 , 2.01 , 2.18 , 2.55 , 2.96 , 3.0 , 0.82 , 0.95 , 1.22 , 1.33 , 1.6 , 2.16 , 1.9 , 2.2 , 2.28 , 2.2 , 1.93 , 1.69 , 1.78 , 1.96 , 2.05 , 2.1 , 2.66 , 2.6 , 0.79 , 0.89 , 1.1 , 1.12 , 1.13 , 1.14 , 0.0 , 1.17 , 0.0 , 1.2 , 0.0 , 1.22 , 1.23 , 1.24 , 1.25 , 0.0 , 1.27 , 1.3 , 1.5 , 2.36 , 1.9 , 2.2 , 2.2 , 2.28 , 2.54 , 2.0 , 1.62 , 2.33 , 2.02 , 2.0 , 2.2 , 0.0 , 0.7 , 0.9 , 1.1 , 1.3 , 1.5 , 1.38 , 1.36 , 1.28 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 0.0 /)

The Pauling electronegativity for this element.

- integer, dimension(nz), parameter `element_maxbonds` = (/ 1 , 0 , 1 , 2 , 4 , 4 , 4 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 8 , 6 , 6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 12 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 /)

The maximum expected number of bonds to this element.

- integer, dimension(nz), parameter `element_numel` = (/ 1 , 2 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 /)

Last shell number of electrons.

- character(50), dimension(nz), parameter `element_econf` = [character(50) :: "1s" , "1s2" , "1s22s" , "1s22s2" , "1s22s22p" , "1s22s22p2" , "1s22s22p3" , "1s22s22p4" , "1s22s22p5" , "1s22s22p6" , "[Ne]3s" , "[Ne]3s2" , "[Ne]3s23p" , "[Ne]3s23p2" , "[Ne]3s23p3" , "[Ne]3s23p4" , "[Ne]3s23p5" , "[Ne]3s23p6" , "[Ar]4s" , "[Ar]4s2" , "[Ar]3d4s2" , "[Ar]3d24s2" , "[Ar]3d34s2" , "[Ar]3d54s" , "[Ar]3d54s2" , "[Ar]3d64s2" , "[Ar]3d74s2" , "[Ar]3d84s2" , "[Ar]3d104s" , "[Ar]3d104s2" , "[Ar]3d104s24p" , "[Ar]3d104s24p2" , "[Ar]3d104s24p3" , "[Ar]3d104s24p4" , "[Ar]3d104s24p5" , "[Ar]3d104s24p6" , "[Kr]5s" , "[Kr]5s2" , "[Kr]4d5s2" , "[Kr]4d25s2" , "[Kr]4d45s" , "[Kr]4d55s" , "[Kr]4d55s2" , "[Kr]4d75s" , "[Kr]4d85s" , "[Kr]4d10" , "[Kr]4d105s" , "[Kr]4d105s2" , "[Cd]5p" , "[Cd]5p2" , "[Cd]5p3" , "[Cd]5p4" , "[Cd]5p5" , "[Cd]5p6" , "[Xe]6s" , "[Xe]6s2" , "[Xe]5d6s2" , "[Xe]4f5d6s2" , "[Xe]4f36s2" , "[Xe]4f46s2" , "[Xe]4f56s2" , "[Xe]4f66s2" , "[Xe]4f76s2" , "[Xe]4f75d6s2" , "[Xe]4f96s2" , "[Xe]4f106s2" , "[Xe]4f116s2" , "[Xe]4f126s2" , "[Xe]4f136s2" , "[Xe]4f146s2" , "[Xe]4f145d6s2" , "[Xe]4f145d26s2" , "[Xe]4f145d36s2" , "[Xe]4f145d46s2" , "[Xe]4f145d56s2" , "[Xe]4f145d66s2" , "[Xe]4f145d76s2" , "[Xe]4f145d96s" , "[Xe]4f145d106s" , "[Xe]4f145d106s2" , "[Hg]6p" , "[Hg]6p2" , "[Hg]6p3" , "[Hg]6p4" , "[Hg]6p5" , "[Hg]6p6" , "[Rn]7s" , "[Rn]7s2" , "[Rn]6d7s2" , "[Rn]6d27s2" , "[Rn]5f26d7s2" , "[Rn]5f36d7s2" , "[Rn]5f46d7s2" , "[Rn]5f67s2" , "[Rn]5f77s2" , "[Rn]5f76d7s2" , "[Rn]5f97s2" , "[Rn]5f107s2" , "[Rn]5f117s2" , "[Rn]5f127s2" , "[Rn]5f137s2" , "[Rn]5f147s2" , "[Rn]5f147s27p"]

The electronic configuration.

9.19.1 Detailed Description

Periodic table of elements.

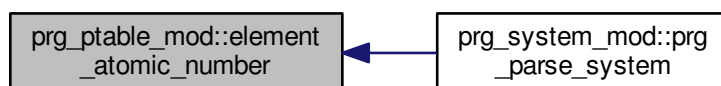
This data was prg_generated with pybabel and openbable packages Openbabel: <http://openbabel.org/dev-api/index.shtml> Pybel: https://openbabel.org/docs/dev/UseTheLibrary/Python_Pybel.html# Other sources includes NIST: http://www.nist.gov/pml/data/ion_energy.cfm

9.19.2 Function/Subroutine Documentation

9.19.2.1 integer function, public prg_ptable_mod::element_atomic_number (character(len=*) *symbol*)

Definition at line 394 of file prg_ptable_mod.F90.

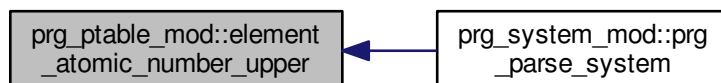
Here is the caller graph for this function:



9.19.2.2 integer function prg_ptable_mod::element_atomic_number_upper (character(len=*) *symbol*)

Definition at line 408 of file prg_ptable_mod.F90.

Here is the caller graph for this function:



9.19.3 Variable Documentation

9.19.3.1 `real(dp), dimension(nz), parameter prg_ptable_mod::atom_en = (/ 2.2 , 0.0 , 0.98 , 1.57 , 2.04 , 2.55 , 3.04 , 3.44 , 3.98 , 0.0 , 0.93 , 1.31 , 1.61 , 1.9 , 2.19 , 2.58 , 3.16 , 0.0 , 0.82 , 1.0 , 1.36 , 1.54 , 1.63 , 1.66 , 1.55 , 1.83 , 1.88 , 1.91 , 1.9 , 1.65 , 1.81 , 2.01 , 2.18 , 2.55 , 2.96 , 3.0 , 0.82 , 0.95 , 1.22 , 1.33 , 1.6 , 2.16 , 1.9 , 2.2 , 2.28 , 2.2 , 1.93 , 1.69 , 1.78 , 1.96 , 2.05 , 2.1 , 2.66 , 2.6 , 0.79 , 0.89 , 1.1 , 1.12 , 1.13 , 1.14 , 0.0 , 1.17 , 0.0 , 1.2 , 0.0 , 1.22 , 1.23 , 1.24 , 1.25 , 0.0 , 1.27 , 1.3 , 1.5 , 2.36 , 1.9 , 2.2 , 2.2 , 2.28 , 2.54 , 2.0 , 1.62 , 2.33 , 2.02 , 2.0 , 2.2 , 0.0 , 0.7 , 0.9 , 1.1 , 1.3 , 1.5 , 1.38 , 1.36 , 1.28 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 0.0 /)`

The Pauling electronegativity for this element.

Definition at line 266 of file prg_ptable_mod.F90.

9.19.3.2 integer, parameter, private prg_ptable_mod::dp = kind(1.0d0) [private]

Definition at line 13 of file prg_ptable_mod.F90.

9.19.3.3 real(dp), dimension(nz), parameter prg_ptable_mod::element_covr = (/ 0.31 , 0.28 , 1.28 , 0.96 , 0.84 , 0.76 , 0.71 , 0.66 , 0.57 , 0.58 , 1.66 , 1.41 , 1.21 , 1.11 , 1.07 , 1.05 , 1.02 , 1.06 , 2.03 , 1.76 , 1.7 , 1.6 , 1.53 , 1.39 , 1.39 , 1.32 , 1.26 , 1.24 , 1.32 , 1.22 , 1.22 , 1.2 , 1.19 , 1.2 , 1.2 , 1.16 , 2.2 , 1.95 , 1.9 , 1.75 , 1.64 , 1.54 , 1.47 , 1.46 , 1.42 , 1.39 , 1.45 , 1.44 , 1.42 , 1.39 , 1.39 , 1.38 , 1.39 , 1.4 , 2.44 , 2.15 , 2.07 , 2.04 , 2.03 , 2.01 , 1.99 , 1.98 , 1.98 , 1.96 , 1.94 , 1.92 , 1.92 , 1.89 , 1.9 , 1.87 , 1.87 , 1.75 , 1.7 , 1.62 , 1.51 , 1.44 , 1.41 , 1.36 , 1.36 , 1.32 , 1.45 , 1.46 , 1.48 , 1.4 , 1.5 , 1.5 , 2.6 , 2.21 , 2.15 , 2.06 , 2.0 , 1.96 , 1.9 , 1.87 , 1.8 , 1.69 , 1.6 , 1.6 , 1.6 , 1.6 , 1.6 , 1.6 , 1.6 /)

Covalent radius (in Angstroms)

Definition at line 173 of file prg_ptable_mod.F90.

9.19.3.4 real(dp), dimension(nz), parameter prg_ptable_mod::element_ea = (/ 0.75420375 , 0.0 , 0.618049 , 0.0 , 0.279723 , 1.262118 , -0.07 , 1.461112 , 3.4011887 , 0.0 , 0.547926 , 0.0 , 0.43283 , 1.389521 , 0.7465 , 2.0771029 , 3.612724 , 0.0 , 0.501459 , 0.02455 , 0.188 , 0.084 , 0.525 , 0.67584 , 0.0 , 0.151 , 0.6633 , 1.15716 , 1.23578 , 0.0 , 0.41 , 1.232712 , 0.814 , 2.02067 , 3.363588 , 0.0 , 0.485916 , 0.05206 , 0.307 , 0.426 , 0.893 , 0.7472 , 0.55 , 1.04638 , 1.14289 , 0.56214 , 1.30447 , 0.0 , 0.404 , 1.112066 , 1.047401 , 1.970875 , 3.059038 , 0.0 , 0.471626 , 0.14462 , 0.47 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.0 , 0.322 , 0.815 , 0.15 , 1.0778 , 1.56436 , 2.1251 , 2.30861 , 0.0 , 0.377 , 0.364 , 0.942363 , 1.9 , 2.8 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 /)

Electron affprg_inity (in eV)

Definition at line 235 of file prg_ptable_mod.F90.

9.19.3.5 character(50), dimension(nz), parameter prg_ptable_mod::element_econf = [character(50) :: "1s", "1s2", "1s22s", "1s22s2", "1s22s22p", "1s22s22p2", "1s22s22p3", "1s22s22p4", "1s22s22p5", "1s22s22p6", "[Ne]3s", "[Ne]3s2", "[Ne]3s23p", "[Ne]3s23p2", "[Ne]3s23p3", "[Ne]3s23p4", "[Ne]3s23p5", "[Ne]3s23p6", "[Ar]4s", "[Ar]4s2", "[Ar]3d4s2", "[Ar]3d24s2", "[Ar]3d34s2", "[Ar]3d54s", "[Ar]3d54s2", "[Ar]3d64s2", "[Ar]3d74s2", "[Ar]3d84s2", "[Ar]3d104s", "[Ar]3d104s2", "[Ar]3d104s24p", "[Ar]3d104s24p2", "[Ar]3d104s24p3", "[Ar]3d104s24p4", "[Ar]3d104s24p5", "[Ar]3d104s24p6", "[Kr]5s", "[Kr]5s2", "[Kr]4d5s2", "[Kr]4d25s2", "[Kr]4d45s", "[Kr]4d55s", "[Kr]4d55s2", "[Kr]4d75s", "[Kr]4d85s", "[Kr]4d10", "[Kr]4d105s", "[Kr]4d105s2", "[Cd]5p", "[Cd]5p2", "[Cd]5p3", "[Cd]5p4", "[Cd]5p5", "[Cd]5p6", "[Xe]6s", "[Xe]6s2", "[Xe]5d6s2", "[Xe]4f5d6s2", "[Xe]4f36s2", "[Xe]4f46s2", "[Xe]4f56s2", "[Xe]4f66s2", "[Xe]4f76s2", "[Xe]4f75d6s2", "[Xe]4f96s2", "[Xe]4f106s2", "[Xe]4f116s2", "[Xe]4f126s2", "[Xe]4f136s2", "[Xe]4f146s2", "[Xe]4f145d6s2", "[Xe]4f145d26s2", "[Xe]4f145d36s2", "[Xe]4f145d46s2", "[Xe]4f145d56s2", "[Xe]4f145d66s2", "[Xe]4f145d76s2", "[Xe]4f145d96s", "[Xe]4f145d106s", "[Xe]4f145d106s2", "[Hg]6p", "[Hg]6p2", "[Hg]6p3", "[Hg]6p4", "[Hg]6p5", "[Hg]6p6", "[Rn]7s", "[Rn]7s2", "[Rn]6d7s2", "[Rn]6d27s2", "[Rn]5f26d7s2", "[Rn]5f36d7s2", "[Rn]5f46d7s2", "[Rn]5f67s2", "[Rn]5f77s2", "[Rn]5f76d7s2", "[Rn]5f97s2", "[Rn]5f107s2", "[Rn]5f117s2", "[Rn]5f127s2", "[Rn]5f137s2", "[Rn]5f147s2", "[Rn]5f147s27p"]]

The electronic configuration.

Definition at line 360 of file prg_ptable_mod.F90.

9.19.3.6 `real(dp), dimension(nz), parameter prg_ptable_mod::element_ip = (/ 13.5984 , 24.5874 , 5.3917 , 9.3227 , 8.298 , 11.2603 , 14.5341 , 13.6181 , 17.4228 , 21.5645 , 5.1391 , 7.6462 , 5.9858 , 8.1517 , 10.4867 , 10.36 , 12.9676 , 15.7596 , 4.3407 , 6.1132 , 6.5615 , 6.8281 , 6.7462 , 6.7665 , 7.434 , 7.9024 , 7.881 , 7.6398 , 7.7264 , 9.3942 , 5.9993 , 7.8994 , 9.7886 , 9.7524 , 11.8138 , 13.9996 , 4.1771 , 5.6949 , 6.2173 , 6.6339 , 6.7589 , 7.0924 , 7.28 , 7.3605 , 7.4589 , 8.3369 , 7.5762 , 8.9938 , 5.7864 , 7.3439 , 8.6084 , 9.0096 , 10.4513 , 12.1298 , 3.8939 , 5.2117 , 5.5769 , 5.5387 , 5.473 , 5.525 , 5.582 , 5.6437 , 5.6704 , 6.1498 , 5.8638 , 5.9389 , 6.0215 , 6.1077 , 6.1843 , 6.2542 , 5.4259 , 6.8251 , 7.5496 , 7.864 , 7.8335 , 8.4382 , 8.967 , 8.9588 , 9.2255 , 10.4375 , 6.1082 , 7.4167 , 7.2855 , 8.414 , 0.0 , 10.7485 , 4.0727 , 5.2784 , 5.17 , 6.3067 , 5.89 , 6.1941 , 6.2657 , 6.026 , 5.9738 , 5.9914 , 6.1979 , 6.2817 , 6.42 , 6.5 , 6.58 , 6.65 , 4.9 /)`

Ionization energy (in eV)

Definition at line 204 of file `prg_ptable_mod.F90`.

9.19.3.7 `real(dp), dimension(nz), parameter prg_ptable_mod::element_mass = (/ 1.007825032 , 4.002603254 , 7.01600455 , 9.0121822 , 11.0093054 , 12.0 , 14.003074005 , 15.99491462 , 18.99840322 , 19.992440175 , 22.989769281 , 23.9850417 , 26.98153863 , 27.976926532 , 30.97376163 , 31.972071 , 34.96885268 , 39.962383123 , 38.96370668 , 39.96259098 , 44.9559119 , 47.9479463 , 50.9439595 , 51.9405075 , 54.9380451 , 55.9349375 , 58.933195 , 57.9353429 , 62.9295975 , 63.929142 , 68.925573 , 73.921177 , 74.921596 , 79.916521 , 78.918337 , 83.911507 , 84.911789 , 87.905612 , 88.905848 , 89.904704 , 92.906378 , 97.905408 , 97.907216 , 101.904349 , 102.905504 , 105.903486 , 106.905097 , 113.903358 , 114.903878 , 119.902194 , 120.903815 , 129.906224 , 126.904473 , 131.904153 , 132.905451 , 137.905247 , 138.906353 , 139.905438 , 140.907652 , 141.907723 , 144.912749 , 151.919732 , 152.92123 , 157.924103 , 158.925346 , 163.929174 , 164.930322 , 165.930293 , 168.934213 , 173.938862 , 174.940771 , 179.94655 , 180.947995 , 183.950931 , 186.955753 , 191.96148 , 192.962926 , 194.964791 , 196.966568 , 201.970643 , 204.974427 , 207.976652 , 208.980398 , 208.98243 , 209.987148 , 222.017577 , 223.019735 , 226.025409 , 227.027752 , 232.038055 , 231.035884 , 238.050788 , 237.048173 , 244.064204 , 243.061381 , 247.070354 , 247.070307 , 251.079587 , 252.08298 , 257.095105 , 258.098431 , 259.10103 , 262.10963 /)`

Element mass in atomic mass units (1.66×10^{-27} kg)

Definition at line 110 of file `prg_ptable_mod.F90`.

9.19.3.8 `integer, dimension(nz), parameter prg_ptable_mod::element_maxbonds = (/ 1 , 0 , 1 , 2 , 4 , 4 , 4 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 8 , 6 , 6 , 6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 12 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 6 /)`

The maximum expected number of bonds to this element.

Definition at line 297 of file `prg_ptable_mod.F90`.

9.19.3.9 `character(20), dimension(nz), parameter prg_ptable_mod::element_name = [character(20) :: "Hydrogen" , "Helium" , "Lithium" , "Beryllium" , "Boron" , "Carbon" , "Nitrogen" , "Oxygen" , "Fluorine" , "Neon" , "Sodium" , "Magnesium" , "Aluminium" , "Silicon" , "Phosphorus" , "Sulfur" , "Chlorine" , "Argon" , "Potassium" , "Calcium" , "Scandium" , "Titanium" , "Vanadium" , "Chromium" , "Manganese" , "Iron" , "Cobalt" , "Nickel" , "Copper" , "Zinc" , "Gallium" , "Germanium" , "Arsenic" , "Selenium" , "Bromine" , "Krypton" , "Rubidium" , "Strontium" , "Yttrium" , "Zirconium" , "Niobium" , "Molybdenum" , "Technetium" , "Ruthenium" , "Rhodium" , "Palladium" , "Silver" , "Cadmium" , "Indium" , "Tin" , "Antimony" , "Tellurium" , "Iodine" , "Xenon" , "Caesium" , "Barium" , "Lanthanum" , "Cerium" , "Praseodymium" , "Neodymium" , "Promethium" , "Samarium" , "Europium" , "Gadolinium" , "Terbium" , "Dysprosium" , "Holmium" , "Erbium" , "Thulium" , "Ytterbium" , "Lutetium" , "Hafnium" , "Tantalum" , "Tungsten" , "Rhenium" , "Osmium" , "Iridium" , "Platinum" , "Gold" , "Mercury" , "Thallium" , "Lead" , "Bismuth" , "Polonium" , "Astatine" , "Radon" , "Francium" , "Radium" , "Actinium" , "Thorium" , "Protactinium" , "Uranium" , "Neptunium" , "Plutonium" , "Americium" , "Curium" , "Berkelium" , "Californium" , "Einsteinium" , "Fermium" , "Mendelevium" , "Nobelium" , "Lawrencium"]`

Element name.

Definition at line 79 of file `prg_ptable_mod.F90`.

9.19.3.10 integer, dimension(nz), parameter prg_ptable_mod::element_numel = (/ 1, 2, 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 /)

Last shell number of electrons.

Definition at line 329 of file prg_ptable_mod.F90.

9.19.3.11 character(2), dimension(nz), parameter prg_ptable_mod::element_symbol = [character(2) :: "H", "He", "Li", "Be", "B", "C", "N", "O", "F", "Ne", "Na", "Mg", "Al", "Si", "P", "S", "Cl", "Ar", "K", "Ca", "Sc", "Ti", "V", "Cr", "Mn", "Fe", "Co", "Ni", "Cu", "Zn", "Ga", "Ge", "As", "Se", "Br", "Kr", "Rb", "Sr", "Y", "Zr", "Nb", "Mo", "Tc", "Ru", "Rh", "Pd", "Ag", "Cd", "In", "Sn", "Sb", "Te", "I", "Xe", "Cs", "Ba", "La", "Ce", "Pr", "Nd", "Pm", "Sm", "Eu", "Gd", "Tb", "Dy", "Ho", "Er", "Tm", "Yb", "Lu", "Hf", "Ta", "W", "Re", "Os", "Ir", "Pt", "Au", "Hg", "Tl", "Pb", "Bi", "Po", "At", "Rn", "Fr", "Ra", "Ac", "Th", "Pa", "U", "Np", "Pu", "Am", "Cm", "Bk", "Cf", "Es", "Fm", "Md", "No", "Lr"]

Element symbol.

Definition at line 17 of file prg_ptable_mod.F90.

9.19.3.12 character(2), dimension(nz), parameter prg_ptable_mod::element_symbol_upper = [character(2) :: "H", "HE", "LI", "BE", "B", "C", "N", "O", "F", "NE", "NA", "MG", "AL", "SI", "P", "S", "CL", "AR", "K", "CA", "SC", "TI", "V", "CR", "MN", "FE", "CO", "NI", "CU", "ZN", "GA", "GE", "AS", "SE", "BR", "KR", "RB", "SR", "Y", "ZR", "NB", "MO", "TC", "RU", "RH", "PD", "AG", "CD", "IN", "SN", "SB", "TE", "I", "XE", "CS", "BA", "LA", "CE", "PR", "ND", "PM", "SM", "EU", "GD", "TB", "DY", "HO", "ER", "TM", "YB", "LU", "HF", "TA", "W", "RE", "OS", "IR", "PT", "AU", "HG", "TL", "PB", "BI", "PO", "AT", "RN", "FR", "RA", "AC", "TH", "PA", "U", "NP", "PU", "AM", "CM", "BK", "CF", "ES", "FM", "MD", "NO", "LR"]

Element symbol upper.

Definition at line 48 of file prg_ptable_mod.F90.

9.19.3.13 real(dp), dimension(nz), parameter prg_ptable_mod::element_vdwr = (/ 1.1, 1.4, 1.81, 1.53, 1.92, 1.7, 1.55, 1.52, 1.47, 1.54, 2.27, 1.73, 1.84, 2.1, 1.8, 1.8, 1.75, 1.88, 2.75, 2.31, 2.3, 2.15, 2.05, 2.05, 2.05, 2.05, 2.0, 2.0, 2.0, 2.1, 1.87, 2.11, 1.85, 1.9, 1.83, 2.02, 3.03, 2.49, 2.4, 2.3, 2.15, 2.1, 2.05, 2.05, 2.0, 2.05, 2.1, 2.2, 2.2, 1.93, 2.17, 2.06, 1.98, 2.16, 3.43, 2.68, 2.5, 2.48, 2.47, 2.45, 2.43, 2.42, 2.4, 2.38, 2.37, 2.35, 2.33, 2.32, 2.3, 2.28, 2.27, 2.25, 2.2, 2.1, 2.05, 2.0, 2.0, 2.05, 2.1, 2.05, 1.96, 2.02, 2.07, 1.97, 2.02, 2.2, 3.48, 2.83, 2.0, 2.4, 2.0, 2.3, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0 /)

van der Waals radius (in Angstroms)

Definition at line 141 of file prg_ptable_mod.F90.

9.19.3.14 integer, parameter prg_ptable_mod::nz = 103

Definition at line 12 of file prg_ptable_mod.F90.

9.20 prg_pulaycomponent_mod Module Reference

Produces a matrix to get the Pulay Component of the forces.

Functions/Subroutines

- subroutine, public [prg_pulaycomponent0](#) (rho_bml, ham_bml, pcm_bml, threshold, M, bml_type, verbose)
 $At T = 0K, P = \rho H \rho.$
- subroutine, public [prg_pulaycomponentt](#) (rho_bml, ham_bml, zmat_bml, pcm_bml, threshold, M, bml_type, verbose)
 $At T > 0K, P = \rho H S^{-1} + S^{-1} H \rho.$
- subroutine, public [prg_get_pulayforce](#) (nats, zmat_bml, ham_bml, rho_bml, dSx_bml, dSy_bml, dSz_bml, hindex, FPUL, threshold)
Pulay Force FPUL from $2Tr[ZZ'HD\frac{dS}{dR}]$.

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.20.1 Detailed Description

Produces a matrix to get the Pulay Component of the forces.

For a further explanation please see Niklasson 2008 [\[3\]](#)

9.20.2 Function/Subroutine Documentation

- 9.20.2.1 subroutine, public prg_pulaycomponent_mod::prg_get_pulayforce (integer, intent(in) *nats*, type(bml_matrix_t), intent(in) *zmat_bml*, type(bml_matrix_t), intent(in) *ham_bml*, type(bml_matrix_t), intent(in) *rho_bml*, type(bml_matrix_t), intent(in) *dSx_bml*, type(bml_matrix_t), intent(in) *dSy_bml*, type(bml_matrix_t), intent(in) *dSz_bml*, integer, dimension(:,:), intent(in) *hindex*, real(dp), dimension(:,:), intent(inout), allocatable *FPUL*, real(dp), intent(in) *threshold*)

Pulay Force FPUL from $2Tr[ZZ'HD\frac{dS}{dR}]$.

Parameters

<i>nats</i>	Number of atoms.
<i>zmat_bml</i>	Congruence transform in bml format.
<i>rho_bml</i>	Density matrix.
<i>dSx_bml</i>	x derivative of S.
<i>dSy_bml</i>	y derivative of S.
<i>dSz_bml</i>	z derivative of S.
<i>hindex</i>	Contains the Hamiltonian indices for every atom (see get_hindex).

Definition at line 152 of file prg_pulaycomponent_mod.F90.

9.20.2.2 subroutine, public prg_pulaycomponent_mod::prg_pulaycomponent0 (type(bml_matrix_t), intent(in) *rho_bml*, type(bml_matrix_t), intent(in) *ham_bml*, type(bml_matrix_t), intent(inout) *pcm_bml*, real(dp), intent(in) *threshold*, integer, intent(in) *M*, character(20), intent(in) *bml_type*, integer *verbose*)

At $T = 0K$, $P = \rho H \rho$.

Parameters

<i>rho_bml</i>	Density matrix in bml format.
<i>ham_bml</i>	Hamiltonian matrix in bml format.
<i>pcm_bml</i>	Pulay matrix output in bml format.
<i>threshold</i>	Threshold for the matrix elements.
<i>M</i>	Maximum nonzero values per row.
<i>bml_type</i>	Bml format type.
<i>verbose</i>	Verbosity level.

Todo M and bml_type will have to be removed from the input parameter.

Definition at line 32 of file prg_pulaycomponent_mod.F90.

9.20.2.3 subroutine, public prg_pulaycomponent_mod::prg_pulaycomponentt (type(bml_matrix_t), intent(in) *rho_bml*, type(bml_matrix_t), intent(in) *ham_bml*, type(bml_matrix_t), intent(in) *zmat_bml*, type(bml_matrix_t), intent(inout) *pcm_bml*, real(dp), intent(in) *threshold*, integer, intent(in) *M*, character(20), intent(in) *bml_type*, integer *verbose*)

At $T > 0K$, $P = \rho H S^{-1} + S^{-1} H \rho$.

Parameters

<i>rho_bml</i>	Density matrix in bml format.
<i>ham_bml</i>	Hamiltonian matrix in bml format.
<i>Z_bml</i>	Congruence transform in bml format.
<i>pcm_bml</i>	Pulay matrix output in bml format.
<i>threshold</i>	Threshold for the matrix elements.
<i>M</i>	Maximum nonzero values per row.
<i>bml_type</i>	Bml format type.
<i>verbose</i>	Verbosity level.

Todo M and bml_type will have to be removed from the input parameter.

Definition at line 83 of file prg_pulaycomponent_mod.F90.

9.20.3 Variable Documentation

9.20.3.1 integer, parameter prg_pulaycomponent_mod::dp = kind(1.0d0) [private]

Definition at line 13 of file prg_pulaycomponent_mod.F90.

9.21 prg_pulaymixer_mod Module Reference

Pulay mixer mode.

Data Types

- type [mx_type](#)

Functions/Subroutines

- subroutine, public [prg_parse_mixer](#) (input, filename)
The parser for the mixer routines.
- subroutine, public [prg_qmixer](#) (charges, oldcharges, dqin, dqout, scferror, piter, pulaycoef, mpulay, verbose)
Mixing the charges to accelerate scf convergence.
- subroutine, public [prg_linearmixer](#) (charges, oldcharges, scferror, linmixcoef, verbose)
Routine to perform linear mixing.

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.21.1 Detailed Description

Pulay mixer mode.

Gets the best coefficient for mixing the charges during scf.

Todo add the density matrix mixer.

9.21.2 Function/Subroutine Documentation

9.21.2.1 subroutine, public prg_pulaymixer_mod::prg_linearmixer (real([dp](#)), dimension(:), intent(inout), allocatable *charges*, real([dp](#)), dimension(:), intent(inout), allocatable *oldcharges*, real([dp](#)), intent(inout) *scferror*, real([dp](#)), intent(in) *linmixcoef*, integer, intent(in) *verbose*)

Routine to perform linear mixing.

Parameters

<i>charges</i>	Actual charges of the system.
<i>oldcharges</i>	Previous scf charges.
<i>scferror</i>	SCF error.
<i>linmixcoef</i>	Mixing coefficient.
<i>verbose</i>	Verbosity level.

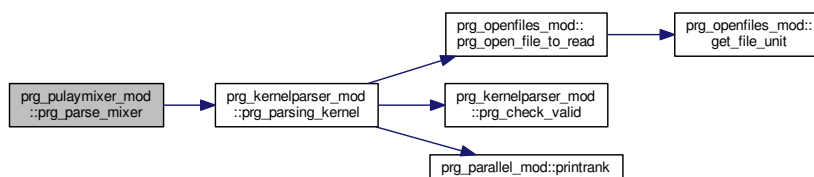
Definition at line 237 of file prg_pulaymixer_mod.F90.

9.21.2.2 subroutine, public prg_pulaymixer_mod::prg_parse_mixer (type(mx_type), intent(inout) *input*, character(len=*) *filename*)

The parser for the mixer routines.

Definition at line 42 of file prg_pulaymixer_mod.F90.

Here is the call graph for this function:



9.21.2.3 subroutine, public prg_pulaymixer_mod::prg_qmixer (real(dp), dimension(:), intent(inout) *charges*, real(dp), dimension(:), intent(inout), allocatable *oldcharges*, real(dp), dimension(:,:), intent(inout), allocatable *dqin*, real(dp), dimension(:,:), intent(inout), allocatable *dqout*, real(dp), intent(inout) *scferror*, integer *piter*, real(dp), intent(in) *pulaycoef*, integer, intent(in) *mpulay*, integer, intent(in) *verbose*)

Mixing the charges to accelerate scf convergence.

Parameters

<i>charges</i>	System charges.
<i>oldcharges</i>	Old charges of the system.
<i>dqin</i>	Matrix for charges history in.
<i>dqout</i>	Matrix for charges history out.
<i>scferror</i>	SCF error.
<i>piter</i>	scf iteration number.
<i>pulaycoef</i>	Coefficient for pulay mixing (generally between 0.01 and 0.1).
<i>mpulay</i>	Number of matrices stored (generally 3-5).
<i>verbose</i>	Different levels of verbosity.

Definition at line 103 of file prg_pulaymixer_mod.F90.

9.21.3 Variable Documentation

9.21.3.1 integer, parameter prg_pulaymixer_mod::dp = kind(1.0d0) [private]

Definition at line 14 of file prg_pulaymixer_mod.F90.

9.22 prg_quantumdynamics_mod Module Reference

A module to add in common quantum dynamical operations.

Functions/Subroutines

- subroutine, public [prg_kick_density](#) (kick_dirac, kick_mag, dens, norbs, mdim, S, SINV, which_atom, r, bml_type, thresh)
Provides perturbation to initial density matrix in the form of an electric field kick. This routine does: $\rho_{kick} = \exp \frac{-i}{\hbar} \hat{V} \hat{\rho} \hat{S} \exp \frac{i}{\hbar} \hat{V} \hat{S}^{-1}$ where \hat{V} is the field disturbance.
- subroutine, public [prg_get_sparsity_cplxmat](#) (matrix_type, element_type, thresh, a_dense)
This computes the sparsity of a complex matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ where f is the sparsity, N_0 is the number of values less than the threshold, and N_{tot} is the total number of values. The sparsity and threshold are printed to the screen.
- subroutine, public [prg_get_sparsity_realmat](#) (matrix_type, element_type, thresh, a_dense)
This computes the sparsity of a real matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ where f is the sparsity, N_0 is the number of values less than the threshold, and N_{tot} is the total number of values. The sparsity and threshold are printed to the screen.
- subroutine, public [prg_kick_density_bml](#) (kick_dirac, kick_mag, rho_bml, s_bml, sinv_bml, mdim, which_atom, r, matrix_type, thresh)
Provides perturbation to initial density matrix in the form of an electric field kick given input matrices in BML format. This routine does: $\rho_{kick} = \exp \frac{-i}{\hbar} \hat{V} \hat{\rho} \hat{S} \exp \frac{i}{\hbar} \hat{V} \hat{S}^{-1}$ where \hat{V} is the field disturbance.
- subroutine, public [prg_lvni_bml](#) (h1_bml, sinv_bml, dt, hbar, rhoold_bml, rho_bml, aux_bml, matrix_type, mdim, thresh)
Performs Liouville-von Neumann integration using leap-frog method. This routine does: $\hat{\rho}(t + \Delta t) = \hat{\rho}(t - \Delta t) + 2\Delta t \frac{\partial \hat{\rho}(t)}{\partial t}$ where the time derivative of the density matrix is defined as follows: $\frac{\partial \hat{\rho}(t)}{\partial t} = \frac{-i}{\hbar} (S^{-1} \hat{H}(t) \hat{\rho}(t) - \hat{\rho}(t) \hat{H}(t) S^{-1})$.
- subroutine, public [prg_getcharge](#) (rho_bml, s_bml, charges, aux_bml, z, spindex, N, nats, thresh)
Constructs the charges from the density matrix.
- subroutine, public [prg_getdipole](#) (charges, r, mu)
This routine computes the dipole moment of the system with units determined by the units of the coordinate matrix and charges given.
- subroutine, public [prg_excitation](#) (fill_mat, orbit_orig, orbit_exc)
Produce an excitation in the initially calculated density matrix to.

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.22.1 Detailed Description

A module to add in common quantum dynamical operations.

This module contains routines that perform the following tasks: apply a apply an excitation or perturbation to the initial density matrix, compute the comutator of two two matrices, calculate the sparsity of a real or complex matrix, and time evolve a density matrix using Liouville-von Neumann equation with the leap-frog method of integration.

Author

J.B. Grindstaff (grindstaff@lanl.gov)
 Alicia Rae Welden (welden@lanl.gov)

9.22.2 Function/Subroutine Documentation

9.22.2.1 subroutine, public prg_quantumdynamics_mod::prg_excitation (integer, dimension(:), intent(inout) *fill_mat*, integer, intent(in) *orbit_orig*, integer, intent(in) *orbit_exci*)

Produce an excitation in the initially calculated density matrix to.

Definition at line 312 of file prg_quantumdynamics_mod.F90.

9.22.2.2 subroutine, public prg_quantumdynamics_mod::prg_get_sparsity_cplxmat (character(len=*) , intent(in) *matrix_type*, character(len=*) , intent(in) *element_type*, real(dp), intent(in) *thresh*, complex(dp), dimension(:, :), intent(in) *a_dense*)

This computes the sparsity of a complex matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ where f is the sparsity, N_0 is the number of values less than the threshold, and N_{tot} is the total number of values. The sparsity and threshold are printed to the screen.

Parameters

<i>matrix_type</i>	the BML matrix type
<i>element_type</i>	the BML element type
<i>thresh</i>	the threshold for sparsity evaluation
<i>a_dense</i>	the dense complex matrix to be evaluated for sparsity

Definition at line 103 of file prg_quantumdynamics_mod.F90.

9.22.2.3 subroutine, public prg_quantumdynamics_mod::prg_get_sparsity_realmat (character(len=*) , intent(in) *matrix_type*, character(len=*) , intent(in) *element_type*, real(dp), intent(in) *thresh*, real(dp), dimension(:, :), intent(in) *a_dense*)

This computes the sparsity of a real matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ where f is the sparsity, N_0 is the number of values less than the threshold, and N_{tot} is the total number of values. The sparsity and threshold are printed to the screen.

Parameters

<i>matrix_type</i>	the BML matrix type
<i>element_type</i>	the BML element type
<i>thresh</i>	the threshold for sparsity evaluation
<i>a_dense</i>	the dense real matrix to be evaluated for sparsity

Definition at line 127 of file prg_quantumdynamics_mod.F90.

9.22.2.4 subroutine, public prg_quantumdynamics_mod::prg_getcharge (type(bml_matrix_t), intent(in) *rho_bml*, type(bml_matrix_t), intent(in) *s_bml*, real(dp), dimension(:), allocatable *charges*, type(bml_matrix_t) *aux_bml*, real(dp), dimension(:), intent(in) *z*, integer, dimension(:), intent(in), allocatable *spindex*, integer, dimension(:), intent(in), allocatable *N*, integer *nats*, real(dp), intent(in) *thresh*)

Constructs the charges from the density matrix.

Parameters

<i>rho_bml</i>	Density matrix in BML format.
<i>over_bml</i>	Overlap matrix in BML format.
<i>charges</i>	the array of charges.
<i>aux_bml</i>	the auxiliary matrix in BML format.
<i>spindex</i>	Start and end index for every atom in the system.
<i>z</i>	
<i>nats</i>	the number of atoms
<i>N</i>	
<i>thresh</i>	threshold for the BML matrix

Definition at line 252 of file prg_quantumdynamics_mod.F90.

9.22.2.5 subroutine, public prg_quantumdynamics_mod::prg_getdipole (real(dp), dimension(:), intent(in) *charges*, real(dp), dimension(:, :), intent(in) *r*, real(dp), dimension(3), intent(inout) *mu*)

This routine computes the dipole moment of the system with units determined by the units of the coordinate matrix and charges given.

Parameters

<i>charges</i>	Charge on each atom.
<i>r</i>	Coordinate matrix of the atoms.
<i>p</i>	Dipole moment vector.

Definition at line 287 of file prg_quantumdynamics_mod.F90.

9.22.2.6 subroutine, public prg_quantumdynamics_mod::prg_kick_density (integer, intent(in) *kick_dirac*, real(dp) *kick_mag*, complex(dp), dimension(:, :), intent(inout), allocatable *dens*, integer, intent(in) *norbs*, integer, intent(in) *mdim*, complex(dp), dimension(:, :), allocatable *S*, complex(dp), dimension(:, :), allocatable *SINV*, integer, dimension(:), intent(in), allocatable *which_atom*, real(dp), dimension(:, :), allocatable *r*, character(len=*) , intent(in) *bmltype*, real(dp) *thresh*)

Provides perturbation to initial density matrix in the form of an electric field kick. This routine does: $\rho_{kick} = \exp \frac{i}{\hbar} \hat{V} \hat{\rho} \hat{S} \exp \frac{i}{\hbar} \hat{V} \hat{S}^{-1}$ where \hat{V} is the field disturbance.

Parameters

<i>kick_dirac</i>	the direction of the kick in the electric field
<i>kick_mag</i>	the magnitude of the kick in the electric field
<i>dens</i>	the initial density matrix to be kicked.
<i>norbs</i>	the number of orbitals in the density matrix
<i>S</i>	the overlap matrix
<i>SINV</i>	the inverse of the overlap matrix
<i>which_atom</i>	vector containing atom identification
<i>r</i>	direction vector for kick based on atom and <i>kick_dirac</i>
<i>bmltype</i>	type of BML matrix desired for faster computation
<i>thresh</i>	threshold for BML matrix conversion

Definition at line 48 of file prg_quantumdynamics_mod.F90.

9.22.2.7 subroutine, public prg_quantumdynamics_mod::prg_kick_density_bml (integer, intent(in) *kick_dirac*, real(dp) *kick_mag*, type(bml_matrix_t) *rho_bml*, type(bml_matrix_t) *s_bml*, type(bml_matrix_t) *sinv_bml*, integer *mdim*, integer, dimension(:), intent(in), allocatable *which_atom*, real(dp), dimension(:, :), allocatable *r*, character(len=*) , intent(in) *matrix_type*, real(dp) *thresh*)

Provides perturbation to initial density matrix in the form of an electric field kick given input matrices in BML format. This routine does: $\rho_{kick} = \exp \frac{-i}{\hbar} \hat{V} \hat{\rho} \hat{S} \exp \frac{i}{\hbar} \hat{V} \hat{S}^{-1}$ where \hat{V} is the field disturbance.

Parameters

<i>kick_dirac</i>	the direction of the kick in the electric field
<i>kick_mag</i>	the magnitude of the kick in the electric field
<i>rho_bml</i>	the initial density matrix to be kicked in BML format.
<i>s_bml</i>	the overlap matrix
<i>sinv_bml</i>	the inverse of the overlap matrix
<i>mdim</i>	maximum number of nonzero values per row in BML matrix
<i>which_atom</i>	vector containing atom identification
<i>r</i>	position vector for kicked atom
<i>matrix_type</i>	the type of BML format
<i>thresh</i>	the threshold for the BML matrix

Definition at line 159 of file prg_quantumdynamics_mod.F90.

9.22.2.8 subroutine, public prg_quantumdynamics_mod::prg_lvni_bml (type(bml_matrix_t) *h1_bml*, type(bml_matrix_t) *sinv_bml*, real(dp) *dt*, real(dp) *hbar*, type(bml_matrix_t) *rhoold_bml*, type(bml_matrix_t) *rho_bml*, type(bml_matrix_t) *aux_bml*, character(len=*) , intent(in) *matrix_type*, integer *mdim*, real(dp), intent(in) *thresh*)

Performs Liouville-von Neumann integration using leap-frog method. This routine does: $\hat{\rho}(t + \Delta t) = \hat{\rho}(t - \Delta t) + 2\Delta t \frac{\partial \hat{\rho}(t)}{\partial t}$ where the time derivative of the density matrix is defined as follows: $\frac{\partial \hat{\rho}(t)}{\partial t} = \frac{-i}{\hbar} \left(S^{-1} \hat{H}(t) \hat{\rho}(t) - \hat{\rho}(t) \hat{H}(t) S^{-1} \right)$.

Parameters

<i>H</i>	the Hamiltonian matrix at time t
<i>sinv_bml</i>	the inverse overlap matrix
<i>dt</i>	the timestep for integration
<i>hbar</i>	the Dirac constant (generally taken to be 1 in simulation units)
<i>rho_old</i>	the density matrix at previous time-step
<i>rho_bml</i>	the density matrix at current time-step
<i>aux_bml</i>	the temp matrix used for value storage during computation
<i>matrix_type</i>	the type of BML matrix
<i>thresh</i>	the threshold for the BML matrix

Definition at line 216 of file prg_quantumdynamics_mod.F90.

9.22.3 Variable Documentation

9.22.3.1 integer, parameter prg_quantumdynamics_mod::dp = kind(1.0d0) [private]

Definition at line 19 of file prg_quantumdynamics_mod.F90.

9.23 prg_response_mod Module Reference

Module to compute the density matrix response and related quantities.

Data Types

- type [respdata_type](#)

Functions/Subroutines

- subroutine, public [prg_parse_response](#) (RespData, filename)
The parser for the calculation of the DM response.
- subroutine, public [prg_compute_dipole](#) (charges, coordinate, dipoleMoment, factor, verbose)
To compute the dipole moment of the system. The units of the dipole moment are determined by the units of the coordinates and charges that are given.
- subroutine, public [prg_write_dipole_tcl](#) (dipoleMoment, file, factor, verbose)
To visualize a dipole moment using VMD. This will prg_generate a .tcl script that could be run using VMD To visualize with VMD: \$ vmd -e dipole.tcl.
- subroutine, public [prg_compute_polarizability](#) (rsp_bml, prt_bml, polarizability, factor, verbose)
To compute the polarizability of the system. The units of the directional polarizability are determined by the units of the perturbation and Hamiltonian. This equation can be found in [5] equation 4a. Note that in equation 4a of the reference there is a 2 that account for the double occupancy which is not present in this case cause the density matrix construction is done by taking the occupancy into account.
- subroutine, public [prg_pert_from_file](#) (prt_bml, norb)
Read perturbation from file.
- subroutine, public [prg_compute_response_rs](#) (ham_bml, prt_bml, rsp_bml, lambda, bndfil, threshold, verbose)
Computes the first order response density matrix using Rayleigh Schrodinger Perturbation theory The transformation hereby performed are:
- subroutine, public [prg_compute_response_fd](#) (ham_bml, prt_bml, rsp_bml, prg_delta, bndfil, threshold, verbose)
Computes the first order response density matrix using finite differences. The transformation hereby performed are:
- subroutine, public [prg_pert_constant_field](#) (field, intensity, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)
Apply a constant field perturbation through the dipole moment operator ($\hat{\mu} = e\hat{r}$). In the matrix representation, this is: $H^{(1)} = \lambda \frac{1}{2} (S e \mathbf{r} \cdot \mathbf{E} + e \mathbf{r} \cdot \mathbf{E} S)$. The symmetrization is done in order to preserve the Hermiticity of H . In this case the whole system will be affected by the field. In a latter version we will add the possibility of applying this field to a region of the system. In this implementation $e = 1$ and units can be transformed by using the parameter λ .
- subroutine, public [prg_pert_sin_pot](#) (direction, lx, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)
Apply a sinusoidal length dependent potential ($\sin(\tilde{\mathbf{r}}_x)$) where \mathbf{r}_x is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2} \lambda (S \sin(\tilde{\mathbf{r}}_x) + \sin(\tilde{\mathbf{r}}_x) S)$. $\tilde{\mathbf{r}}_x = 2\pi(\mathbf{r}/l_x) - \pi$. The symmetrization is done in order to preserve the Hermiticity of H . Units can be transformed by using the parameter λ .

- subroutine, public `prg_pert_cos_pot` (direction, lx, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)
Apply a cosine length dependent potential ($\cos(\tilde{r}_x)$) where \tilde{r}_x is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2}\lambda(S\sin(\tilde{r}_x) + \sin(\tilde{r}_x)S)$. $\tilde{r}_x = 2\pi(\mathbf{r}/l_x) - \pi$. The symmetrization is done in order to preserve the Hermiticity of H . Units can be transformed by using the parameter λ .
- subroutine, public `prg_compute_response_sp2` (ham_bml, prt_bml, rsp_bml, rho_bml, lambda, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtoll, threshold, verbose)
Finds the first order response matrix from a Hamiltonian matrix.
- subroutine, public `prg_project_response` (rsp_bml, over_bml, spindex, norbi, coordinates, rspfunc, verbose)
Project the response onto atomic positions. First order response to the perturbation ($\rho^{(1)}$) projected onto the atomic position. Basically: $rsp(i) = \sum_{\alpha \in i} \rho_{\alpha\alpha}^{(1)}$, where orbital α belong to atom i .

Variables

- integer, parameter `dp` = kind(1.0d0)
- real(`dp`), parameter `pi` = 3.14159265358979323846264338327950_dp

9.23.1 Detailed Description

Module to compute the density matrix response and related quantities.

Todo Add the response scf

Change name response_SP2 to dm_prt_response

Change name response_rs to rs_prt_response

More information about the theory can be found at [4] and Niklasson2015

9.23.2 Function/Subroutine Documentation

9.23.2.1 subroutine, public `prg_response_mod::prg_compute_dipole` (real(`dp`), dimension(:), intent(in) *charges*, real(`dp`), dimension(:, :), intent(in) *coordinate*, real(`dp`), dimension(3), intent(inout) *dipoleMoment*, real(`dp`), intent(in) *factor*, integer *verbose*)

To compute the dipole moment of the system. The units of the dipole moment are determined by the units of the coordinates and charges that are given.

Parameters

<i>charges</i>	Charges on each atomic position.
<i>coordinate</i>	Coordinates of the atoms.
<i>nats</i>	Number of atoms.
<i>dipoleMoment</i>	Dipole moment vector.
<i>factor</i>	Unit conversion factor (use 1.0 if no conversion is required).
<i>verbose</i>	To give different verbosity levels. If coordinates are in Å and charges are in fractions of electron, then transformation ea2debye from LATTE lib can be used to change units to Debye.

Definition at line 120 of file `prg_response_mod.F90`.

9.23.2.2 subroutine, public prg_response_mod::prg_compute_polarizability (type(bml_matrix_t), intent(in) *rsp_bml*, type(bml_matrix_t), intent(in) *prt_bml*, real(dp), intent(inout) *polarizability*, real(dp), intent(in) *factor*, integer *verbose*)

To compute the polarizability of the system. The units of the directional polarizability are determined by the units of the perturbation and Hamiltonian. This equation can be found in [5] equation 4a. Note that in equation 4a of the reference there is a 2 that account for the double occupancy which is not present in this case cause the density matrix construction is done by taking the occupancy into account.

Parameters

<i>charges</i>	Charges on each atomic position.
<i>coordinate</i>	Coordinates of the atoms.
<i>nats</i>	Number of atoms.
<i>dipoleMoment</i>	Dipole moment vector.
<i>factor</i>	Unit conversion factor (use 1.0 is no conversion is required).
<i>verbose</i>	To give different verbosity levels. If coordinates are in Å and charges are in fractions of electron, then transformation ea2debye from LATTE lib can be used to change units to Debye.

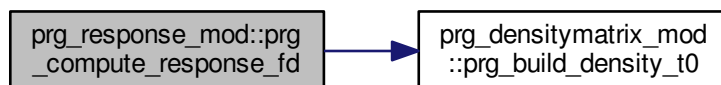
Definition at line 198 of file prg_response_mod.F90.

9.23.2.3 subroutine, public prg_response_mod::prg_compute_response_fd (type(bml_matrix_t), intent(in) *ham_bml*, type(bml_matrix_t), intent(in) *prt_bml*, type(bml_matrix_t), intent(inout) *rsp_bml*, real(dp) *prg_delta*, real(dp), intent(in) *bndfil*, real(dp), intent(in) *threshold*, integer *verbose*)

Computes the first order response density matrix using finite differences. The transformation hereby performed are:

Definition at line 379 of file prg_response_mod.F90.

Here is the call graph for this function:



9.23.2.4 subroutine, public prg_response_mod::prg_compute_response_rs (type(bml_matrix_t), intent(in) *ham_bml*, type(bml_matrix_t), intent(in) *prt_bml*, type(bml_matrix_t), intent(inout) *rsp_bml*, real(dp) *lambda*, real(dp), intent(in) *bndfil*, real(dp), intent(in) *threshold*, integer *verbose*)

Computes the first order response density matrix using Rayleigh Schrodinger Perturbation theory The transformation hereby performed are:

Definition at line 249 of file prg_response_mod.F90.

9.23.2.5 subroutine, public prg_response_mod::prg_compute_response_sp2 (type(bml_matrix_t), intent(in) *ham_bml*, type(bml_matrix_t), intent(in) *pri_bml*, type(bml_matrix_t), intent(inout) *rsp_bml*, type(bml_matrix_t), intent(inout) *rho_bml*, real(dp) *lambda*, real(dp), intent(in) *bndfil*, integer, intent(in) *minsp2iter*, integer, intent(in) *maxsp2iter*, character(len=*), intent(in) *sp2conv*, real(dp), intent(in) *idemtol*, real(dp), intent(in) *threshold*, integer *verbose*)

Finds the first order response matrix from a Hamiltonian matrix.

Definition at line 652 of file prg_response_mod.F90.

9.23.2.6 subroutine, public prg_response_mod::prg_parse_response (type(respdata_type) *RespData*, character(len=*) *filename*)

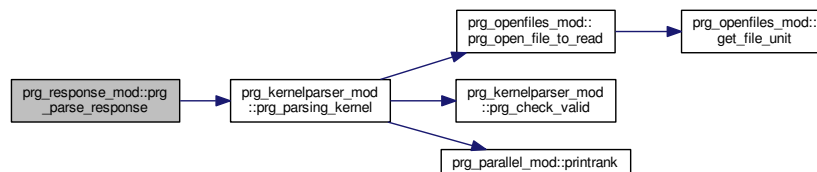
The parser for the calculation of the DM response.

Parameters

<i>RespData</i>	Response data type.
<i>filename</i>	Name of the file to parse.

Definition at line 45 of file prg_response_mod.F90.

Here is the call graph for this function:



9.23.2.7 subroutine, public prg_response_mod::prg_pert_constant_field (real(dp), dimension(3), intent(in) *field*, real(dp) *intensity*, real(dp), dimension(:,:), intent(in) *coordinate*, real(dp) *lambda*, type(bml_matrix_t), intent(inout) *pri_bml*, real(dp) *threshold*, integer, dimension(:), intent(in) *spindex*, integer, dimension(:), intent(in) *norbi*, integer, intent(in) *verbose*, type(bml_matrix_t), intent(in), optional *over_bml*)

Apply a constant field perturbation through the dipole moment operator ($\hat{\mu} = e\hat{\mathbf{r}}$). In the matrix representation, this is: $H^{(1)} = \lambda \frac{1}{2} (S \mathbf{e} \mathbf{r} \cdot \mathbf{E} + \mathbf{e} \mathbf{r} \cdot \mathbf{E} S)$. The symmetrization is done in order to preserve the Hermiticity of H. In this case the whole system will be affected by the field. In a latter version we will add the possibility of applying this field to a region of the system. In this implementation $e = 1$ and units can be transformed by using the parameter λ .

Note

If the Hamiltonian is already in the prg_orthogonalized form, then parameter *over_bml* can be omitted.

Parameters

<i>field</i>	Direction of the applied field ($\hat{\mathbf{E}}$).
<i>intensity</i>	Intensity of the field ($ \mathbf{E} $)..
<i>coordinate</i>	Coordinates of the system (\mathbf{r}).
<i>lambda</i>	Constant to premultiply the perturbation (λ).
<i>prt_bml</i>	Perturbation in bml format ($H^{(1)}$).
<i>threshold</i>	Threshold value for bml format matrices.
<i>spindex</i>	Species index. It gives the species index of a particular atom.
<i>norbi</i>	Number of orbitals for each atomic site.
<i>verbose</i>	Different levels of verbosity.
<i>over_bml</i>	It has to be present for a nonorthogonal representation (S).

Definition at line 446 of file prg_response_mod.F90.

9.23.2.8 subroutine, public prg_response_mod::prg_pert_cos_pot (character *direction*, real(dp) *lx*, real(dp), dimension(:,:), intent(in) *coordinate*, real(dp) *lambda*, type(bml_matrix_t), intent(inout) *prt_bml*, real(dp) *threshold*, integer, dimension(:), intent(in) *spindex*, integer, dimension(:), intent(in) *norbi*, integer, intent(in) *verbose*, type(bml_matrix_t), intent(in), optional *over_bml*)

Apply a cosine length dependent potential ($\cos(\tilde{\mathbf{r}}_x)$) where \mathbf{r}_x is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2}\lambda(S \sin(\tilde{\mathbf{r}}_x) + \sin(\tilde{\mathbf{r}}_x)S)$. $\tilde{\mathbf{r}}_x = 2\pi(\mathbf{r}/l_x) - \pi$. The symmetrization is done in order to preserve the Hermiticity of H. Units can be transformed by using the parameter λ .

Note

If the Hamiltonian is already in the prg_orthogonalized form, then parameter over_bml can be omitted.

Parameters

<i>direction</i>	Direction of the potential gradient (x,y or z).
<i>lx</i>	Lenght of the box in x direction.
<i>coordinate</i>	Coordinates of the system (\mathbf{r}).
<i>lambda</i>	Constant to premultiply the perturbation (λ).
<i>prt_bml</i>	Perturbation in bml format ($H^{(1)}$).
<i>threshold</i>	Threshold value for bml format matrices.
<i>norbi</i>	Number of orbitals for each atomic site.
<i>verbose</i>	Different levels of verbosity.
<i>over_bml</i>	It has to be present for a nonorthogonal representation (S).

Definition at line 591 of file prg_response_mod.F90.

9.23.2.9 subroutine, public prg_response_mod::prg_pert_from_file (type(bml_matrix_t), intent(inout) *prt_bml*, integer *norb*)

Read perturbation from file.

Todo Add read perturbation from file

Definition at line 223 of file prg_response_mod.F90.

9.23.2.10 subroutine, public prg_response_mod::prg_pert_sin_pot (character *direction*, real(dp) *lx*, real(dp), dimension(:,:), intent(in) *coordinate*, real(dp) *lambda*, type(bml_matrix_t), intent(inout) *prt_bml*, real(dp) *threshold*, integer, dimension(:), intent(in) *spindex*, integer, dimension(:), intent(in) *norbi*, integer, intent(in) *verbose*, type(bml_matrix_t), intent(in), optional *over_bml*)

Apply a sinusoidal length dependent potential ($\sin(\tilde{\mathbf{r}}_x)$) where \mathbf{r}_x is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2}\lambda(S \sin(\tilde{\mathbf{r}}_x) + \sin(\tilde{\mathbf{r}}_x)S)$. $\tilde{\mathbf{r}}_x = 2\pi(\mathbf{r}/l_x) - \pi$. The symmetrization is done in order to preserve the Hermiticity of H. Units can be transformed by using the parameter λ .

Note

If the Hamiltonian is already in the prg_orthogonalized form, then parameter over_bml can be omitted.

Parameters

<i>direction</i>	Direction of the potential gradient (x,y or z).
<i>lx</i>	Length of the box in x direction.
<i>coordinate</i>	Coordinates of the system (<i>r</i>).
<i>lambda</i>	Constant to premultiply the perturbation (λ).
<i>prt_bml</i>	Perturbation in bml format ($H^{(1)}$).
<i>threshold</i>	Threshold value for bml format matrices.
<i>norbi</i>	Number of orbitals for each atomic site.
<i>verbose</i>	Different levels of verbosity.
<i>over_bml</i>	It has to be present for a nonorthogonal representation (<i>S</i>).

Definition at line 523 of file prg_response_mod.F90.

9.23.2.11 subroutine, public prg_response_mod::prg_project_response (type(bml_matrix_t), intent(inout) *rsp_bml*, type(bml_matrix_t), intent(in) *over_bml*, integer, dimension(:), intent(in) *spindex*, integer, dimension(:), intent(in) *norbi*, real(dp), dimension(:,:), intent(in) *coordinates*, real(dp), dimension(:), intent(inout), allocatable *rspfunc*, integer, intent(in) *verbose*)

Project the response onto atomic positions. First order response to the perturbation ($\rho^{(1)}$) projected onto the atomic position. Basically: $rsp(i) = \sum_{\alpha \in i} \rho_{\alpha\alpha}^{(1)}$, where orbital α belong to atom i .

Parameters

<i>rsp_bml</i>	First order response density matrix.
<i>spindex</i>	It gives the species index of a particular atom.
<i>norbi</i>	Number of orbitals of species i.
<i>coordinates</i>	Atomic coordinates.
<i>rspfunc</i>	Response function at atomic positions.
<i>verbose</i>	Different levels of verbosity.

Definition at line 795 of file prg_response_mod.F90.

9.23.2.12 subroutine, public prg_response_mod::prg_write_dipole_tcl (real(dp), dimension(3), intent(in) *dipoleMoment*, character(*), intent(in) *file*, real(dp), intent(in) *factor*, integer *verbose*)

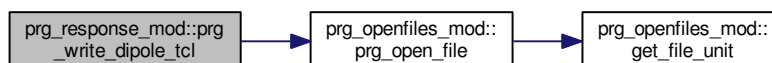
To visualize a dipole moment using VMD. This will prg_generate a .tcl script that could be run using VMD To visualize with VMD: \$ vmd -e dipole.tcl.

Parameters

<i>dipoleMoment</i>	Dipole moment vector.
<i>file</i>	PDB/XYZ file to load for visualization.
<i>factor</i>	Arbitrary scale for visualization.
<i>verbose</i>	To give different verbosity levels.

Definition at line 158 of file prg_response_mod.F90.

Here is the call graph for this function:



9.23.3 Variable Documentation

9.23.3.1 integer, parameter prg_response_mod::dp = kind(1.0d0) [private]

Definition at line 18 of file prg_response_mod.F90.

9.23.3.2 real(dp), parameter prg_response_mod::pi = 3.14159265358979323846264338327950_dp [private]

Definition at line 19 of file prg_response_mod.F90.

9.24 prg_sp2_fermi_mod Module Reference

The SP2 Fermi module.

Functions/Subroutines

- subroutine, public [prg_sp2_fermi_init](#) (h_bml, nsteps, nocc, tscale, threshold, occErrLimit, traceLimit, x_bml, mu, beta, h1, hN, sgnlist)
Truncated SP2 prg_initialization.
- subroutine, public [prg_sp2_fermi_init_norecs](#) (h_bml, nsteps, nocc, tscale, threshold, occErrLimit, traceLimit, x_bml, mu, beta, h1, hN, sgnlist, verbose)
Truncated SP2 prg_initialization. This routine also gives back the Number of SP2 recursive steps that gets a Pseudo-Fermi distribution with a temperature close to the target temperature which is entered using parameter beta = (1/KbT).
- subroutine, public [prg_sp2_fermi](#) (h_bml, osteps, nsteps, nocc, mu, beta, h1, hN, sgnlist, threshold, eps, traceLimit, x_bml)
Calculate Truncated SP2.
- subroutine, public [prg_sp2_entropy_function](#) (mu, h1, hN, nsteps, sgnlist, GG, ee)
Calculate SP2 entropy function using gaussian quadrature. Note that GG and ee are allocated and returned from this routine.
- real(dp) function, public [sp2_entropy_ts](#) (D0_bml, GG, ee)
Test SP2 entropy. Get the entropy contribution TS to the total free energy.
- real(dp) function, public [sp2_inverse](#) (f, mu, h1, hN, nsteps, sgnlist)
Calculate the SP2 inverse.
- real(dp) function [absmaxderivative](#) (func, de)
Gets the absolute maximum of the derivative of a function.

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.24.1 Detailed Description

The SP2 Fermi module.

9.24.2 Function/Subroutine Documentation

9.24.2.1 [real\(dp\) function prg_sp2_fermi_mod::absmaxderivative \(real\(dp\), dimension\(:\), intent\(in\) func, real\(dp\), intent\(in\) de \) \[private\]](#)

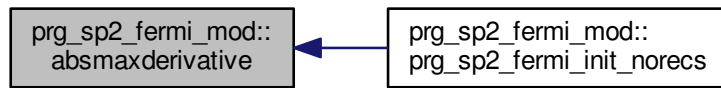
Gets the absolute maximum of the derivative of a function.

Parameters

<i>func.</i>	
<i>de</i>	Energy step.

Definition at line 615 of file prg_sp2_fermi_mod.F90.

Here is the caller graph for this function:



9.24.2.2 subroutine, public `prg_sp2_fermi_mod::prg_sp2_entropy_function` (`real(dp)`, `intent(in)` *mu*, `real(dp)`, `intent(in)` *h1*, `real(dp)`, `intent(in)` *hN*, `integer`, `intent(in)` *nsteps*, `integer`, `dimension(:)`, `intent(in)` *sgnlist*, `real(dp)`, `dimension(:)`, `intent(inout)`, allocatable *GG*, `real(dp)`, `dimension(:)`, `intent(inout)`, allocatable *ee*)

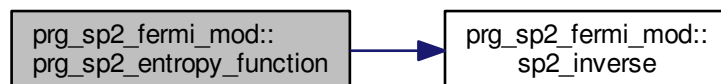
Calculate SP2 entropy function using gaussian quadrature. Note that GG and ee are allocated and returned from this routine.

Parameters

<i>mu</i>	Shifted chemical potential
<i>h1</i>	Minimum scaled Gershgorin bound
<i>hN</i>	Maximum scaled Gershgorin bound
<i>nsteps</i>	Number of SP2 steps
<i>sgnlist</i>	SP2 sequence
<i>GG</i>	Entropy function
<i>ee</i>	1D mesh

Definition at line 480 of file `prg_sp2_fermi_mod.F90`.

Here is the call graph for this function:



9.24.2.3 subroutine, public `prg_sp2_fermi_mod::prg_sp2_fermi` (`type(bml_matrix_t)`, `intent(in)` *h_bml*, `integer`, `intent(in)` *osteps*, `integer`, `intent(in)` *nsteps*, `real(dp)`, `intent(in)` *nocc*, `real(dp)`, `intent(inout)` *mu*, `real(dp)`, `intent(inout)` *beta*, `real(dp)`, `intent(inout)` *h1*, `real(dp)`, `intent(inout)` *hN*, `integer`, `dimension(:)`, `intent(in)` *sgnlist*, `real(dp)`, `intent(in)` *threshold*, `real(dp)`, `intent(in)` *eps*, `real(dp)`, `intent(in)` *traceLimit*, `type(bml_matrix_t)`, `intent(inout)` *x_bml*)

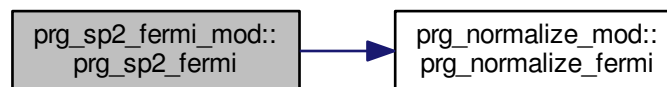
Calculate Truncated SP2.

Parameters

<i>h_bml</i>	Hamiltonian matrix
<i>osteps</i>	Outer loop steps
<i>nsteps</i>	Number of sequence branches
<i>nocc</i>	Number of occupation states
<i>mu</i>	Shifted chemical potential
<i>beta</i>	Inverse temperature
<i>h1</i>	Minimum scaled Gershgorin bound.
<i>hN</i>	Maximum scaled Gershgorin bound.
<i>sgnlist</i>	SP2 sequence
<i>threshold</i>	Threshold for multiplies
<i>eps</i>	Occupation error limit
<i>traceLimit</i>	Trace limit
<i>x_bml</i>	Output density matrix

Definition at line 387 of file `prg_sp2_fermi_mod.F90`.

Here is the call graph for this function:



9.24.2.4 subroutine, public `prg_sp2_fermi_mod::prg_sp2_fermi_init` (`type(bml_matrix_t)`, `intent(in) h_bml`, `integer`, `intent(in) nsteps`, `real(dp)`, `intent(in) nocc`, `real(dp)`, `intent(in) tscale`, `real(dp)`, `intent(in) threshold`, `real(dp)`, `intent(in) occErrLimit`, `real(dp)`, `intent(in) traceLimit`, `type(bml_matrix_t)`, `intent(inout) x_bml`, `real(dp)`, `intent(inout) mu`, `real(dp)`, `intent(inout) beta`, `real(dp)`, `intent(inout) h1`, `real(dp)`, `intent(inout) hN`, `integer`, `dimension(:)`, `intent(inout) sgnlist`)

Truncated SP2 `prg_initialization`.

Parameters

<i>h_bml</i>	Input Hamiltonian matrix.
<i>nsteps</i>	Number of sp2 iterations.
<i>nocc</i>	Number of occupied states.
<i>tscale</i>	Temperature rescaling factor.
<i>threshold</i>	Threshold for multiplication.
<i>occErrLimit</i>	Occupation error limit.
<i>traceLimit</i>	Trace limit.
<i>x_bml</i>	Output <code>prg_initial</code> matrix.
<i>mu</i>	Shifted chemical potential

Parameters

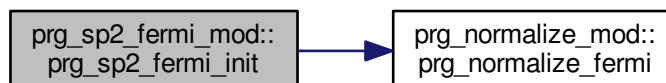
<i>beta</i>	Output inverse temperature.
<i>h1</i>	Output temperature-scaled minimum gershgorin bound.
<i>hN</i>	Output temperature-scaled maximum gershgorin bound.
<i>sgnlist</i>	SP2 sequence

Calculate Gershgorin bounds and rescale

Determine sequence branching first time through

Definition at line 46 of file prg_sp2_fermi_mod.F90.

Here is the call graph for this function:



9.24.2.5 subroutine, public prg_sp2_fermi_mod::prg_sp2_fermi_init_norecs (type(bml_matrix_t), intent(in) *h_bml*, integer, intent(inout) *nsteps*, real(dp), intent(in) *nocc*, real(dp), intent(in) *tscale*, real(dp), intent(in) *threshold*, real(dp), intent(in) *occErrLimit*, real(dp), intent(in) *traceLimit*, type(bml_matrix_t), intent(inout) *x_bml*, real(dp), intent(inout) *mu*, real(dp), intent(inout) *beta*, real(dp), intent(inout) *h1*, real(dp), intent(inout) *hN*, integer, dimension(:), intent(inout) *sgnlist*, integer, optional *verbose*)

Truncated SP2 prg_initialization. This routine also gives back the Number of SP2 recursive steps that gets a Pseudo-Fermi distribution with a temperature close to the target temperature which is entered using parameter *beta* = (1/KbT).

Parameters

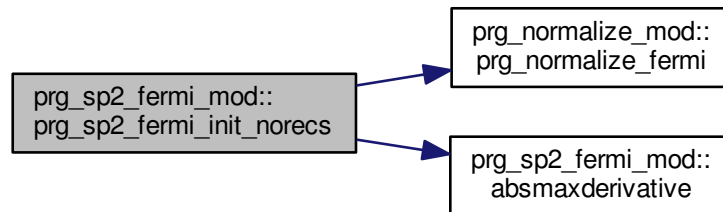
<i>h_bml</i>	Input Hamiltonian matrix.
<i>nsteps</i>	Output number of sp2 iterations.
<i>nocc</i>	Number of occupied states.
<i>tscale</i>	Temperature rescaling factor.
<i>threshold</i>	Threshold for multiplication.
<i>occErrLimit</i>	Occupation error limit.
<i>traceLimit</i>	Trace limit.
<i>x_bml</i>	Output prg_initial matrix.
<i>mu</i>	Shifted chemical potential
<i>beta</i>	Input guess and output inverse temperature.
<i>h1</i>	Output temperature-scaled minimum gershgorin bound.
<i>hN</i>	Output temperature-scaled maximum gershgorin bound.
<i>sgnlist</i>	SP2 sequence
<i>verbose</i>	Optional parameter for verbosity.

Calculate Gershgorin bounds and rescale

Determine sequence branching first time through

Definition at line 201 of file `prg_sp2_fermi_mod.F90`.

Here is the call graph for this function:



9.24.2.6 `real(dp)` function, public `prg_sp2_fermi_mod::sp2_entropy_ts` (`type(bml_matrix_t)`, `intent(in)` `D0_bml`, `real(dp)`, `dimension(*)`, `intent(in)` `GG`, `real(dp)`, `dimension(*)`, `intent(in)` `ee`)

Test SP2 entropy. Get the entropy contribution TS to the total free energy.

Parameters

<i>D0_bml</i>	BML matrix
<i>GG</i>	Entropy function
<i>ee</i>	1D mesh
<i>TS</i>	Energy contribution

Definition at line 538 of file `prg_sp2_fermi_mod.F90`.

9.24.2.7 `real(dp)` function, public `prg_sp2_fermi_mod::sp2_inverse` (`real(dp)`, `intent(in)` `f`, `real(dp)`, `intent(in)` `mu`, `real(dp)`, `intent(in)` `h1`, `real(dp)`, `intent(in)` `hN`, `integer`, `intent(in)` `nsteps`, `integer`, `dimension(:)`, `intent(in)` `sgnlist`)

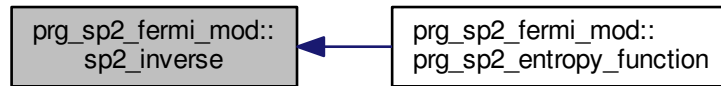
Calculate the SP2 inverse.

Parameters

<i>f</i>	Occupation factor
<i>mu</i>	Shifted chemical potential
<i>h1</i>	Minimum scaled Gershgorin bound
<i>hN</i>	Maximum scaled Gershgorin bound
<i>nsteps</i>	Numbers of SP2 iterations
<i>sgnlist</i>	SP2 sequence
<i>ee</i>	Energy value

Definition at line 590 of file prg_sp2_fermi_mod.F90.

Here is the caller graph for this function:



9.24.3 Variable Documentation

9.24.3.1 integer, parameter `prg_sp2_fermi_mod::dp = kind(1.0d0)` [private]

Definition at line 19 of file prg_sp2_fermi_mod.F90.

9.25 prg_sp2_mod Module Reference

The SP2 module.

Functions/Subroutines

- subroutine, public [prg_sp2_basic](#) (`h_bml`, `rho_bml`, `threshold`, `bndfil`, `minsp2iter`, `maxsp2iter`, `sp2conv`, `idemtol`, `verbose`)
Calculates the density matrix from a Hamiltonian matrix by purification. The method implemented here is the very first verion of the SP2 method.
- subroutine, public [prg_sp2_alg2](#) (`h_bml`, `rho_bml`, `threshold`, `bndfil`, `minsp2iter`, `maxsp2iter`, `sp2conv`, `idemtol`, `verbose`)
- subroutine, public [prg_sp2_alg2_genseq](#) (`h_bml`, `rho_bml`, `threshold`, `bndfil`, `minsp2iter`, `maxsp2iter`, `sp2conv`, `idemtol`, `pp`, `icount`, `vv`, `verbose`)
- subroutine, public [prg_sp2_alg2_seq](#) (`h_bml`, `rho_bml`, `threshold`, `pp`, `icount`, `vv`, `verbose`)
- subroutine, public [prg_prg_sp2_alg2_seq_inplace](#) (`rho_bml`, `threshold`, `pp`, `icount`, `vv`, `mineval`, `maxeval`, `verbose`)
- subroutine, public [prg_sp2_alg1](#) (`h_bml`, `rho_bml`, `threshold`, `bndfil`, `minsp2iter`, `maxsp2iter`, `sp2conv`, `idemtol`, `verbose`)
- subroutine, public [prg_sp2_alg1_genseq](#) (`h_bml`, `rho_bml`, `threshold`, `bndfil`, `minsp2iter`, `maxsp2iter`, `sp2conv`, `idemtol`, `pp`, `icount`, `vv`)
- subroutine, public [prg_sp2_alg1_seq](#) (`h_bml`, `rho_bml`, `threshold`, `pp`, `icount`, `vv`)
- subroutine, public [prg_prg_sp2_alg1_seq_inplace](#) (`rho_bml`, `threshold`, `pp`, `icount`, `vv`, `mineval`, `maxeval`)
- subroutine, public [prg_sp2_submatrix](#) (`ham_bml`, `rho_bml`, `threshold`, `pp`, `icount`, `vv`, `mineval`, `maxeval`, `core_↔_size`)
Perform SP2 algorithm using sequence and calculate norm for a submatrix.
- subroutine, public [prg_sp2_submatrix_inplace](#) (`rho_bml`, `threshold`, `pp`, `icount`, `vv`, `mineval`, `maxeval`, `core_↔_size`)

Variables

- integer, parameter `dp = kind(1.0d0)`

9.25.1 Detailed Description

The SP2 module.

Author

S. Mniszewski (smn@lanl.gov)

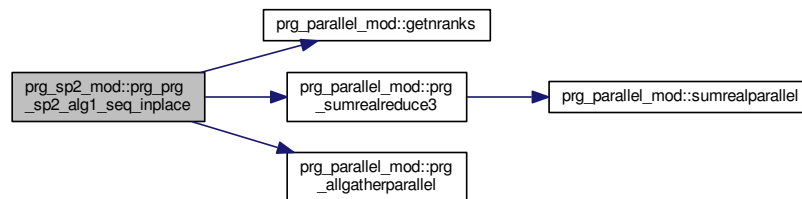
This subroutine implements Niklasson's SP2 density matrix purification algorithm.

9.25.2 Function/Subroutine Documentation

9.25.2.1 `subroutine, public prg_sp2_mod::prg_prg_sp2_alg1_seq_inplace (type(bml_matrix_t), intent(inout) rho_bml, real(dp), intent(in) threshold, integer, dimension(:), intent(inout) pp, integer, intent(inout) icount, real(dp), dimension(:), intent(inout) vv, real(dp), intent(in) mineval, real(dp), intent(in) maxeval)`

Definition at line 971 of file `prg_sp2_mod.F90`.

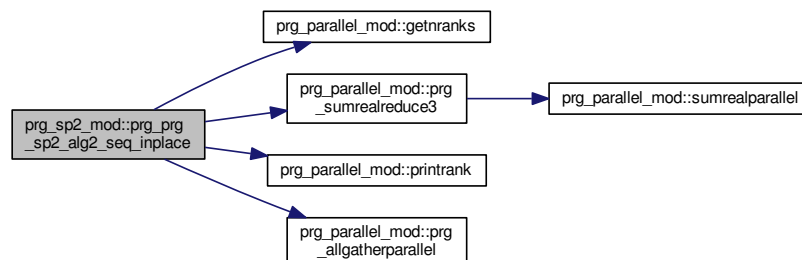
Here is the call graph for this function:



9.25.2.2 `subroutine, public prg_sp2_mod::prg_prg_sp2_alg2_seq_inplace (type(bml_matrix_t), intent(inout) rho_bml, real(dp), intent(in) threshold, integer, dimension(:), intent(inout) pp, integer, intent(inout) icount, real(dp), dimension(:), intent(inout) vv, real(dp), intent(in), optional mineval, real(dp), intent(in), optional maxeval, integer, intent(in), optional verbose)`

Definition at line 528 of file `prg_sp2_mod.F90`.

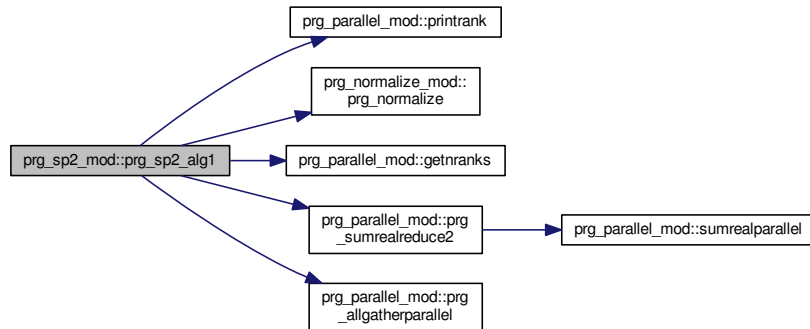
Here is the call graph for this function:



9.25.2.3 subroutine, public prg_sp2_mod::prg_sp2_alg1 (type(bml_matrix_t), intent(in) *h_bml*, type(bml_matrix_t), intent(inout) *rho_bml*, real(dp), intent(in) *threshold*, real(dp), intent(in) *bndfil*, integer, intent(in) *minsp2iter*, integer, intent(in) *maxsp2iter*, character(len=*), intent(in) *sp2conv*, real(dp), intent(in) *idemtol*, integer, intent(in), optional *verbose*)

Definition at line 624 of file prg_sp2_mod.F90.

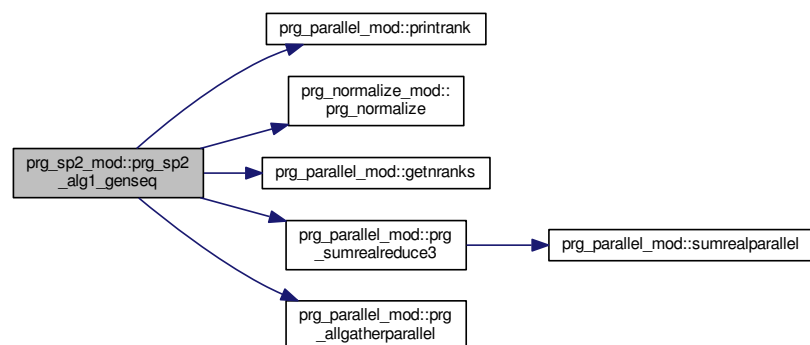
Here is the call graph for this function:



9.25.2.4 subroutine, public prg_sp2_mod::prg_sp2_alg1_genseq (type(bml_matrix_t), intent(in) *h_bml*, type(bml_matrix_t), intent(inout) *rho_bml*, real(dp), intent(in) *threshold*, real(dp), intent(in) *bndfil*, integer, intent(in) *minsp2iter*, integer, intent(in) *maxsp2iter*, character(len=*), intent(in) *sp2conv*, real(dp), intent(in) *idemtol*, integer, dimension(:), intent(inout) *pp*, integer, intent(inout) *icount*, real(dp), dimension(:), intent(inout) *vv*)

Definition at line 751 of file prg_sp2_mod.F90.

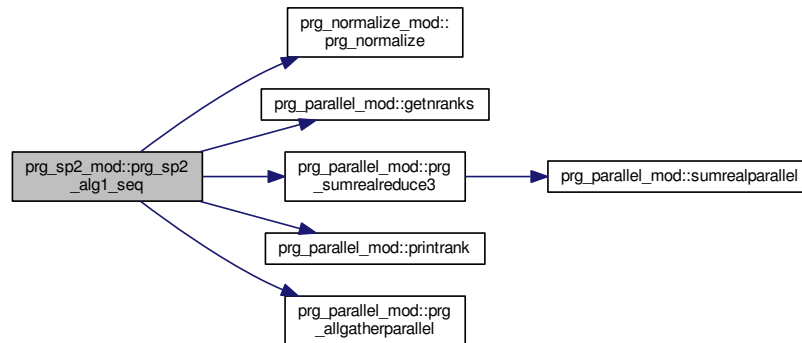
Here is the call graph for this function:



9.25.2.5 subroutine, public prg_sp2_mod::prg_sp2_alg1_seq (type(bml_matrix_t), intent(in) *h_bml*, type(bml_matrix_t), intent(inout) *rho_bml*, real(dp), intent(in) *threshold*, integer, dimension(:), intent(inout) *pp*, integer, intent(inout) *icount*, real(dp), dimension(:), intent(inout) *vv*)

Definition at line 880 of file prg_sp2_mod.F90.

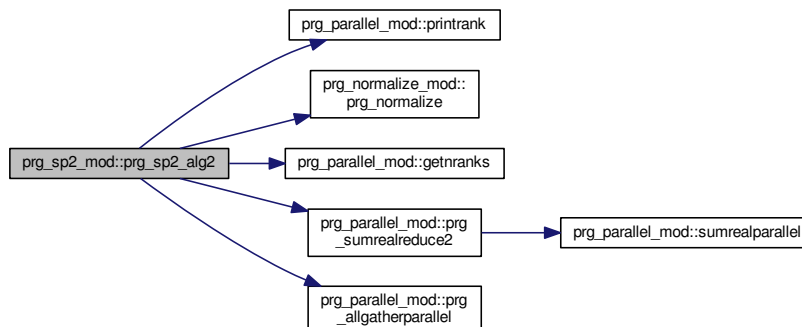
Here is the call graph for this function:



9.25.2.6 subroutine, public prg_sp2_mod::prg_sp2_alg2 (type(bml_matrix_t), intent(in) *h_bml*, type(bml_matrix_t), intent(inout) *rho_bml*, real(dp), intent(in) *threshold*, real(dp), intent(in) *bndfil*, integer, intent(in) *minsp2iter*, integer, intent(in) *maxsp2iter*, character(len=*), intent(in) *sp2conv*, real(dp), intent(in) *idemtol*, integer, intent(in), optional *verbose*)

Definition at line 153 of file prg_sp2_mod.F90.

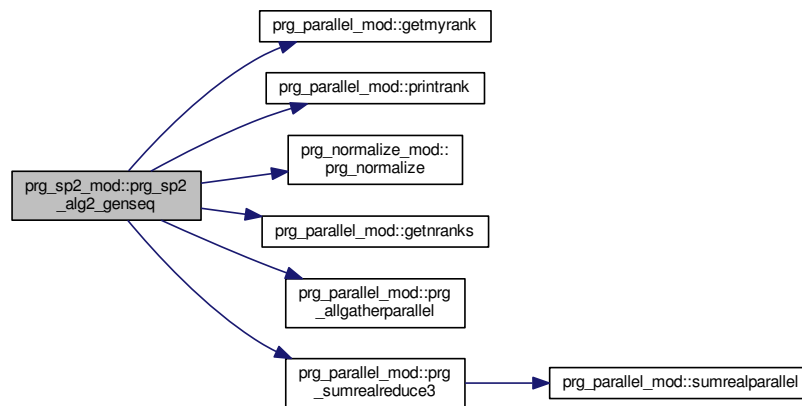
Here is the call graph for this function:



9.25.2.7 subroutine, public prg_sp2_mod::prg_sp2_alg2_genseq (type(bml_matrix_t), intent(in) *h_bml*, type(bml_matrix_t), intent(inout) *rho_bml*, real(dp), intent(in) *threshold*, real(dp), intent(in) *bndfil*, integer, intent(in) *minsp2iter*, integer, intent(in) *maxsp2iter*, character(len=*), intent(in) *sp2conv*, real(dp), intent(in) *idemtol*, integer, dimension(:), intent(inout) *pp*, integer, intent(inout) *icount*, real(dp), dimension(:), intent(inout) *vv*, integer, intent(in), optional *verbose*)

Definition at line 285 of file prg_sp2_mod.F90.

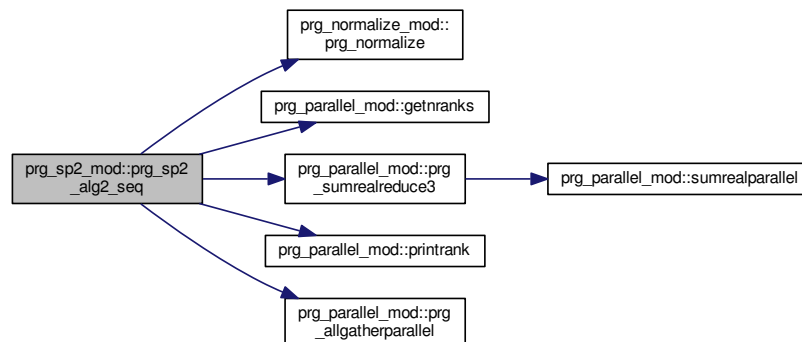
Here is the call graph for this function:



9.25.2.8 subroutine, public prg_sp2_mod::prg_sp2_alg2_seq (type(bml_matrix_t), intent(in) *h_bml*, type(bml_matrix_t), intent(inout) *rho_bml*, real(dp), intent(in) *threshold*, integer, dimension(:), intent(inout) *pp*, integer, intent(inout) *icount*, real(dp), dimension(:), intent(inout) *vv*, integer, intent(in), optional *verbose*)

Definition at line 432 of file prg_sp2_mod.F90.

Here is the call graph for this function:



9.25.2.9 subroutine, public prg_sp2_mod::prg_sp2_basic (type(bml_matrix_t), intent(in) *h_bml*, type(bml_matrix_t), intent(inout) *rho_bml*, real(dp) *threshold*, real(dp) *bndfil*, integer *minsp2iter*, integer *maxsp2iter*, character(len=*), intent(in) *sp2conv*, real(dp) *idemtol*, integer *verbose*)

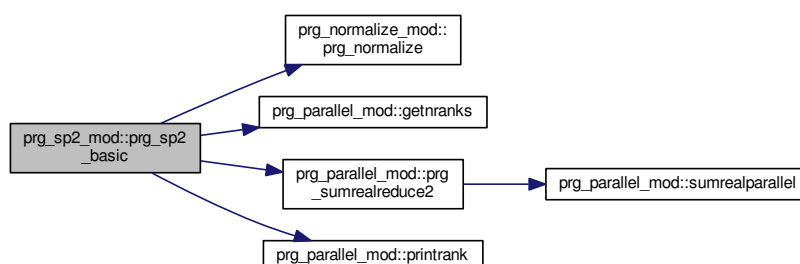
Calculates the density matrix from a Hamiltonian matrix by purification. The method implemented here is the very first version of the SP2 method.

Parameters

<i>h_bml</i>	Input Hamiltonian matrix
<i>rho_bml</i>	Output density matrix
<i>threshold</i>	Threshold for sparse matrix algebra
<i>bndfil</i>	Bond
<i>minsp2iter</i>	Minimum sp2 iterations
<i>maxsp2iter</i>	Maximum SP2 iterations
<i>sp2conv</i>	Convergence type
<i>idemtol</i>	Idempotency tolerance
<i>verbose</i>	A verbosity level

Definition at line 52 of file prg_sp2_mod.F90.

Here is the call graph for this function:



9.25.2.10 subroutine, public prg_sp2_mod::prg_sp2_submatrix (type(bml_matrix_t), intent(in) *ham_bml*, type(bml_matrix_t), intent(inout) *rho_bml*, real(dp), intent(in) *threshold*, integer, dimension(:), intent(in) *pp*, integer, intent(in) *icount*, real(dp), dimension(:), intent(inout) *vv*, real(dp), intent(in) *mineval*, real(dp), intent(in) *maxeval*, integer, intent(in) *core_size*)

Perform SP2 algorithm using sequence and calculate norm for a submatrix.

Parameters

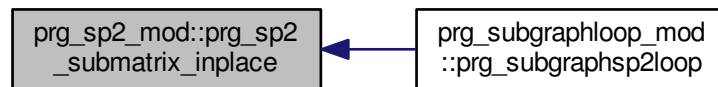
<i>rho_bml</i>	Input Hamiltonian/Output density matrix
<i>threshold</i>	Threshold for sparse matrix algebra
<i>pp</i>	Vector containing sequence of 0s and 1s
<i>icount</i>	Sequence count
<i>vv</i>	Vector of sum of squares per iteration
<i>mineval</i>	Min value used for normalization (optional)
<i>maxeval</i>	Max value used for normalization (optional)
<i>core_size</i>	Number of core rows

Definition at line 1061 of file prg_sp2_mod.F90.

9.25.2.11 subroutine, public prg_sp2_mod::prg_sp2_submatrix_inplace (type(bml_matrix_t), intent(inout) *rho_bml*, real(dp), intent(in) *threshold*, integer, dimension(:), intent(inout) *pp*, integer, intent(inout) *icount*, real(dp), dimension(:), intent(inout) *vv*, real(dp), intent(in) *mineval*, real(dp), intent(in) *maxeval*, integer, intent(in) *core_size*)

Definition at line 1130 of file prg_sp2_mod.F90.

Here is the caller graph for this function:



9.25.3 Variable Documentation

9.25.3.1 integer, parameter prg_sp2_mod::dp = kind(1.0d0) [private]

Definition at line 21 of file prg_sp2_mod.F90.

9.26 prg_sp2parser_mod Module Reference

SP2 parser.

Data Types

- type [sp2data_type](#)
General SP2 solver type.

Functions/Subroutines

- subroutine, public [prg_parse_sp2](#) (sp2data, filename)
The parser for SP2 solver.

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.26.1 Detailed Description

SP2 parser.

This module is used to parse all the input variables for the SP2 method electronic structure solver. Adding a new input keyword to the parser:

- If the variable is real, we have to increase nkey_re.
- Add the keyword (character type) in the keyvector_re vector.
- Add a default value (real type) in the valvector_re.
- Define a new variable and pass the value through valvector_re(num) where num is the position of the new keyword in the vector.

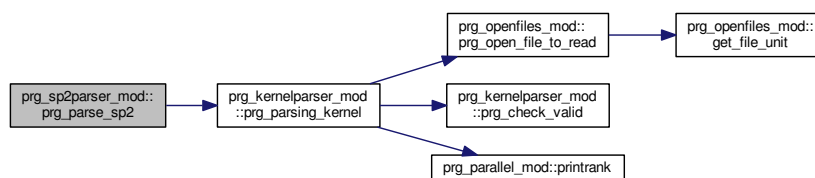
9.26.2 Function/Subroutine Documentation

9.26.2.1 subroutine, public prg_sp2parser_mod::prg_parse_sp2 (type(sp2data_type), intent(inout) sp2data, character(len=*) filename)

The parser for SP2 solver.

Definition at line 50 of file prg_sp2parser_mod.F90.

Here is the call graph for this function:



9.26.3 Variable Documentation

9.26.3.1 integer, parameter prg_sp2parser_mod::dp = kind(1.0d0) [private]

Definition at line 22 of file prg_sp2parser_mod.F90.

9.27 prg_subgraphloop_mod Module Reference

The subgraphloop module.

Functions/Subroutines

- subroutine, public [prg_subgraphsp2loop](#) (h_bml, g_bml, rho_bml, gp, threshold)
- subroutine, public [prg_collectmatrixfromparts](#) (gp, rho_bml)
Collect distributed parts into same matrix.
- subroutine, public [prg_balanceparts](#) (gp)
- subroutine, public [prg_partordering](#) (gp)
Set row ordering bases on parts.
- subroutine, public [prg_getgrouppartitionhalosfromgraph](#) (gp, g_bml, hnode, djflag)
Get core+halo indeces for all partitions only using the graph.
- subroutine, public [prg_getpartitionhalosfromgraph](#) (gp, g_bml, djflag)
Get core+halo indeces for all partitions only using the graph.

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.27.1 Detailed Description

The subgraphloop module.

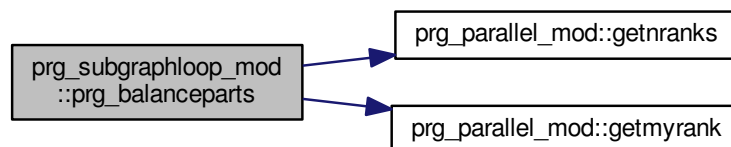
9.27.2 Function/Subroutine Documentation

9.27.2.1 subroutine, public `prg_subgraphloop_mod::prg_balanceparts (type (graph_partitioning_t), intent(inout) gp)`

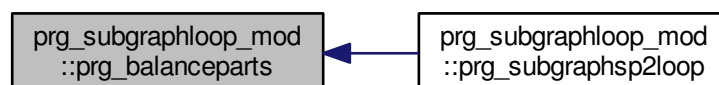
Renumber parts Handle unbalanced numbers of parts.

Definition at line 165 of file `prg_subgraphloop_mod.F90`.

Here is the call graph for this function:



Here is the caller graph for this function:



9.27.2.2 subroutine, public prg_subgraphloop_mod::prg_collectmatrixfromparts (type (graph_partitioning_t), intent(inout) *gp*, type (bml_matrix_t), intent(inout) *rho_bml*)

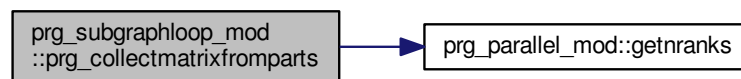
Collect distributed parts into same matrix.

Parameters

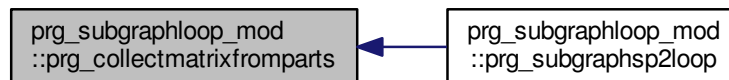
<i>gp</i>	Graph partitioning
<i>rho_bml</i>	Matrix to be collected into

Definition at line 133 of file prg_subgraphloop_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.27.2.3 subroutine, public prg_subgraphloop_mod::prg_getgrouppartitionhalosfromgraph (type (graph_partitioning_t), intent(inout) *gp*, type (bml_matrix_t), intent(in) *g_bml*, integer, dimension(*), intent(in) *hnode*, logical, intent(in) *djflag*)

Get core+halo indeces for all partitions only using the graph.

Parameters

<i>gp</i>	Graph partitioning
<i>g_bml</i>	Graph
<i>hnode</i>	Group start indeces
<i>djflg</i>	Double jump flag (true/false)

Determine halo elements for each subgraph

Definition at line 292 of file prg_subgraphloop_mod.F90.

9.27.2.4 subroutine, public prg_subgraphloop_mod::prg_getpartitionhalosfromgraph (type (graph_partitioning_t), intent(inout) *gp*, type (bml_matrix_t), intent(in) *g_bml*, logical, intent(in) *djflag*)

Get core+halo indeces for all partitions only using the graph.

Parameters

<i>gp</i>	Graph partitioning
<i>g_bml</i>	Graph
<i>djflg</i>	Double jump flag (true/false)

Determine halo elements for each subgraph

Definition at line 337 of file prg_subgraphloop_mod.F90.

9.27.2.5 subroutine, public prg_subgraphloop_mod::prg_partordering (type (graph_partitioning_t), intent(inout) *gp*)

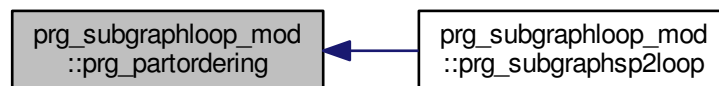
Set row ordering bases on parts.

Parameters

<i>gp</i>	Graph partitioning
-----------	--------------------

Definition at line 263 of file prg_subgraphloop_mod.F90.

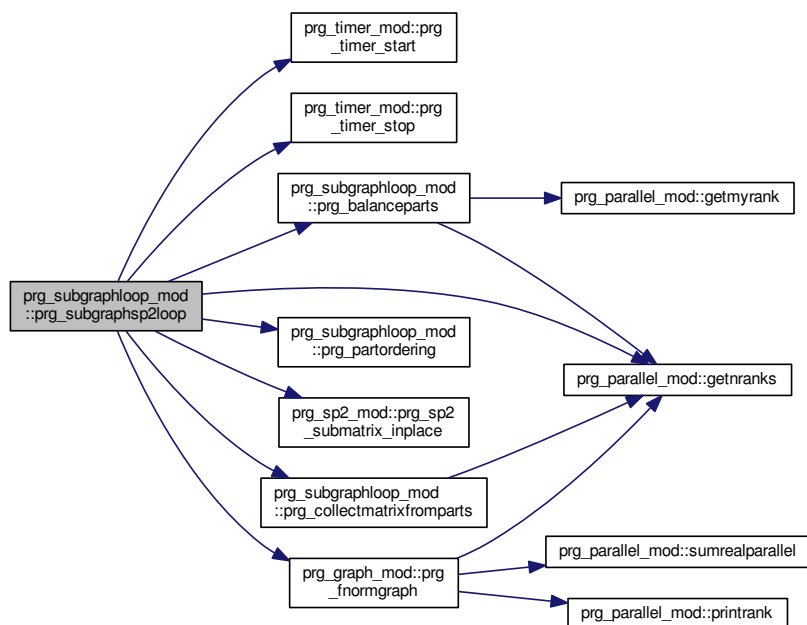
Here is the caller graph for this function:



9.27.2.6 subroutine, public prg_subgraphloop_mod::prg_subgraphsp2loop (type (bml_matrix_t), intent(in) *h_bml*, type (bml_matrix_t), intent(in) *g_bml*, type (bml_matrix_t), intent(inout) *rho_bml*, type (graph_partitioning_t), intent(inout) *gp*, real(dp), intent(in) *threshold*)

Definition at line 37 of file prg_subgraphloop_mod.F90.

Here is the call graph for this function:



9.27.3 Variable Documentation

9.27.3.1 integer, parameter `prg_subgraphloop_mod::dp = kind(1.0d0)` [private]

Definition at line 18 of file `prg_subgraphloop_mod.F90`.

9.28 `prg_syrotation_mod` Module Reference

A module to rotate the coordinates of a sybsystem in chemical systems.

Data Types

- type `rotation_type`
Rotation type.

Functions/Subroutines

- subroutine, public `prg_parse_rotation` (rot, filename)
The parser for rotation.
- subroutine, public `prg_rotate` (rot, r, verbose)
Rotation routine.

Variables

- integer, parameter `dp = kind(1.0d0)`

9.28.1 Detailed Description

A module to rotate the coordinates of a sybsystem in chemical systems.

It works by specifying two orientations and a rotation point.

Author

C. F. A. Negre (cnegre@lanl.gov)

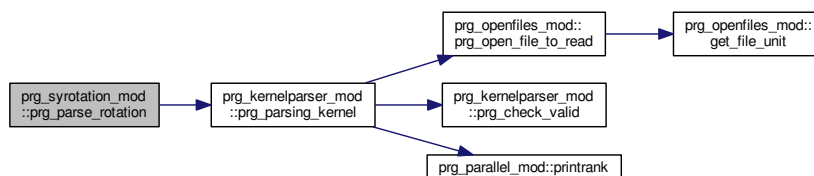
9.28.2 Function/Subroutine Documentation

9.28.2.1 subroutine, public `prg_syrotation_mod::prg_parse_rotation (type(rotation_type), intent(inout) rot, character(len=*) filename)`

The parser for rotation.

Definition at line 49 of file `prg_syrotation_mod.F90`.

Here is the call graph for this function:



9.28.2.2 subroutine, public `prg_syrotation_mod::prg_rotate (type(rotation_type), intent(in) rot, real(dp), dimension(:,:), intent(inout) r, integer, intent(in) verbose)`

Rotation routine.

It works by indicating the orientations ($v1$ and $v1$) and a rotation center. The orientation can be passed either directly by setting $v1$ and $v2$ or by indicating two points $pQ1$ and $pQ2$. Orientation can also be specified with an atom position if $patom1$ and $patom2$ indices are not zero this atoms are used to determine the initial and final orientation.

Parameters

<i>rot</i>	Rotation type
<i>r</i>	Coordinates to be rotated
<i>verbose</i>	Verbosity level

Example:

```
rot%patom1 = 4
rot%patom2 = 0
rot%catom2 = 6
rot%v2 = 0.0 ; rot%v2(1) = 1
call prg_rotate(rot,r)
```

The latter will orient the system such that atom 4 points to the (1,0,0) direction.

Definition at line 142 of file prg_syrotation_mod.F90.

9.28.3 Variable Documentation

9.28.3.1 integer, parameter prg_syrotation_mod::dp = kind(1.0d0) [private]

Definition at line 14 of file prg_syrotation_mod.F90.

9.29 prg_system_mod Module Reference

A module to read and handle chemical systems.

Data Types

- type [estruct_type](#)
Electronic structure type.
- type [system_type](#)
System type.

Functions/Subroutines

- subroutine, public [prg_get_nameandext](#) (fullfilename, filename, ext)
Get the name and extension of a file.
- subroutine, public [prg_parse_system](#) (system, filename, extin)
The parser for the chemical system.
- subroutine, public [prg_write_system](#) (system, filename, extension)
Write system in .xyz, .dat or pdb file.
- subroutine, public [prg_write_trajectory](#) (system, iter, each, prg_deltat, filename, extension)
Write trajectory in .xyz, .dat or pdb file.
- subroutine, public [prg_write_trajectoryandproperty](#) (system, iter, each, prg_deltat, scalarprop, filename, extension)
Write trajectory and atomic properties. Only pdb file.
- subroutine, public [prg_make_random_system](#) (system, nats, seed, lx, ly, lz)
Make random Xx system.
- subroutine [prg_parameters_to_vectors](#) (abc_angles, lattice_vector)
Transforms the lattice parameters into lattice vectors.
- subroutine [prg_vectors_to_parameters](#) (lattice_vector, abc_angles)

- Transforms the lattice vectors into lattice parameters.*
- subroutine, public [prg_get_origin](#) (coords, origin)
 - Get the origin of the coordinates.*
- subroutine, public [prg_get_distancematrix](#) (coords, dmat)
 - Get the distance matrix.*
- subroutine, public [prg_translateandfoldtobox](#) (coords, lattice_vectors, origin, verbose)
 - Translate and fold to box.*
- subroutine, public [prg_centeratbox](#) (coords, lattice_vectors, verbose)
 - Translate geometric center to the center of the box.*
- subroutine, public [prg_wraparound](#) (coords, lattice_vectors, index, verbose)
 - Wrap around atom i using pbc.*
- subroutine, public [prg_translatetogeomcandfoldtobox](#) (coords, lattice_vectors, origin)
 - Translate to geometric center.*
- subroutine, public [prg_replicate](#) (coords, symbols, lattice_vectors, nx, ny, nz)
 - Extend/replicate system along lattice vectors.*
- subroutine, public [prg_get_recip_vects](#) (lattice_vectors, recip_vectors, volr, volk)
 - Get the volume of the cell and the reciprocal vectors: This subroutine computes:*
- subroutine, public [prg_get_dihedral](#) (coords, id1, id2, id3, id4, dihedral)
 - Get the dihedral angle given four atomic positions.*
- subroutine, public [prg_get_covgraph](#) (sy, nnStructMindist, nnStruct, nrrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)
 - Get the covalency graph in bml format.*
- subroutine [prg_get_covgraph_int](#) (sy, nnStructMindist, nnStruct, nrrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)
 - Get the covanlency graph.*
- subroutine, public [prg_get_subsystem](#) (sy, lsize, indices, sbsy, verbose)
 - Get a subsystem out of the total system.*
- subroutine, public [prg_destroy_subsystems](#) (sbsy, verbose)
 - Destroy allocated subsystem.*
- subroutine, public [prg_molpartition](#) (sy, npart, nnStructMindist, nnStruct, nrrnnstruct, hetatm, gp, verbose)
 - Partition by molecule.*
- subroutine, public [prg_get_partial_atomgraph](#) (rho_bml, hindex, gch_bml, threshold, verbose)
 - Get partial subgraph based on the Density matrix.*
- subroutine, public [prg_collect_graph_p](#) (rho_bml, nc, nats, hindex, chindex, graph_p, threshold, mdimin, verbose)
 - Collect the small graph to build the full graph.*
- subroutine, public [prg_merge_graph](#) (graph_p, graph_h)
 - Get partial subgraph based on the Density matrix.*
- subroutine, public [prg_merge_graph_adj](#) (graph_p, graph_h, xadj, adjncy)
 - Get partial subgraph based on the Density matrix.*
- subroutine, public [prg_adj2bml](#) (xadj, adjncy, bml_type, g_bml)
 - prg_adj2bml*
- subroutine, public [prg_graph2bml](#) (graph, bml_type, g_bml)
 - Graph2bml.*
- subroutine, public [prg_graph2vector](#) (graph, vector, maxnz)
 - Vectorize graph.*
- subroutine, public [prg_vector2graph](#) (vector, graph, maxnz)
 - Back to graph.*
- subroutine, public [prg_sortadj](#) (xadj, adjncy)
 - Sort adj NOTE: this might not be needed anymore since the bml_get_adj routine is sorting the values.*

Variables

- integer, parameter `dp` = kind(1.0d0)

9.29.1 Detailed Description

A module to read and handle chemical systems.

This module will be used to build and handle a molecular system.

Author

C. F. A. Negre (cnegre@lanl.gov)

9.29.2 Function/Subroutine Documentation

9.29.2.1 subroutine, public `prg_system_mod::prg_adj2bml` (integer, dimension(:), intent(in) *xadj*, integer, dimension(:), intent(in) *adjncy*, character(20), intent(in) *bml_type*, type(bml_matrix_t), intent(inout) *g_bml*)

`prg_adj2bml`

Parameters

<i>xadj</i>	CSR start values for the adjacency matrix.
<i>adjncy</i>	CSR positions of adjacency matrix.
<i>bml_type</i>	bml format.
<i>g_bml</i>	graph in bml format.

Definition at line 2299 of file `prg_system_mod.F90`.

9.29.2.2 subroutine, public `prg_system_mod::prg_centeratbox` (real(`dp`), dimension(:,:), intent(inout), allocatable *coords*, real(`dp`), dimension(:,:), intent(in) *lattice_vectors*, integer, intent(in), optional *verbose*)

Translate geometric center to the center of the box.

Parameters

<i>coords</i>	Coordinates of the system (see system_type).
<i>lattice_vectors</i>	System lattice vectors.
<i>verbose</i>	Verbosity level.

Definition at line 1293 of file `prg_system_mod.F90`.

9.29.2.3 subroutine, public prg_system_mod::prg_collect_graph_p (type(bml_matrix_t), intent(in) *rho_bml*, integer, intent(in) *nc*, integer, intent(in) *nats*, integer, dimension(:,:), intent(in) *hindex*, integer, dimension(:), intent(in) *chindex*, integer, dimension(:,:), intent(inout), allocatable *graph_p*, real(dp), intent(in) *threshold*, integer, intent(in) *mdimin*, integer, intent(in), optional *verbose*)

Collect the small graph to build the full graph.

Parameters

<i>rho_bml</i>	Density matix in bml format.
<i>nc</i>	Number of core atoms.
<i>nats</i>	Number of atoms.
<i>hindex</i>	Hindex for the small part (see haindex)
<i>chindex</i>	Core-hallo index for the small part.
<i>graph_p</i>	Graph in an "ellpack" format.
<i>threshold</i>	Threshold to buil the density based atom projected graph.
<i>verbose</i>	Verbosity level.

Definition at line 2082 of file prg_system_mod.F90.

9.29.2.4 subroutine, public prg_system_mod::prg_destroy_subsystems (type(system_type), intent(inout) *sbsy*, integer, intent(in), optional *verbose*)

Destroy allocated subsystem.

This routine will deallocate all the arrays of the structures.

Parameters

<i>sy</i>	System to de deallocated (see system_type).
-----------	--

Definition at line 1867 of file prg_system_mod.F90.

9.29.2.5 subroutine, public prg_system_mod::prg_get_covgraph (type(system_type), intent(in) *sy*, real(dp), dimension(:,:), intent(in) *nnStructMindist*, integer, dimension(:,:), intent(in) *nnStruct*, integer, dimension(:), intent(in) *nrnnstruct*, character(20), intent(in) *bml_type*, real(dp) *factor*, type(bml_matrix_t), intent(inout) *gconv_bml*, integer, intent(in) *mdimin*, integer, intent(in), optional *verbose*)

Get the covalency graph in bml format.

This is the graph composed by the covalent bonds (edges) that are determined with the VDW radius.

Parameters

<i>sy</i>	System structure (see system_type).
<i>nnStructMindist</i>	Minimun distance between atoms.
<i>nnStruct</i>	The neighbors J to I within Rcut that are all within the box.
<i>nrnnstruct</i>	Number of neighbors to I within Rcut that are all within the box.
<i>bml_type</i>	The bml type for constructing the graph.
<i>gconv_bml</i>	Covanlency graph in bml format.
<i>verbose</i>	Verbosity level.

Definition at line 1582 of file prg_system_mod.F90.

9.29.2.6 subroutine, public prg_system_mod::prg_get_covgraph_h (type(system_type), intent(in) sy, real(dp), dimension(:,:), intent(in) nnStructMindist, integer, dimension(:,:), intent(in) nnStruct, integer, dimension(:), intent(in) nrrnstruct, real(dp), intent(in) rcut, integer, dimension(:,:), intent(inout), allocatable graph_h, integer, intent(in) mdimin, integer, intent(in), optional verbose)

Get the covanlency graph.

This is the graph composed by the covalent bonds (edges) that are determined with the VDW radius.

Parameters

sy	System structure (see system_type).
nnStructMindist	Minimun distance between atoms.
nnStruct	The neighbors J to I within Rcut that are all within the box.
nrrnstruct	Number of neighbors to I within Rcut that are all within the box.
bml_type	The bml type for constructing the graph.
gconv_bml	Covanlency graph in bml format.
verbose	Verbosity level.

Definition at line 1715 of file prg_system_mod.F90.

9.29.2.7 subroutine prg_system_mod::prg_get_covgraph_int (type(system_type), intent(in) sy, real(dp), dimension(:,:), intent(in) nnStructMindist, integer, dimension(:,:), intent(in) nnStruct, integer, dimension(:), intent(in) nrrnstruct, character(20), intent(in) bml_type, real(dp) factor, type(bml_matrix_t), intent(inout) gcov_bml, integer, intent(in) mdimin, integer, intent(in), optional verbose) [private]

Definition at line 1656 of file prg_system_mod.F90.

9.29.2.8 subroutine, public prg_system_mod::prg_get_dihedral (real(dp), dimension(:,:), intent(in) coords, integer, intent(in) id1, integer, intent(in) id2, integer, intent(in) id3, integer, intent(in) id4, real(dp), intent(out) dihedral)

Get the dihedral angle given four atomic positions.

Parameters

sy	System structure
id1	Atom index 1
id2	Atom index 1
id3	Atom index 1
id4	Atom index 1
dihedral	Output dihedral angle

Definition at line 1526 of file prg_system_mod.F90.

9.29.2.9 subroutine, public prg_system_mod::prg_get_distancematrix (real(dp), dimension(:, :), intent(in) *coords*, real(dp), dimension(:, :), intent(out), allocatable *dmat*)

Get the distance matrix.

Parameters

<i>coords</i>	Coordinates of the system (see system_type).
<i>dmat</i>	Distance matrix (nats x nats).

Definition at line 1219 of file prg_system_mod.F90.

9.29.2.10 subroutine, public prg_system_mod::prg_get_nameandext (character(30), intent(in) *fullfilename*, character(30), intent(inout) *filename*, character(3), intent(inout) *ext*)

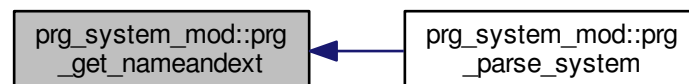
Get the name and extension of a file.

Parameters

<i>fullfilename</i>	Full filename.
<i>filename</i>	Filename of the system.
<i>extension</i>	Extension of the file.

Definition at line 206 of file prg_system_mod.F90.

Here is the caller graph for this function:



9.29.2.11 subroutine, public prg_system_mod::prg_get_origin (real(dp), dimension(:, :), intent(in) *coords*, real(dp), dimension(:, :), intent(inout), allocatable *origin*)

Get the origin of the coordinates.

Parameters

<i>coords</i>	Coordinates of the system (see system_type).
<i>origin</i>	(min(x),min(y),min(z)) set as the origin of the system.

Definition at line 1183 of file prg_system_mod.F90.

9.29.2.12 subroutine, public prg_system_mod::prg_get_partial_atomgraph (type(bml_matrix_t), intent(in) rho_bml, integer, dimension(:,:), intent(in) hindex, type(bml_matrix_t), intent(inout) gch_bml, real(dp), intent(in) threshold, integer, intent(in), optional verbose)

Get partial subgraph based on the Density matrix.

Parameters

<i>rho_bml</i>	Density matix in bml format.
<i>hindex</i>	Start and end index for every atom in the system.
<i>gch_bml</i>	Atom based graph in bml format.
<i>threshold</i>	Threshold value for constructing the graph.
<i>verbose</i>	Verbosity levels.

Definition at line 2016 of file prg_system_mod.F90.

9.29.2.13 subroutine, public prg_system_mod::prg_get_recip_vects (real(dp), dimension(:,:), intent(in) lattice_vectors, real(dp), dimension(:,:), intent(inout), allocatable recip_vectors, real(dp), intent(inout) volr, real(dp), intent(inout) volk)

Get the volume of the cell and the reciprocal vectors: This soubroutine computes:

- $b_1 = \frac{1}{V_c} a_1 \times a_2$
- $b_2 = \frac{1}{V_c} a_2 \times a_3$
- $b_3 = \frac{1}{V_c} a_3 \times a_1$
- $V_c = ||a_1 \cdot (a_2 \times a_3)||$
- $V_{BZ} = ||b_1 \cdot (b_2 \times b_3)||$

Parameters

<i>lattice_vectors</i>	Lattice vectors for the system.
<i>recip_vectors</i>	Reciprocal vectors of the system.
<i>volr</i>	Volume of the cell.
<i>volk</i>	Volume of the reciprocal cell.

Definition at line 1477 of file prg_system_mod.F90.

9.29.2.14 subroutine, public prg_system_mod::prg_get_subsystem (type(system_type), intent(in) sy, integer, intent(in) lsize, integer, dimension(:), intent(in) indices, type(system_type), intent(inout) sbsy, integer, intent(in), optional verbose)

Get a subsystem out of the total system.

This will get a subsystem from the total system guided by a partition.

Parameters

<i>sy</i>	System structure (see system_type).
<i>lsize</i>	Core+Hallo subsystem size.
<i>indices</i>	Partition indices.
<i>sbsy</i>	Subsystem to be extracted.

Definition at line 1780 of file prg_system_mod.F90.

9.29.2.15 subroutine, public prg_system_mod::prg_graph2bml (integer, dimension(:, :), intent(inout), allocatable *graph*, character(20), intent(in) *bml_type*, type(bml_matrix_t), intent(inout) *g_bml*)

Graph2bml.

Parameters

<i>graph</i>	Atom based graph in "ellpack" like format.
<i>bml_type</i>	Bml type (usually ellpack for graph storage)
<i>g_bml</i>	Graph in bml format.

Definition at line 2333 of file prg_system_mod.F90.

9.29.2.16 subroutine, public prg_system_mod::prg_graph2vector (integer, dimension(:, :), intent(inout) *graph*, integer, dimension(:), allocatable *vector*, integer *maxnz*)

Vectorize graph.

Parameters

<i>graph</i>	Ellpack graph.
<i>vector</i>	Vector to store the graph.

Definition at line 2376 of file prg_system_mod.F90.

9.29.2.17 subroutine, public prg_system_mod::prg_make_random_system (type(system_type), intent(out) *system*, integer *nats*, integer *seed*, real(dp) *lx*, real(dp) *ly*, real(dp) *lz*)

Make random Xx system.

Parameters

<i>system</i>	System to be constructed.
<i>nats</i>	Number of atoms.
<i>lx</i>	length of the box for the x coordinate.
<i>ly</i>	length of the box for the y coordinate.
<i>lz</i>	length of the box for the z coordinate.

Definition at line 1059 of file prg_system_mod.F90.

9.29.2.18 subroutine, public prg_system_mod::prg_merge_graph (integer, dimension(:, :), intent(inout) *graph_p*, integer, dimension(:, :), intent(inout) *graph_h*)

Get partial subgraph based on the Density matrix.

Parameters

<i>graph_p</i>	Density matrix based graph in bml format.
<i>graph_h</i>	Hamiltonian matrix based graph in bml format.

Definition at line 2174 of file prg_system_mod.F90.

9.29.2.19 subroutine, public prg_system_mod::prg_merge_graph_adj (integer, dimension(:, :), intent(inout), allocatable *graph_p*, integer, dimension(:, :), intent(inout), allocatable *graph_h*, integer, dimension(:), intent(inout), allocatable *xadj*, integer, dimension(:), intent(inout), allocatable *adjncy*)

Get partial subgraph based on the Density matrix.

Parameters

<i>graph_p</i>	Density matrix based graph in "ellpack type format".
<i>graph_h</i>	Hamiltonian matrix based graph in "ellpack type format".
<i>xadj</i>	CSR start values for the adjacency matrix.
<i>adjncy</i>	CSR positions of adjacency matrix.

Definition at line 2225 of file prg_system_mod.F90.

9.29.2.20 subroutine, public prg_system_mod::prg_molpartition (type(system_type), intent(in) *sy*, integer, intent(inout) *npart*, real(dp), dimension(:, :), intent(in) *nnStructMindist*, integer, dimension(:, :), intent(in) *nnStruct*, integer, dimension(:), intent(in) *nrnnstruct*, character(2), intent(in) *hetatm*, type(graph_partitioning_t), intent(inout) *gp*, integer, intent(inout), optional *verbose*)

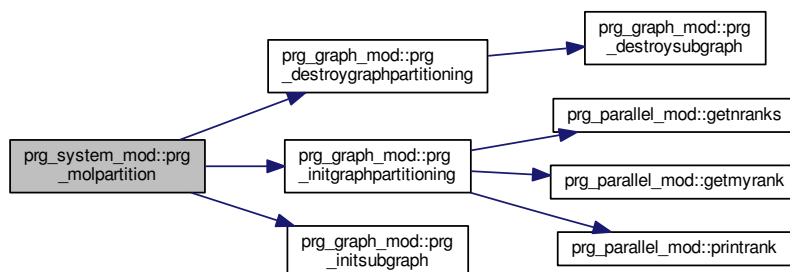
Partition by molecule.

Parameters

<i>sy</i>	System structure.
<i>npart</i>	Number of parts.
<i>nnStructMindist</i>	Minimum distance between neighbors.
<i>nnStruct</i>	The neighbors J to I within Rcut that are all within the box.
<i>nrnnstruct</i>	Number of neighbors to I within Rcut that are all within the box.
<i>hetatm</i>	Atom to be taken as the "center" of the by molecule partition.
<i>gp</i>	Graph partition structure.
<i>verbose</i>	Verbosity level.

Definition at line 1929 of file prg_system_mod.F90.

Here is the call graph for this function:



9.29.2.21 subroutine `prg_system_mod::prg_parameters_to_vectors` (`real(dp)`, `dimension(2,3)`, `intent(in)` `abc_angles`, `real(dp)`, `dimension(3,3)`, `intent(out)` `lattice_vector`) [`private`]

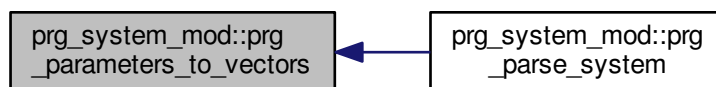
Transforms the lattice parameters into lattice vectors.

Parameters

<i>abc_angles</i>	2x3 array containing the lattice parameters. <code>abc_angles(1,1) = a</code> , <code>abc_angles(1,2) = b</code> , and <code>abc_angles(1,3) = c</code> <code>abc_angles(2,1) = α</code> , <code>abc_angles(2,2) = β</code> and <code>abc_angles(2,3) = γ</code>
<i>lattice_vector</i>	3x3 array containing the lattice vectors. <code>lattice_vector(1,:) = \vec{a}</code>

Definition at line 1105 of file prg_system_mod.F90.

Here is the caller graph for this function:



9.29.2.22 subroutine, public `prg_system_mod::prg_parse_system` (`type(system_type)`, `intent(out)` `system`, `character(len=*)` `filename`, `character(3)`, `intent(in)`, optional `extin`)

The parser for the chemical system.

Parameters

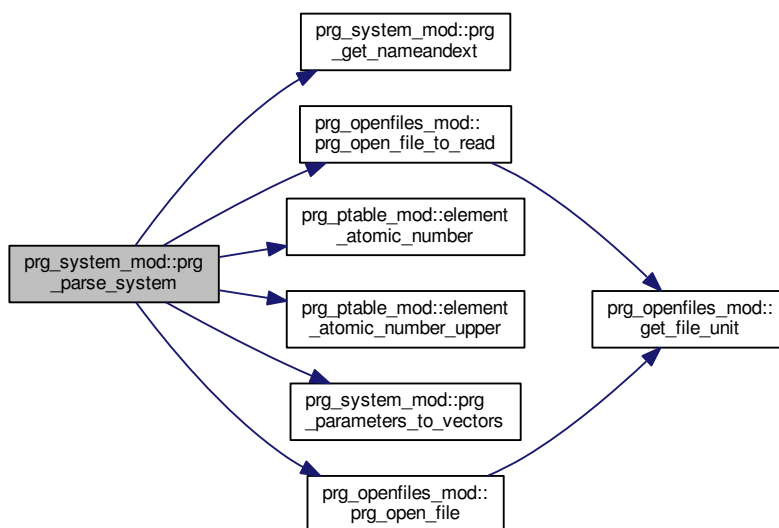
<i>system</i>	System to be constructed.
<i>filename</i>	Filename of the system.
<i>extin</i>	Extension of the file.

Assignment of species index for every atom.

Todo Integrate this loop in the loop for building the splist.

Definition at line 229 of file prg_system_mod.F90.

Here is the call graph for this function:



9.29.2.23 subroutine, public `prg_system_mod::prg_replicate (real(dp), dimension(:,:), intent(inout), allocatable coords, character(2), dimension(:), intent(inout), allocatable symbols, real(dp), dimension(:,:), intent(inout) lattice_vectors, integer, intent(in) nx, integer, intent(in) ny, integer, intent(in) nz)`

Extend/replicate system along lattice vectors.

Parameters

<i>coords</i>	Coordinates of the system (see system_type).
<i>symbols</i>	Symbols for elements.
<i>lattice_vectors</i>	System lattice vectors.
<i>nx</i>	Number of lattice points in the v1 direction.
<i>ny</i>	Number of lattice points in the v2 direction.
<i>nz</i>	Number of lattice points in the v2 direction.

Definition at line 1417 of file prg_system_mod.F90.

9.29.2.24 subroutine, public prg_system_mod::prg_sortadj (integer, dimension(:), intent(inout) *xadj*, integer, dimension(:), intent(inout), allocatable *adjncy*)

Sort adj NOTE: this might not be needed anymore since the bml_get_adj routine is sorting the values.

Definition at line 2433 of file prg_system_mod.F90.

9.29.2.25 subroutine, public prg_system_mod::prg_translateandfoldtobox (real(dp), dimension(:, :), intent(inout), allocatable *coords*, real(dp), dimension(:, :), intent(in) *lattice_vectors*, real(dp), dimension(:), intent(inout), allocatable *origin*, integer, intent(in), optional *verbose*)

Translate and fold to box.

Parameters

<i>coords</i>	Coordinates of the system (see system_type).
<i>lattice_vectors</i>	System lattice vectors.
<i>origin</i>	(min(x),min(y),min(z)) set as the origin of the system.

Definition at line 1243 of file prg_system_mod.F90.

9.29.2.26 subroutine, public prg_system_mod::prg_translategeomcandfoldtobox (real(dp), dimension(:, :), intent(inout), allocatable *coords*, real(dp), dimension(:, :), intent(in) *lattice_vectors*, real(dp), dimension(:), intent(inout), allocatable *origin*)

Translate to geometric center.

Parameters

<i>coords</i>	Coordinates of the system (see system_type).
<i>lattice_vectors</i>	System lattice vectors.
<i>origin</i>	(min(x),min(y),min(z)) set as the origin of the system.

Definition at line 1376 of file prg_system_mod.F90.

9.29.2.27 subroutine, public prg_system_mod::prg_vector2graph (integer, dimension(:), intent(inout), allocatable *vector*, integer, dimension(:, :), intent(inout) *graph*, integer *maxnz*)

Back to graph.

Parameters

<i>vector</i>	Vector to store the graph.
<i>graph</i>	Ellpack graph.

Definition at line 2405 of file prg_system_mod.F90.

9.29.2.28 subroutine prg_system_mod::prg_vectors_to_parameters (real(dp), dimension(3,3), intent(in) *lattice_vector*,
real(dp), dimension(2,3), intent(out) *abc_angles*) [private]

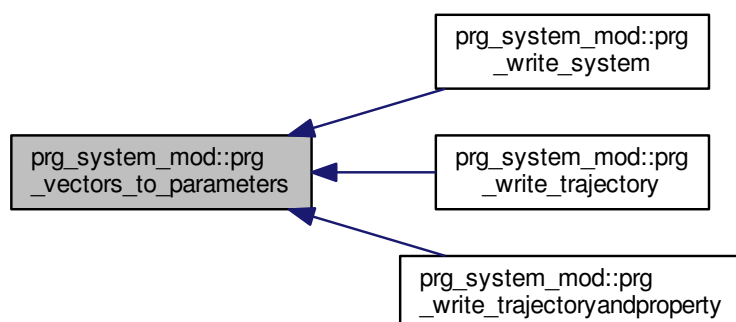
Transforms the lattice vectors into lattice parameters.

Parameters

<i>lattice_vector</i>	3x3 array containing the lattice vectors. $\text{lattice_vector}(1,:) = \vec{a}$
<i>abc_angles</i>	2x3 array containing the lattice parameters. $\text{abc_angles}(1,1) = a$, $\text{abc_angles}(1,2) = b$ and $\text{abc_angles}(1,3) = c$ $\text{abc_angles}(2,1) = \alpha$, $\text{abc_angles}(2,2) = \beta$, and $\text{abc_angles}(2,3) = \gamma$.

Definition at line 1147 of file prg_system_mod.F90.

Here is the caller graph for this function:



9.29.2.29 subroutine, public prg_system_mod::prg_wraparound (real(dp), dimension(:,,:), intent(inout), allocatable *coords*,
real(dp), dimension(:,,:), intent(in) *lattice_vectors*, integer, intent(in) *index*, integer, intent(in), optional *verbose*)

Wrap around atom i using pbc.

Parameters

<i>coords</i>	Coordinates of the system (see system_type).
<i>lattice_vectors</i>	System lattice vectors.
<i>index</i>	Index atom to wrap around

Definition at line 1333 of file prg_system_mod.F90.

9.29.2.30 subroutine, public prg_system_mod::prg_write_system (type(system_type), intent(in) system, character(*) filename, character(3) extension)

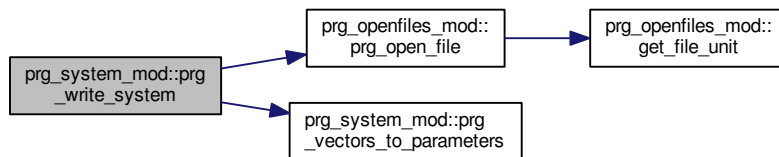
Write system in .xyz, .dat or pdb file.

Parameters

<i>system</i>	System to be constructed.
<i>filename</i>	File name.
<i>extension</i>	Extension of the file.

Definition at line 638 of file prg_system_mod.F90.

Here is the call graph for this function:



9.29.2.31 subroutine, public prg_system_mod::prg_write_trajectory (type(system_type), intent(in) system, integer, intent(in) iter, integer, intent(in) each, real(dp), intent(in) prg_deltat, character(*) filename, character(3) extension)

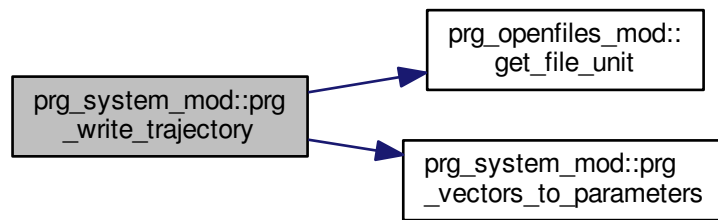
Write trajectory in .xyz, .dat or pdb file.

Parameters

<i>system</i>	System to be appended to the trajectory file.
<i>iter</i>	Simulation step.
<i>each</i>	Writing frequency.
<i>filename</i>	File name for the trajectory.
<i>extension</i>	Extension of the file.

Definition at line 826 of file prg_system_mod.F90.

Here is the call graph for this function:



9.29.2.32 subroutine, public `prg_system_mod::prg_write_trajectoryandproperty` (`type(system_type)`, `intent(in) system`, `integer`, `intent(in) iter`, `integer`, `intent(in) each`, `real(dp)`, `intent(in) prg_deltat`, `real(dp)`, `dimension(:)`, `intent(in) scalarprop`, `character(*) filename`, `character(3) extension`)

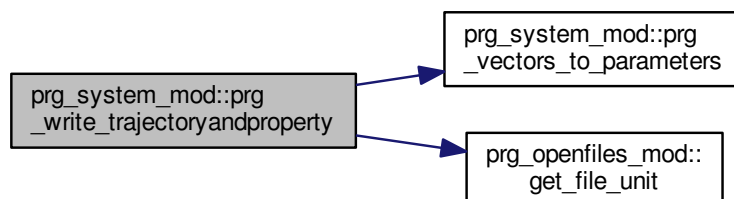
Write trajectory and atomic properties. Only pdb file.

Parameters

<i>system</i>	System to be appended to the trajectory file.
<i>iter</i>	Simulation step.
<i>each</i>	Writing frequency.
<i>prg_deltat</i>	Integration step.
<i>scalarprop</i>	Scalar property to plot on atoms.
<i>filename</i>	File name for the trajectory.
<i>extension</i>	Extension of the file.

Definition at line 949 of file `prg_system_mod.F90`.

Here is the call graph for this function:



9.29.3 Variable Documentation

9.29.3.1 integer, parameter prg_system_mod::dp = kind(1.0d0) [private]

Definition at line 19 of file prg_system_mod.F90.

9.30 prg_timer_mod Module Reference

The timer module.

Data Types

- type [timer_status_t](#)
Timer status type.

Functions/Subroutines

- subroutine, public [timer_prg_init](#) ()
Initialize timers.
- subroutine [prg_timer_getid](#) ()
Get timer id.
- subroutine, public [prg_timer_shutdown](#) ()
Done with timers.
- subroutine, public [prg_timer_start](#) (itimer, tag)
Start Timing.
- subroutine, public [prg_timer_stop](#) (itimer, verbose)
Stop timing.
- subroutine, public [prg_timer_collect](#) ()
- subroutine, public [prg_timer_results](#) ()
- real(8) function, public [time2milliseconds](#) ()
- subroutine, public [prg_print_date_and_time](#) (tag)
- character(2) function, private [int2char](#) (ival)

Variables

- integer, parameter [dp](#) = kind(1.0d0)
- integer, public [loop_timer](#)
- integer, public [sp2_timer](#)
- integer, public [genx_timer](#)
- integer, public [part_timer](#)
- integer, public [subgraph_timer](#)
- integer, public [deortho_timer](#)
- integer, public [ortho_timer](#)
- integer, public [zdiag_timer](#)
- integer, public [graphsp2_timer](#)
- integer, public [subind_timer](#)
- integer, public [subext_timer](#)
- integer, public [subsp2_timer](#)

- integer, public [suball_timer](#)
- integer, public [bmult_timer](#)
- integer, public [badd_timer](#)
- integer, public [dyn_timer](#)
- integer, public [mdloop_timer](#)
- integer, public [buildz_timer](#)
- integer, public [realcoul_timer](#)
- integer, public [recipcoul_timer](#)
- integer, public [pairpot_timer](#)
- integer, public [halfverlet_timer](#)
- integer, public [pos_timer](#)
- integer, public [nlist_timer](#)
- integer [tstart_clock](#)
- integer [tstop_clock](#)
- integer [tclock_rate](#)
- integer [tclock_max](#)
- integer [num_timers](#)
- type([timer_status_t](#)), dimension(:), allocatable [ptimer](#)

9.30.1 Detailed Description

The timer module.

Sets up timers that can be used to time other routines.

Example use of dynamic timing:

call [timer_prg_init\(\)](#)

call [prg_timer_start](#)(dyn_timer,"timer_tag")

.... code lines ...

call [prg_timer_stop](#)(dyn_timer,1)

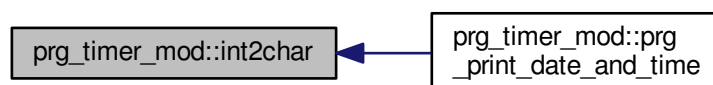
This will write the time it takes to execute "code lines" and it will name it "timer_tag"

9.30.2 Function/Subroutine Documentation

9.30.2.1 `character(2) function, private prg_timer_mod::int2char (integer, intent(in) ival) [private]`

Definition at line 394 of file `prg_timer_mod.F90`.

Here is the caller graph for this function:



9.30.2.2 subroutine, public prg_timer_mod::prg_print_date_and_time (character(len=*), intent(in) tag)

Definition at line 371 of file prg_timer_mod.F90.

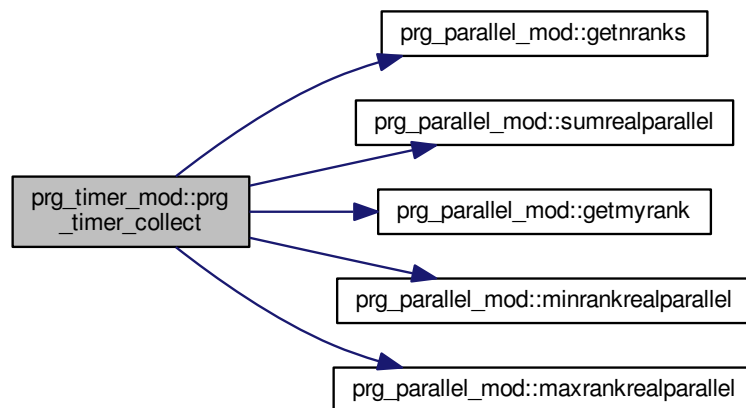
Here is the call graph for this function:



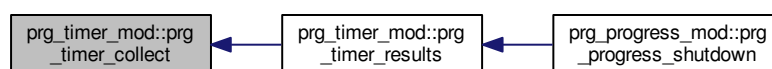
9.30.2.3 subroutine, public prg_timer_mod::prg_timer_collect ()

Definition at line 253 of file prg_timer_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



9.30.2.4 subroutine prg_timer_mod::prg_timer_getid () [private]

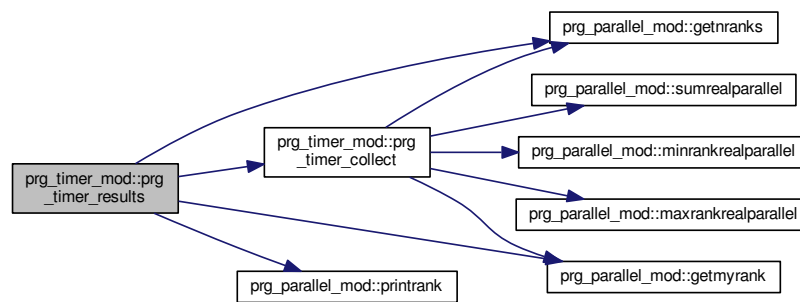
Get timer id.

Definition at line 200 of file prg_timer_mod.F90.

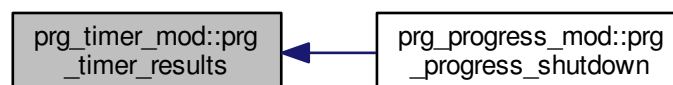
9.30.2.5 subroutine, public prg_timer_mod::prg_timer_results ()

Definition at line 317 of file prg_timer_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

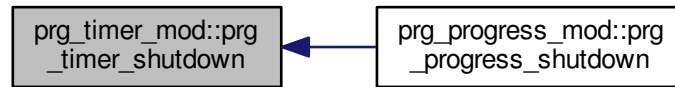


9.30.2.6 subroutine, public prg_timer_mod::prg_timer_shutdown ()

Done with timers.

Definition at line 205 of file prg_timer_mod.F90.

Here is the caller graph for this function:



9.30.2.7 subroutine, public `prg_timer_mod::prg_timer_start (integer, intent(in) itimer, character(len=*), intent(in), optional tag)`

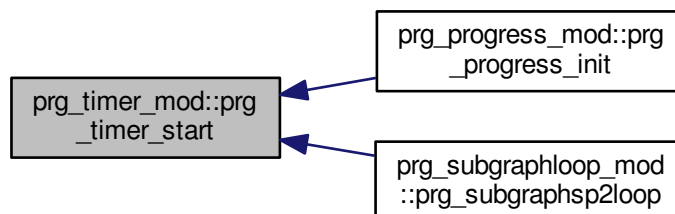
Start Timing.

Parameters

<i>itimer</i>	The index of the timer to start.
<i>tag</i>	Optional parameter to retag the timer on the fly.

Definition at line 215 of file `prg_timer_mod.F90`.

Here is the caller graph for this function:



9.30.2.8 subroutine, public `prg_timer_mod::prg_timer_stop (integer, intent(in) itimer, integer, intent(in), optional verbose)`

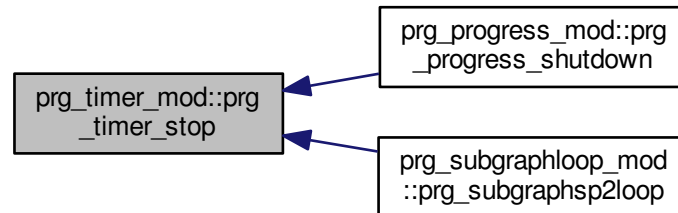
Stop timing.

Parameters

<i>itimer</i>	The index of the timer to stop.
<i>verbose</i>	Optional parameters to print partial times.

Definition at line 233 of file prg_timer_mod.F90.

Here is the caller graph for this function:



9.30.2.9 real(8) function, public prg_timer_mod::time2milliseconds ()

Definition at line 360 of file prg_timer_mod.F90.

9.30.2.10 subroutine, public prg_timer_mod::timer_prg_init ()

Initialize timers.

Definition at line 132 of file prg_timer_mod.F90.

Here is the caller graph for this function:



9.30.3 Variable Documentation

9.30.3.1 integer, public prg_timer_mod::badd_timer

Definition at line 48 of file prg_timer_mod.F90.

9.30.3.2 integer, public prg_timer_mod::bmult_timer

Definition at line 48 of file prg_timer_mod.F90.

9.30.3.3 integer, public prg_timer_mod::buildz_timer

Definition at line 49 of file prg_timer_mod.F90.

9.30.3.4 integer, public prg_timer_mod::deortho_timer

Definition at line 45 of file prg_timer_mod.F90.

9.30.3.5 integer, parameter prg_timer_mod::dp = kind(1.0d0) [private]

Definition at line 32 of file prg_timer_mod.F90.

9.30.3.6 integer, public prg_timer_mod::dyn_timer

Definition at line 49 of file prg_timer_mod.F90.

9.30.3.7 integer, public prg_timer_mod::genx_timer

Definition at line 44 of file prg_timer_mod.F90.

9.30.3.8 integer, public prg_timer_mod::graphsp2_timer

Definition at line 46 of file prg_timer_mod.F90.

9.30.3.9 integer, public prg_timer_mod::halfverlet_timer

Definition at line 51 of file prg_timer_mod.F90.

9.30.3.10 integer, public prg_timer_mod::loop_timer

Definition at line 44 of file prg_timer_mod.F90.

9.30.3.11 integer, public prg_timer_mod::mdloop_timer

Definition at line 49 of file prg_timer_mod.F90.

9.30.3.12 integer, public prg_timer_mod::nlist_timer

Definition at line 51 of file prg_timer_mod.F90.

9.30.3.13 integer prg_timer_mod::num_timers [private]

Definition at line 122 of file prg_timer_mod.F90.

9.30.3.14 integer, public prg_timer_mod::ortho_timer

Definition at line 46 of file prg_timer_mod.F90.

9.30.3.15 integer, public prg_timer_mod::pairpot_timer

Definition at line 50 of file prg_timer_mod.F90.

9.30.3.16 integer, public prg_timer_mod::part_timer

Definition at line 45 of file prg_timer_mod.F90.

9.30.3.17 integer, public prg_timer_mod::pos_timer

Definition at line 51 of file prg_timer_mod.F90.

9.30.3.18 type (timer_status_t), dimension(:), allocatable prg_timer_mod::ptimer [private]

Definition at line 124 of file prg_timer_mod.F90.

9.30.3.19 integer, public prg_timer_mod::realcoul_timer

Definition at line 50 of file prg_timer_mod.F90.

9.30.3.20 integer, public prg_timer_mod::recipcoul_timer

Definition at line 50 of file prg_timer_mod.F90.

9.30.3.21 integer, public prg_timer_mod::sp2_timer

Definition at line 44 of file prg_timer_mod.F90.

9.30.3.22 integer, public prg_timer_mod::suball_timer

Definition at line 48 of file prg_timer_mod.F90.

9.30.3.23 integer, public prg_timer_mod::subext_timer

Definition at line 47 of file prg_timer_mod.F90.

9.30.3.24 integer, public prg_timer_mod::subgraph_timer

Definition at line 45 of file prg_timer_mod.F90.

9.30.3.25 integer, public prg_timer_mod::subind_timer

Definition at line 47 of file prg_timer_mod.F90.

9.30.3.26 integer, public prg_timer_mod::subsp2_timer

Definition at line 47 of file prg_timer_mod.F90.

9.30.3.27 integer prg_timer_mod::tclock_max [private]

Definition at line 121 of file prg_timer_mod.F90.

9.30.3.28 integer prg_timer_mod::tclock_rate [private]

Definition at line 121 of file prg_timer_mod.F90.

9.30.3.29 integer prg_timer_mod::tstart_clock [private]

Definition at line 121 of file prg_timer_mod.F90.

9.30.3.30 integer prg_timer_mod::tstop_clock [private]

Definition at line 121 of file prg_timer_mod.F90.

9.30.3.31 integer, public prg_timer_mod::zdiag_timer

Definition at line 46 of file prg_timer_mod.F90.

9.31 prg_xlbo_mod Module Reference

A module to perform XLBO integration.

Data Types

- type `xlbo_type`
General xlbo solver type.

Functions/Subroutines

- subroutine, public `prg_parse_xlbo` (xlbo, filename)
The parser for XLBO parser.
- subroutine, public `prg_xlbo_nint` (charges, n, n_0, n_1, n_2, n_3, n_4, n_5, mdstep, xl)
This routine integrates the dynamical variable "n".
- subroutine, public `prg_xlbo_fcoulupdate` (fcoul, charges, n)
Adjust forces for the linearized XLBOMD functional.

Variables

- integer, parameter `dp` = kind(1.0d0)
- real(`dp`), parameter `c0` = -6.0_dp
Coefficients for modified Verlet integration.
- real(`dp`), parameter `c1` = 14.0_dp
- real(`dp`), parameter `c2` = -8.0_dp
- real(`dp`), parameter `c3` = -3.0_dp
- real(`dp`), parameter `c4` = 4.0_dp
- real(`dp`), parameter `c5` = -1.0_dp
- real(`dp`), parameter `kappa` = 1.82_dp
Coefficients for modified Verlet integration.
- real(`dp`), parameter `alpha` = 0.018_dp
- real(`dp`), parameter `cc` = 0.9_dp

9.31.1 Detailed Description

A module to perform XLBO integration.

This module will be used to compute integrate the dynamical variable "n" in xlbo.

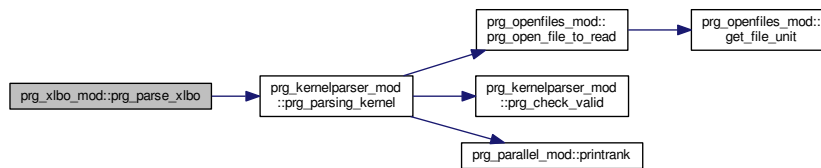
9.31.2 Function/Subroutine Documentation

9.31.2.1 subroutine, public prg_xlbo_mod::prg_parse_xlbo (type(xlbo_type), intent(inout) *xlbo*, character(len=*) *filename*)

The parser for XLBO parser.

Definition at line 62 of file prg_xlbo_mod.F90.

Here is the call graph for this function:



9.31.2.2 subroutine, public prg_xlbo_mod::prg_xlbo_fcoulupdate (real(dp), dimension(:, :), intent(inout) *fcoul*, real(dp), dimension(:), intent(inout) *charges*, real(dp), dimension(:), intent(inout) *n*)

Adjust forces for the linearized XLBOMD functional.

Parameters

<i>charges</i>	
----------------	--

Definition at line 158 of file prg_xlbo_mod.F90.

9.31.2.3 subroutine, public prg_xlbo_mod::prg_xlbo_nint (real(dp), dimension(:), intent(in), allocatable *charges*, real(dp), dimension(:), intent(inout), allocatable *n*, real(dp), dimension(:), intent(inout), allocatable *n_0*, real(dp), dimension(:), intent(inout), allocatable *n_1*, real(dp), dimension(:), intent(inout), allocatable *n_2*, real(dp), dimension(:), intent(inout), allocatable *n_3*, real(dp), dimension(:), intent(inout), allocatable *n_4*, real(dp), dimension(:), intent(inout), allocatable *n_5*, integer, intent(in) *mdstep*, type(xlbo_type), intent(in) *xl*)

This routine integrates the dynamical variable "n".

Parameters

<i>charges</i>	
----------------	--

Definition at line 118 of file prg_xlbo_mod.F90.

9.31.3 Variable Documentation

9.31.3.1 `real(dp), parameter prg_xlbo_mod::alpha = 0.018_dp` `[private]`

Definition at line 28 of file prg_xlbo_mod.F90.

9.31.3.2 `real(dp), parameter prg_xlbo_mod::c0 = -6.0_dp` `[private]`

Coefficients for modified Verlet integration.

Definition at line 19 of file prg_xlbo_mod.F90.

9.31.3.3 `real(dp), parameter prg_xlbo_mod::c1 = 14.0_dp` `[private]`

Definition at line 20 of file prg_xlbo_mod.F90.

9.31.3.4 `real(dp), parameter prg_xlbo_mod::c2 = -8.0_dp` `[private]`

Definition at line 21 of file prg_xlbo_mod.F90.

9.31.3.5 `real(dp), parameter prg_xlbo_mod::c3 = -3.0_dp` `[private]`

Definition at line 22 of file prg_xlbo_mod.F90.

9.31.3.6 `real(dp), parameter prg_xlbo_mod::c4 = 4.0_dp` `[private]`

Definition at line 23 of file prg_xlbo_mod.F90.

9.31.3.7 `real(dp), parameter prg_xlbo_mod::c5 = -1.0_dp` `[private]`

Definition at line 24 of file prg_xlbo_mod.F90.

9.31.3.8 `real(dp), parameter prg_xlbo_mod::cc = 0.9_dp` `[private]`

Definition at line 29 of file prg_xlbo_mod.F90.

9.31.3.9 `integer, parameter prg_xlbo_mod::dp = kind(1.0d0)` `[private]`

Definition at line 16 of file prg_xlbo_mod.F90.

9.31.3.10 `real(dp), parameter prg_xlbo_mod::kappa = 1.82_dp` `[private]`

Coefficients for modified Verlet integration.

Definition at line 27 of file prg_xlbo_mod.F90.

9.32 prg_xlkernel_mod Module Reference

Add name.

Data Types

- type [xlk_type](#)

Functions/Subroutines

- subroutine, public [prg_parse_xlkernel](#) (input, filename)
The parser for the mixer routines.
- subroutine, public [prg_fermi](#) (D0, QQ, ee, gap, Fe_vec, mu0, H, Z, Nocc, T, OccErrLim, MaxIt, HDIM)
- subroutine, public [prg_kernel_fermi_full](#) (KK, JJ, D0, mu0, mu1, T, RX, RY, RZ, LBox, Hubbard_U, Element↵_Type, Nr_atoms, MaxIt, eps, m, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERATIO, nnRx, nnRy, nnRz, nrnnlist, nnType, H_INDEX_START, H_INDEX_END, H, S, Z, Nocc, Znuc, QQ, ee, Fe_vec)
- subroutine, public [prg_v_kernel_fermi](#) (D0, dq_dv, v, mu0, mu1, T, RX, RY, RZ, LBox, Hubbard_U, Element↵_Type, Nr_atoms, MaxIt, eps, m, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERATIO, nnRx, nnRy, nnRz, nrnnlist, nnType, H_INDEX_START, H_INDEX_END, H, S, Z, Nocc, Znuc, QQ, ee, Fe_vec)
- subroutine, private [prg_get_deriv_finite_temp](#) (P1, H0, H1, Nocc, T, Q, ev, fe, mu0, eps, HDIM)
- subroutine, private [prg_mmult](#) (alpha, A, B, beta, C, TA, TB, HDIM)
- subroutine, private [prg_eig](#) (A, Q, ee, type, HDIM)
- subroutine, private [prg_inv](#) (X, XI, HDIM)
- subroutine, public [prg_rank1](#) (verbose)
Rank1 kernel

Variables

- integer, parameter [dp](#) = kind(1.0d0)

9.32.1 Detailed Description

Add name.

XL kernel (To be integrated)

Note

This module is still not functional

9.32.2 Function/Subroutine Documentation

- 9.32.2.1 subroutine, private prg_xlkernel_mod::prg_eig (real(prec), dimension(hdim,hdim), intent(in) A, real(prec), dimension(hdim,hdim), intent(out) Q, real(prec), dimension(hdim), intent(out) ee, character(1), intent(in) type, integer(prec), intent(in) HDIM) [private]

Definition at line 384 of file prg_xlkernel_mod.F90.

9.32.2.2 subroutine, public prg_xlkernel_mod::prg_fermi (real(prec), dimension(hdim,hdim), intent(out) *D0*, real(prec), dimension(hdim,hdim), intent(out) *QQ*, real(prec), dimension(hdim), intent(out) *ee*, real(prec), intent(out) *gap*, real(prec), dimension(hdim), intent(out) *Fe_vec*, real(prec), intent(inout) *mu0*, real(prec), dimension(hdim,hdim), intent(in) *H*, real(prec), dimension(hdim,hdim), intent(in) *Z*, integer(prec), intent(in) *Nocc*, real(prec), intent(in) *T*, real(prec), intent(in) *OccErrLim*, integer(prec), intent(in) *MaxIt*, integer(prec), intent(in) *HDIM*)

Definition at line 88 of file prg_xlkernel_mod.F90.

9.32.2.3 subroutine, private prg_xlkernel_mod::prg_get_deriv_finite_temp (real(prec), dimension(hdim,hdim), intent(out) *P1*, real(prec), dimension(hdim,hdim), intent(in) *H0*, real(prec), dimension(hdim,hdim), intent(in) *H1*, integer(prec), intent(in) *Nocc*, real(prec), intent(in) *T*, real(prec), dimension(hdim,hdim), intent(in) *Q*, real(prec), dimension(hdim), intent(in) *ev*, real(prec), dimension(hdim), intent(in) *fe*, real(prec), intent(inout) *mu0*, real(prec), intent(in) *eps*, integer(prec), intent(in) *HDIM*) [private]

Definition at line 306 of file prg_xlkernel_mod.F90.

9.32.2.4 subroutine, private prg_xlkernel_mod::prg_inv (real(prec), dimension(hdim,hdim), intent(in) *X*, real(prec), dimension(hdim,hdim), intent(out) *XI*, integer(prec), intent(in) *HDIM*) [private]

Definition at line 411 of file prg_xlkernel_mod.F90.

9.32.2.5 subroutine, public prg_xlkernel_mod::prg_kernel_fermi_full (real(prec), dimension(nr_atoms,nr_atoms), intent(out) *KK*, real(prec), dimension(nr_atoms,nr_atoms), intent(out) *JJ*, real(prec), dimension(hdim,hdim), intent(inout) *D0*, real(prec), intent(inout) *mu0*, real(prec), intent(inout) *mu1*, real(prec), intent(in) *T*, real(prec), dimension(nr_atoms), intent(in) *RX*, real(prec), dimension(nr_atoms), intent(in) *RY*, real(prec), dimension(nr_atoms), intent(in) *RZ*, real(prec), dimension(3), intent(in) *LBox*, real(prec), dimension(nr_atoms), intent(in) *Hubbard_U*, character(10), dimension(nr_atoms), intent(in) *Element_Type*, integer(prec), intent(in) *Nr_atoms*, integer(prec), intent(in) *MaxIt*, real(prec), intent(in) *eps*, integer(prec), intent(in) *m*, integer(prec), intent(in) *HDIM*, integer(prec), intent(in) *Max_Nr_Neigh*, real(prec), intent(in) *Coulomb_acc*, real(prec), intent(in) *TIMERATIO*, real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) *nnRx*, real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) *nnRy*, real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) *nnRz*, integer(prec), dimension(nr_atoms), intent(in) *nnnlist*, integer(prec), dimension(nr_atoms,max_nr_neigh), intent(in) *nnType*, integer(prec), dimension(nr_atoms), intent(in) *H_INDEX_START*, integer(prec), dimension(nr_atoms), intent(in) *H_INDEX_END*, real(prec), dimension(hdim,hdim), intent(in) *H*, real(prec), dimension(hdim,hdim), intent(in) *S*, real(prec), dimension(hdim,hdim), intent(in) *Z*, integer(prec), intent(in) *Nocc*, real(prec), dimension(nr_atoms), intent(in) *Znuc*, real(prec), dimension(hdim,hdim), intent(in) *QQ*, real(prec), dimension(hdim), intent(in) *ee*, real(prec), dimension(hdim), intent(in) *Fe_vec*)

Definition at line 144 of file prg_xlkernel_mod.F90.

9.32.2.6 subroutine, private prg_xlkernel_mod::prg_mmult (real(prec), intent(in) *alpha*, real(prec), dimension(hdim,hdim), intent(in) *A*, real(prec), dimension(hdim,hdim), intent(in) *B*, real(prec), intent(in) *beta*, real(prec), dimension(hdim,hdim), intent(inout) *C*, character(1), intent(in) *TA*, character(1), intent(in) *TB*, integer(prec), intent(in) *HDIM*) [private]

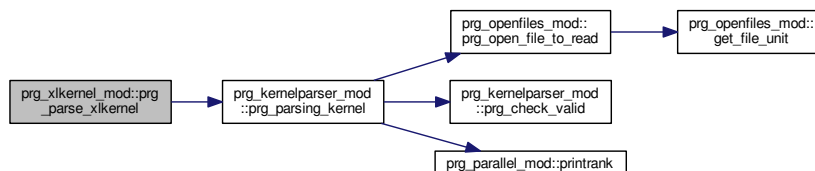
Definition at line 366 of file prg_xlkernel_mod.F90.

9.32.2.7 subroutine, public prg_xlkernel_mod::prg_parse_xlkernel (type(xlk_type), intent(inout) *input*, character(len=*) *filename*)

The parser for the mixer routines.

Definition at line 39 of file prg_xlkernel_mod.F90.

Here is the call graph for this function:



9.32.2.8 subroutine, public prg_xlkernel_mod::prg_rank1 (integer, intent(in) *verbose*)

Rank1 kernel

Parameters

<i>param1</i>	..
<i>verbose</i>	Different levels of verbosity.

Definition at line 439 of file prg_xlkernel_mod.F90.

9.32.2.9 subroutine, public prg_xlkernel_mod::prg_v_kernel_fermi (real(prec), dimension(hdim,hdim), intent(inout) *D0*, real(prec), dimension(nr_atoms), intent(out) *dq_dv*, real(prec), dimension(nr_atoms), intent(in) *v*, real(prec), intent(inout) *mu0*, real(prec), intent(inout) *mu1*, real(prec), intent(in) *T*, real(prec), dimension(nr_atoms), intent(in) *RX*, real(prec), dimension(nr_atoms), intent(in) *RY*, real(prec), dimension(nr_atoms), intent(in) *RZ*, real(prec), dimension(3), intent(in) *LBox*, real(prec), dimension(nr_atoms), intent(in) *Hubbard_U*, character(10), dimension(nr_atoms), intent(in) *Element_Type*, integer(prec), intent(in) *Nr_atoms*, integer(prec), intent(in) *MaxIt*, real(prec), intent(in) *eps*, integer(prec), intent(in) *m*, integer(prec), intent(in) *HDIM*, integer(prec), intent(in) *Max_Nr_Neigh*, real(prec), intent(in) *Coulomb_acc*, real(prec), intent(in) *TIMERATIO*, real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) *nnRx*, real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) *nnRy*, real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) *nnRz*, integer(prec), dimension(nr_atoms), intent(in) *nnnlist*, integer(prec), dimension(nr_atoms,max_nr_neigh), intent(in) *nnType*, integer(prec), dimension(nr_atoms), intent(in) *H_INDEX_START*, integer(prec), dimension(nr_atoms), intent(in) *H_INDEX_END*, real(prec), dimension(hdim,hdim), intent(in) *H*, real(prec), dimension(hdim,hdim), intent(in) *S*, real(prec), dimension(hdim,hdim), intent(in) *Z*, integer(prec), intent(in) *Nocc*, real(prec), dimension(nr_atoms), intent(in) *Znuc*, real(prec), dimension(hdim,hdim), intent(in) *QQ*, real(prec), dimension(hdim), intent(in) *ee*, real(prec), dimension(hdim), intent(in) *Fe_vec*)

Definition at line 235 of file prg_xlkernel_mod.F90.

9.32.3 Variable Documentation

9.32.3.1 integer, parameter `prg_xlkernel_mod::dp = kind(1.0d0)` `[private]`

Definition at line 15 of file `prg_xlkernel_mod.F90`.

Chapter 10

Data Type Documentation

10.1 prg_chebyshev_mod::chebdata_type Type Reference

General Cheb solver type.

Public Attributes

- character(100) [flavor](#)
- character(100) [bml_type](#)
- character(100) [jobname](#)
- integer [mdim](#)
- integer [ncoeffs](#)
- integer [ndim](#)
- integer [verbose](#)
- integer [npts](#)
- real(dp) [atr](#)
- real(dp) [bndfil](#)
- real(dp) [ef](#)
- real(dp) [estep](#)
- real(dp) [fermitol](#)
- real(dp) [kbt](#)
- real(dp) [threshold](#)
- logical [getef](#)
- logical [jon](#)
- logical [trkfunc](#)

10.1.1 Detailed Description

General Cheb solver type.

Definition at line 28 of file prg_chebyshev_mod.F90.

10.1.2 Member Data Documentation

10.1.2.1 `real(dp) prg_chebyshev_mod::chebdata_type::atr`

Definition at line 33 of file `prg_chebyshev_mod.F90`.

10.1.2.2 `character(100) prg_chebyshev_mod::chebdata_type::bml_type`

Definition at line 30 of file `prg_chebyshev_mod.F90`.

10.1.2.3 `real(dp) prg_chebyshev_mod::chebdata_type::bndfil`

Definition at line 33 of file `prg_chebyshev_mod.F90`.

10.1.2.4 `real(dp) prg_chebyshev_mod::chebdata_type::ef`

Definition at line 33 of file `prg_chebyshev_mod.F90`.

10.1.2.5 `real(dp) prg_chebyshev_mod::chebdata_type::estep`

Definition at line 33 of file `prg_chebyshev_mod.F90`.

10.1.2.6 `real(dp) prg_chebyshev_mod::chebdata_type::fermitol`

Definition at line 34 of file `prg_chebyshev_mod.F90`.

10.1.2.7 `character(100) prg_chebyshev_mod::chebdata_type::flavor`

Definition at line 29 of file `prg_chebyshev_mod.F90`.

10.1.2.8 `logical prg_chebyshev_mod::chebdata_type::getef`

Definition at line 35 of file `prg_chebyshev_mod.F90`.

10.1.2.9 `character(100) prg_chebyshev_mod::chebdata_type::jobname`

Definition at line 30 of file `prg_chebyshev_mod.F90`.

10.1.2.10 `logical prg_chebyshev_mod::chebdata_type::jon`

Definition at line 35 of file `prg_chebyshev_mod.F90`.

10.1.2.11 `real(dp) prg_chebyshev_mod::chebdata_type::kbt`

Definition at line 34 of file `prg_chebyshev_mod.F90`.

10.1.2.12 `integer prg_chebyshev_mod::chebdata_type::mdim`

Definition at line 31 of file `prg_chebyshev_mod.F90`.

10.1.2.13 `integer prg_chebyshev_mod::chebdata_type::ncoeffs`

Definition at line 31 of file `prg_chebyshev_mod.F90`.

10.1.2.14 `integer prg_chebyshev_mod::chebdata_type::ndim`

Definition at line 31 of file `prg_chebyshev_mod.F90`.

10.1.2.15 `integer prg_chebyshev_mod::chebdata_type::npts`

Definition at line 32 of file `prg_chebyshev_mod.F90`.

10.1.2.16 `real(dp) prg_chebyshev_mod::chebdata_type::threshold`

Definition at line 34 of file `prg_chebyshev_mod.F90`.

10.1.2.17 `logical prg_chebyshev_mod::chebdata_type::trkfunc`

Definition at line 35 of file `prg_chebyshev_mod.F90`.

10.1.2.18 `integer prg_chebyshev_mod::chebdata_type::verbose`

Definition at line 31 of file `prg_chebyshev_mod.F90`.

The documentation for this type was generated from the following file:

- `/home/christian/qmd-progress/src/prg_chebyshev_mod.F90`

10.2 prg_system_mod::estruct_type Type Reference

Electronic structure type.

Public Attributes

- integer [norbs](#)
Number of orbitals of the system.
- integer [nel](#)
Number of electrons.
- integer, dimension(:,:), allocatable [hindex](#)
Hindex.
- type(bml_matrix_t) [ham](#)
SCC-Hamiltonian of the system.
- type(bml_matrix_t) [ham0](#)
Hamiltonian of the system.
- type(bml_matrix_t) [oham](#)
Orthogonalized Hamiltonian.
- type(bml_matrix_t) [over](#)
Overlap matrix of the system.
- type(bml_matrix_t) [rho](#)
Density matrix of the system.
- type(bml_matrix_t) [orho](#)
Orthogonalized density matrix.
- type(bml_matrix_t) [zmat](#)
Congruence transformation.
- real([dp](#)), dimension(:), allocatable [coul_pot_r](#)
Real Coulombic contribution.
- real([dp](#)), dimension(:), allocatable [coul_pot_k](#)
Reciprocal Coulombic contribution.
- real([dp](#)), dimension(:,:), allocatable [skforce](#)
Slater Koster force.
- real([dp](#)), dimension(:,:), allocatable [fpul](#)
Pulay force.
- real([dp](#)), dimension(:,:), allocatable [fscoul](#)
Nonorthogonal Coulombic force.
- real([dp](#)) [eband](#)
Band energy.

10.2.1 Detailed Description

Electronic structure type.

The electronic structure type.

Definition at line 22 of file prg_system_mod.F90.

10.2.2 Member Data Documentation

10.2.2.1 `real(dp), dimension(:), allocatable prg_system_mod::estruct_type::coul_pot_k`

Reciprocal Coulombic contribution.

Definition at line 58 of file prg_system_mod.F90.

10.2.2.2 real(dp), dimension(:), allocatable prg_system_mod::estruct_type::coul_pot_r

Real Coulombic contribution.

Definition at line 55 of file prg_system_mod.F90.

10.2.2.3 real(dp) prg_system_mod::estruct_type::eband

Band energy.

Definition at line 70 of file prg_system_mod.F90.

10.2.2.4 real(dp), dimension(:, :), allocatable prg_system_mod::estruct_type::fpul

Pulay force.

Definition at line 64 of file prg_system_mod.F90.

10.2.2.5 real(dp), dimension(:, :), allocatable prg_system_mod::estruct_type::fscoul

Nonorthogonal Coulombic force.

Definition at line 67 of file prg_system_mod.F90.

10.2.2.6 type(bml_matrix_t) prg_system_mod::estruct_type::ham

SCC-Hamiltonian of the system.

Definition at line 34 of file prg_system_mod.F90.

10.2.2.7 type(bml_matrix_t) prg_system_mod::estruct_type::ham0

Hamiltonian of the system.

Definition at line 37 of file prg_system_mod.F90.

10.2.2.8 integer, dimension(:, :), allocatable prg_system_mod::estruct_type::hindex

Hindex.

Definition at line 31 of file prg_system_mod.F90.

10.2.2.9 integer prg_system_mod::estruct_type::nel

Number of electrons.

Definition at line 28 of file prg_system_mod.F90.

10.2.2.10 integer prg_system_mod::estruct_type::norbs

Number of orbitals of the system.

Definition at line 25 of file prg_system_mod.F90.

10.2.2.11 type(bml_matrix_t) prg_system_mod::estruct_type::oham

Orthogonalized Hamiltonian.

Definition at line 40 of file prg_system_mod.F90.

10.2.2.12 type(bml_matrix_t) prg_system_mod::estruct_type::orho

Orthogonalized density matrix.

Definition at line 49 of file prg_system_mod.F90.

10.2.2.13 type(bml_matrix_t) prg_system_mod::estruct_type::over

Overlap matrix of the system.

Definition at line 43 of file prg_system_mod.F90.

10.2.2.14 type(bml_matrix_t) prg_system_mod::estruct_type::rho

Density matrix of the system.

Definition at line 46 of file prg_system_mod.F90.

10.2.2.15 real(dp), dimension(:, :), allocatable prg_system_mod::estruct_type::skforce

Slater Koster force.

Definition at line 61 of file prg_system_mod.F90.

10.2.2.16 type(bml_matrix_t) prg_system_mod::estruct_type::zmat

Congruence transformation.

Definition at line 52 of file prg_system_mod.F90.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_system_mod.F90](#)

10.3 prg_genz_mod::genzspdata Type Reference

contains the data for the genZ driver.

Public Member Functions

- procedure [prg_init](#)
- procedure [prg_generate](#)
- procedure [prg_allocatezspmat](#)

Public Attributes

- integer [verbose](#)
- integer [nfirst](#)
- integer [nrefi](#)
- integer [nreff](#)
- real(dp) [numthresi](#)
- real(dp) [numthresf](#)
- logical [integration](#)

10.3.1 Detailed Description

contains the data for the genZ driver.

Definition at line 67 of file prg_genz_mod.F90.

10.3.2 Member Function/Subroutine Documentation

10.3.2.1 procedure prg_genz_mod::genzspdata::prg_allocatezspmat ()

Definition at line 79 of file prg_genz_mod.F90.

10.3.2.2 procedure prg_genz_mod::genzspdata::prg_generate ()

Definition at line 78 of file prg_genz_mod.F90.

10.3.2.3 procedure prg_genz_mod::genzspdata::prg_init ()

Definition at line 77 of file prg_genz_mod.F90.

10.3.3 Member Data Documentation

10.3.3.1 logical prg_genz_mod::genzspdata::integration

Definition at line 75 of file prg_genz_mod.F90.

10.3.3.2 integer prg_genz_mod::genzspdata::nfirst

Definition at line 70 of file prg_genz_mod.F90.

10.3.3.3 integer prg_genz_mod::genzspdata::nreff

Definition at line 72 of file prg_genz_mod.F90.

10.3.3.4 integer prg_genz_mod::genzspdata::nrefi

Definition at line 71 of file prg_genz_mod.F90.

10.3.3.5 real(dp) prg_genz_mod::genzspdata::numthresf

Definition at line 74 of file prg_genz_mod.F90.

10.3.3.6 real(dp) prg_genz_mod::genzspdata::numthresi

Definition at line 73 of file prg_genz_mod.F90.

10.3.3.7 integer prg_genz_mod::genzspdata::verbose

Definition at line 69 of file prg_genz_mod.F90.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_genz_mod.F90](#)

10.4 prg_genz_mod::genzspinp Type Reference

Input for the genz driver.

Public Attributes

- integer [verbose](#)
To have different levels of verbose.
- integer [nfirst](#)
!Lentgth of the "firsts iteration period".
- integer [nrefi](#)
!Initial number of recursive refinements.
- integer [nreff](#)
!Initial number of recursive refinements.
- real(dp) [numthresi](#)
Initial threshold value.
- real(dp) [numthresf](#)
Final threshold value.
- logical [integration](#)
If we want to do XL integration scheme for Z.
- integer [igenz](#)
To keep track of the genz iterations.
- logical [zsp](#)
Logical variable to compute in sparse or dense mode.
- integer [mdim](#)
Max nonzero elements per row for every row see [1] .
- character(20) [bml_type](#)
Matrix format (Dense or Ellpack).

10.4.1 Detailed Description

Input for the genz driver.

This type controls all the variables that are needed by genz

Definition at line 28 of file prg_genz_mod.F90.

10.4.2 Member Data Documentation

10.4.2.1 character(20) prg_genz_mod::genzspin::bml_type

Matrix format (Dense or Ellpack).

Definition at line 61 of file prg_genz_mod.F90.

10.4.2.2 integer prg_genz_mod::genzspin::igenz

To keep track of the genz iterations.

Definition at line 52 of file prg_genz_mod.F90.

10.4.2.3 logical prg_genz_mod::genzspinp::integration

If we want to do XL integration scheme for Z.

Definition at line 49 of file prg_genz_mod.F90.

10.4.2.4 integer prg_genz_mod::genzspinp::mdim

Max nonzero elements per row for every row see [1] .

Definition at line 58 of file prg_genz_mod.F90.

10.4.2.5 integer prg_genz_mod::genzspinp::nfirst

!Lentgth of the "firsts iteration period".

Definition at line 34 of file prg_genz_mod.F90.

10.4.2.6 integer prg_genz_mod::genzspinp::nreff

!Initial number of recursive refinements.

Definition at line 40 of file prg_genz_mod.F90.

10.4.2.7 integer prg_genz_mod::genzspinp::nrefi

!Initial number of recursive refinements.

Definition at line 37 of file prg_genz_mod.F90.

10.4.2.8 real(dp) prg_genz_mod::genzspinp::numthresf

Final threshold value.

Definition at line 46 of file prg_genz_mod.F90.

10.4.2.9 real(dp) prg_genz_mod::genzspinp::numthresi

Initial threshold value.

Definition at line 43 of file prg_genz_mod.F90.

10.4.2.10 integer prg_genz_mod::genzspinp::verbose

To have different levels of verbose.

Definition at line 31 of file prg_genz_mod.F90.

10.4.2.11 logical prg_genz_mod::genzspinp::zsp

Logical variable to compute in sparse or dense mode.

Definition at line 55 of file prg_genz_mod.F90.

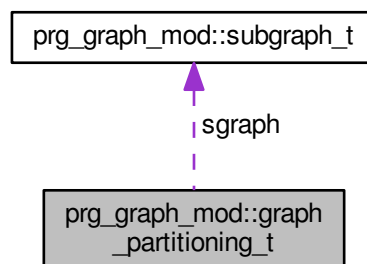
The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_genz_mod.F90](#)

10.5 prg_graph_mod::graph_partitioning_t Type Reference

Trace per iteration.

Collaboration diagram for prg_graph_mod::graph_partitioning_t:



Private Attributes

- character(len=100) [pname](#)
Partition name.
- integer [myrank](#)
Local processor.
- integer [totalprocs](#)
Number of processors.
- integer [totalparts](#)
Total number of global partitions.
- integer [totalnodes](#)
Total number of global groups, nodes (or matrix rows)
- integer [totalnodes2](#)
Total number of global nodes (or matrix rows)
- integer [globalpartmin](#)
Minimum global part number.
- integer [globalpartmax](#)
Maximum global part number.

- integer [globalpartextent](#)
Total global parts.
- integer, dimension(:), allocatable [localpartmin](#)
Minimum part per processor.
- integer, dimension(:), allocatable [localpartmax](#)
Maximum part per processor.
- integer, dimension(:), allocatable [localpartextent](#)
Number of parts per processor.
- integer, dimension(:), allocatable [order](#)
Original ordering if required.
- integer, dimension(:), allocatable [reorder](#)
Reordering if required.
- integer [nparts](#)
Total number of local partitions.
- integer, dimension(:), allocatable [nnodesinpart](#)
Number of nodes in each local partition.
- integer, dimension(:), allocatable [nnodesinpartall](#)
Number of nodes in each partition.
- integer, dimension(100) [pp](#)
Sequence for SP2.
- integer [maxiter](#)
Number of SP2 iterations.
- real(dp) [ehomo](#)
Homo value.
- real(dp) [elummo](#)
Lumo value.
- real(dp) [mineval](#)
Min eval for prg_normalize.
- real(dp) [maxeval](#)
Max eval for prg_normalize.
- real(dp), dimension(100) [vv](#)
Trace per iteration.
- type([subgraph_t](#)), dimension(:), allocatable [sgraph](#)
Subgraph details.

10.5.1 Detailed Description

Trace per iteration.

Graph partitioning type

Definition at line 57 of file prg_graph_mod.F90.

10.5.2 Member Data Documentation

10.5.2.1 `real(dp) prg_graph_mod::graph_partitioning_t::ehomo` [private]

Homo value.

Definition at line 117 of file prg_graph_mod.F90.

10.5.2.2 `real(dp) prg_graph_mod::graph_partitioning_t::elumo` `[private]`

Lumo value.

Definition at line 120 of file prg_graph_mod.F90.

10.5.2.3 `integer prg_graph_mod::graph_partitioning_t::globalpartextent` `[private]`

Total global parts.

Definition at line 84 of file prg_graph_mod.F90.

10.5.2.4 `integer prg_graph_mod::graph_partitioning_t::globalpartmax` `[private]`

Maximum global part number.

Definition at line 81 of file prg_graph_mod.F90.

10.5.2.5 `integer prg_graph_mod::graph_partitioning_t::globalpartmin` `[private]`

Minimum global part number.

Definition at line 78 of file prg_graph_mod.F90.

10.5.2.6 `integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::localpartextent` `[private]`

Number of parts per processor.

Definition at line 93 of file prg_graph_mod.F90.

10.5.2.7 `integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::localpartmax` `[private]`

Maximum part per processor.

Definition at line 90 of file prg_graph_mod.F90.

10.5.2.8 `integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::localpartmin` `[private]`

Minimum part per processor.

Definition at line 87 of file prg_graph_mod.F90.

10.5.2.9 `real(dp) prg_graph_mod::graph_partitioning_t::maxeval` `[private]`

Max eval for prg_normalize.

Definition at line 126 of file prg_graph_mod.F90.

10.5.2.10 `integer prg_graph_mod::graph_partitioning_t::maxiter` `[private]`

Number of SP2 iterations.

Definition at line 114 of file `prg_graph_mod.F90`.

10.5.2.11 `real(dp) prg_graph_mod::graph_partitioning_t::mineval` `[private]`

Min eval for `prg_normalize`.

Definition at line 123 of file `prg_graph_mod.F90`.

10.5.2.12 `integer prg_graph_mod::graph_partitioning_t::myrank` `[private]`

Local processor.

Definition at line 63 of file `prg_graph_mod.F90`.

10.5.2.13 `integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::nnodesinpart` `[private]`

Number of nodes in each local partition.

Definition at line 105 of file `prg_graph_mod.F90`.

10.5.2.14 `integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::nnodesinpartall` `[private]`

Number of nodes in each partition.

Definition at line 108 of file `prg_graph_mod.F90`.

10.5.2.15 `integer prg_graph_mod::graph_partitioning_t::nparts` `[private]`

Total number of local partitions.

Definition at line 102 of file `prg_graph_mod.F90`.

10.5.2.16 `integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::order` `[private]`

Original ordering if required.

Definition at line 96 of file `prg_graph_mod.F90`.

10.5.2.17 `character(len=100) prg_graph_mod::graph_partitioning_t::pname` `[private]`

Partition name.

Definition at line 60 of file `prg_graph_mod.F90`.

10.5.2.18 integer, dimension(100) prg_graph_mod::graph_partitioning_t::pp [private]

Sequence for SP2.

Definition at line 111 of file prg_graph_mod.F90.

10.5.2.19 integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::reorder [private]

Reordering if required.

Definition at line 99 of file prg_graph_mod.F90.

10.5.2.20 type (subgraph_t), dimension(:), allocatable prg_graph_mod::graph_partitioning_t::sgraph [private]

Subgraph details.

Definition at line 132 of file prg_graph_mod.F90.

10.5.2.21 integer prg_graph_mod::graph_partitioning_t::totalnodes [private]

Total number of global groups, nodes (or matrix rows)

Definition at line 72 of file prg_graph_mod.F90.

10.5.2.22 integer prg_graph_mod::graph_partitioning_t::totalnodes2 [private]

Total number of global nodes (or matrix rows)

Definition at line 75 of file prg_graph_mod.F90.

10.5.2.23 integer prg_graph_mod::graph_partitioning_t::totalparts [private]

Total number of global partitions.

Definition at line 69 of file prg_graph_mod.F90.

10.5.2.24 integer prg_graph_mod::graph_partitioning_t::totalprocs [private]

Number of processors.

Definition at line 66 of file prg_graph_mod.F90.

10.5.2.25 `real(dp), dimension(100) prg_graph_mod::graph_partitioning_t::vv` `[private]`

Trace per iteration.

Definition at line 129 of file `prg_graph_mod.F90`.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_graph_mod.F90](#)

10.6 prg_graphsp2parser_mod::gsp2data_type Type Reference

General SP2 solver type.

Public Attributes

- character(20) [jobname](#)
- character(50) [hamfile](#)
- integer [verbose](#)
- integer [minsp2iter](#)
- integer [maxsp2iter](#)
- integer [nodesperpart](#)
- integer [natoms](#)
- integer [partition_count](#)
- real(dp) [sp2tol](#)
- real(dp) [threshold](#)
- real(dp) [bndfil](#)
- real(dp) [gthreshold](#)
- real(dp) [errlimit](#)
- integer [mdim](#)
- integer [ndim](#)
- character, dimension(3) [sdim](#)
- real(dp), dimension(3) [pdim](#)
- character(20) [bml_type](#)
- character(10) [sp2conv](#)
- character(10) [graph_element](#)
- character(10) [partition_type](#)
- character(10) [partition_refinement](#)
- logical [double_jump](#)
- real(dp) [covgfact](#)
- real(dp) [nlgcut](#)
- integer [parteach](#)

10.6.1 Detailed Description

General SP2 solver type.

Definition at line 26 of file `prg_graphsp2parser_mod.F90`.

10.6.2 Member Data Documentation

10.6.2.1 character(20) prg_graphsp2parser_mod::gsp2data_type::bml_type

Definition at line 44 of file prg_graphsp2parser_mod.F90.

10.6.2.2 real(dp) prg_graphsp2parser_mod::gsp2data_type::bndfil

Definition at line 37 of file prg_graphsp2parser_mod.F90.

10.6.2.3 real(dp) prg_graphsp2parser_mod::gsp2data_type::covgfact

Definition at line 50 of file prg_graphsp2parser_mod.F90.

10.6.2.4 logical prg_graphsp2parser_mod::gsp2data_type::double_jump

Definition at line 49 of file prg_graphsp2parser_mod.F90.

10.6.2.5 real(dp) prg_graphsp2parser_mod::gsp2data_type::errlimit

Definition at line 39 of file prg_graphsp2parser_mod.F90.

10.6.2.6 character(10) prg_graphsp2parser_mod::gsp2data_type::graph_element

Definition at line 46 of file prg_graphsp2parser_mod.F90.

10.6.2.7 real(dp) prg_graphsp2parser_mod::gsp2data_type::gthreshold

Definition at line 38 of file prg_graphsp2parser_mod.F90.

10.6.2.8 character(50) prg_graphsp2parser_mod::gsp2data_type::hamfile

Definition at line 28 of file prg_graphsp2parser_mod.F90.

10.6.2.9 character(20) prg_graphsp2parser_mod::gsp2data_type::jobname

Definition at line 27 of file prg_graphsp2parser_mod.F90.

10.6.2.10 integer prg_graphsp2parser_mod::gsp2data_type::maxsp2iter

Definition at line 31 of file prg_graphsp2parser_mod.F90.

10.6.2.11 integer prg_graphsp2parser_mod::gsp2data_type::mdim

Definition at line 40 of file prg_graphsp2parser_mod.F90.

10.6.2.12 integer prg_graphsp2parser_mod::gsp2data_type::minsp2iter

Definition at line 30 of file prg_graphsp2parser_mod.F90.

10.6.2.13 integer prg_graphsp2parser_mod::gsp2data_type::natoms

Definition at line 33 of file prg_graphsp2parser_mod.F90.

10.6.2.14 integer prg_graphsp2parser_mod::gsp2data_type::ndim

Definition at line 41 of file prg_graphsp2parser_mod.F90.

10.6.2.15 real(dp) prg_graphsp2parser_mod::gsp2data_type::nlgcut

Definition at line 51 of file prg_graphsp2parser_mod.F90.

10.6.2.16 integer prg_graphsp2parser_mod::gsp2data_type::nodesperpart

Definition at line 32 of file prg_graphsp2parser_mod.F90.

10.6.2.17 integer prg_graphsp2parser_mod::gsp2data_type::parteach

Definition at line 52 of file prg_graphsp2parser_mod.F90.

10.6.2.18 integer prg_graphsp2parser_mod::gsp2data_type::partition_count

Definition at line 34 of file prg_graphsp2parser_mod.F90.

10.6.2.19 character(10) prg_graphsp2parser_mod::gsp2data_type::partition_refinement

Definition at line 48 of file prg_graphsp2parser_mod.F90.

10.6.2.20 character(10) prg_graphsp2parser_mod::gsp2data_type::partition_type

Definition at line 47 of file prg_graphsp2parser_mod.F90.

10.6.2.21 `real(dp), dimension(3) prg_graphsp2parser_mod::gsp2data_type::pdim`

Definition at line 43 of file `prg_graphsp2parser_mod.F90`.

10.6.2.22 `character, dimension(3) prg_graphsp2parser_mod::gsp2data_type::sdim`

Definition at line 42 of file `prg_graphsp2parser_mod.F90`.

10.6.2.23 `character(10) prg_graphsp2parser_mod::gsp2data_type::sp2conv`

Definition at line 45 of file `prg_graphsp2parser_mod.F90`.

10.6.2.24 `real(dp) prg_graphsp2parser_mod::gsp2data_type::sp2tol`

Definition at line 35 of file `prg_graphsp2parser_mod.F90`.

10.6.2.25 `real(dp) prg_graphsp2parser_mod::gsp2data_type::threshold`

Definition at line 36 of file `prg_graphsp2parser_mod.F90`.

10.6.2.26 `integer prg_graphsp2parser_mod::gsp2data_type::verbose`

Definition at line 29 of file `prg_graphsp2parser_mod.F90`.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_graphsp2parser_mod.F90](#)

10.7 prg_pulaymixer_mod::mx_type Type Reference

Public Attributes

- `character(20) mixertype`
Type or mixing scheme to be used (Linear or Pulay)
- `integer verbose`
Verbosity level.
- `integer mpulay`
Pulay dimension for matrix.
- `real(dp) mixcoeff`
Coefficient for mixing.
- `logical mixeron`
Mixer on or off (Not implemented)

10.7.1 Detailed Description

Definition at line 16 of file prg_pulaymixer_mod.F90.

10.7.2 Member Data Documentation

10.7.2.1 `real(dp) prg_pulaymixer_mod::mx_type::mixcoeff`

Coefficient for mixing.

Definition at line 28 of file prg_pulaymixer_mod.F90.

10.7.2.2 `logical prg_pulaymixer_mod::mx_type::mixeron`

Mixer on or off (Not implemented)

Definition at line 31 of file prg_pulaymixer_mod.F90.

10.7.2.3 `character(20) prg_pulaymixer_mod::mx_type::mixertype`

Type or mixing scheme to be used (Linear or Pulay)

Definition at line 19 of file prg_pulaymixer_mod.F90.

10.7.2.4 `integer prg_pulaymixer_mod::mx_type::mpulay`

Pulay dimension for matrix.

Definition at line 25 of file prg_pulaymixer_mod.F90.

10.7.2.5 `integer prg_pulaymixer_mod::mx_type::verbose`

Verbosity level.

Definition at line 22 of file prg_pulaymixer_mod.F90.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_pulaymixer_mod.F90](#)

10.8 `prg_extras_mod::prg_memory_consumption` Interface Reference

Private Member Functions

- subroutine [prg_memory_consumption](#) (vm_peak, vm_size, pid, ppid)

10.8.1 Detailed Description

Definition at line 15 of file prg_extras_mod.F90.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 subroutine prg_extras_mod::prg_memory_consumption::prg_memory_consumption (integer(c_long_long), intent(inout) *vm_peak*, integer(c_long_long), intent(inout) *vm_size*, integer(c_long_long), intent(inout) *pid*, integer(c_long_long), intent(inout) *ppid*) [private]

Definition at line 17 of file prg_extras_mod.F90.

The documentation for this interface was generated from the following file:

- [/home/christian/qmd-progress/src/prg_extras_mod.F90](#)

10.9 prg_parallel_mod::rankreducedata_t Type Reference

Data structure for rection over MPI ranks.

Private Attributes

- real(dp) *val*
Data value.
- integer *rank*
MPI rank.

10.9.1 Detailed Description

Data structure for rection over MPI ranks.

Definition at line 72 of file prg_parallel_mod.F90.

10.9.2 Member Data Documentation

10.9.2.1 integer prg_parallel_mod::rankreducedata_t::rank [private]

MPI rank.

Definition at line 78 of file prg_parallel_mod.F90.

10.9.2.2 `real(dp) prg_parallel_mod::rankreducedata_t::val` `[private]`

Data value.

Definition at line 75 of file `prg_parallel_mod.F90`.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_parallel_mod.F90](#)

10.10 `prg_response_mod::respdata_type` Type Reference

Public Attributes

- character(20) [respmode](#)
- character(20) [typeofpert](#)
- character(20) [bmltype](#)
- integer [mdim](#)
- real(dp) [numthresh](#)
- logical [computedipole](#)
- logical [getresponse](#)
- real(dp) [fieldintensity](#)
- real(dp), dimension(3) [field](#)

10.10.1 Detailed Description

Definition at line 21 of file `prg_response_mod.F90`.

10.10.2 Member Data Documentation

10.10.2.1 `character(20) prg_response_mod::respdata_type::bmltype`

Definition at line 24 of file `prg_response_mod.F90`.

10.10.2.2 `logical prg_response_mod::respdata_type::computedipole`

Definition at line 27 of file `prg_response_mod.F90`.

10.10.2.3 `real(dp), dimension(3) prg_response_mod::respdata_type::field`

Definition at line 30 of file `prg_response_mod.F90`.

10.10.2.4 `real(dp) prg_response_mod::respdata_type::fieldintensity`

Definition at line 29 of file `prg_response_mod.F90`.

10.10.2.5 logical prg_response_mod::respdata_type::getresponse

Definition at line 28 of file prg_response_mod.F90.

10.10.2.6 integer prg_response_mod::respdata_type::mdim

Definition at line 25 of file prg_response_mod.F90.

10.10.2.7 real(dp) prg_response_mod::respdata_type::numthresh

Definition at line 26 of file prg_response_mod.F90.

10.10.2.8 character(20) prg_response_mod::respdata_type::respmode

Definition at line 22 of file prg_response_mod.F90.

10.10.2.9 character(20) prg_response_mod::respdata_type::typeofpert

Definition at line 23 of file prg_response_mod.F90.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_response_mod.F90](#)

10.11 prg_syrotation_mod::rotation_type Type Reference

Rotation type.

Public Attributes

- character(20) [jobname](#)
- character(50) [typeofrot](#)
- integer [patom1](#)
Atomic point to determine the initial orientation.
- integer [patom2](#)
Atomic point to determine initial orientation.
- integer [catom](#)
Atomic point to determine the rotation center.
- integer [catom2](#)
Atomic point to determine a second rotation center.
- real(dp), dimension(3) [pq1](#)
Point to determine initial orientation.
- real(dp), dimension(3) [pq2](#)
Point to determine final orientation.
- real(dp), dimension(3) [v1](#)
Initial orientation.
- real(dp), dimension(3) [v2](#)
Final orientation.
- real(dp), dimension(3) [vq](#)
Center of rotation.
- integer, dimension(2) [rotate_atoms](#)
First and last rotated atom in the list.

10.11.1 Detailed Description

Rotation type.

Definition at line 17 of file prg_syrotation_mod.F90.

10.11.2 Member Data Documentation

10.11.2.1 integer prg_syrotation_mod::rotation_type::catom

Atomic point to determine the rotation center.

Definition at line 25 of file prg_syrotation_mod.F90.

10.11.2.2 integer prg_syrotation_mod::rotation_type::catom2

Atomic point to determine a second rotation center.

Definition at line 27 of file prg_syrotation_mod.F90.

10.11.2.3 character(20) prg_syrotation_mod::rotation_type::jobname

Definition at line 18 of file prg_syrotation_mod.F90.

10.11.2.4 integer prg_syrotation_mod::rotation_type::patom1

Atomic point to determine the initial orientation.

Definition at line 21 of file prg_syrotation_mod.F90.

10.11.2.5 integer prg_syrotation_mod::rotation_type::patom2

Atomic point to determine initial orientation.

Definition at line 23 of file prg_syrotation_mod.F90.

10.11.2.6 real(dp), dimension(3) prg_syrotation_mod::rotation_type::pq1

Point to determine initial orientation.

Definition at line 29 of file prg_syrotation_mod.F90.

10.11.2.7 real(dp), dimension(3) prg_syrotation_mod::rotation_type::pq2

Point to determine final orientation.

Definition at line 31 of file prg_syrotation_mod.F90.

10.11.2.8 integer, dimension(2) prg_syrotation_mod::rotation_type::rotate_atoms

First and last rotated atom in the list.

Definition at line 39 of file prg_syrotation_mod.F90.

10.11.2.9 character(50) prg_syrotation_mod::rotation_type::typeofrot

Definition at line 19 of file prg_syrotation_mod.F90.

10.11.2.10 real(dp), dimension(3) prg_syrotation_mod::rotation_type::v1

Initial orientation.

Definition at line 33 of file prg_syrotation_mod.F90.

10.11.2.11 real(dp), dimension(3) prg_syrotation_mod::rotation_type::v2

Final orientation.

Definition at line 35 of file prg_syrotation_mod.F90.

10.11.2.12 real(dp), dimension(3) prg_syrotation_mod::rotation_type::vq

Center of rotation.

Definition at line 37 of file prg_syrotation_mod.F90.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_syrotation_mod.F90](#)

10.12 prg_sp2parser_mod::sp2data_type Type Reference

General SP2 solver type.

Public Attributes

- character(20) [jobname](#)
- integer [verbose](#)
- integer [minsp2iter](#)
- integer [maxsp2iter](#)
- real(dp) [sp2tol](#)
- real(dp) [threshold](#)
- real(dp) [bndfil](#)
- integer [mdim](#)
- integer [ndim](#)
- character, dimension(3) [sdim](#)
- real(dp), dimension(3) [pdim](#)
- character(20) [bml_type](#)
- character(10) [sp2conv](#)
- character(10) [flavor](#)

10.12.1 Detailed Description

General SP2 solver type.

Definition at line 26 of file `prg_sp2parser_mod.F90`.

10.12.2 Member Data Documentation

10.12.2.1 character(20) `prg_sp2parser_mod::sp2data_type::bml_type`

Definition at line 38 of file `prg_sp2parser_mod.F90`.

10.12.2.2 real(dp) `prg_sp2parser_mod::sp2data_type::bndfil`

Definition at line 33 of file `prg_sp2parser_mod.F90`.

10.12.2.3 character(10) `prg_sp2parser_mod::sp2data_type::flavor`

Definition at line 40 of file `prg_sp2parser_mod.F90`.

10.12.2.4 character(20) `prg_sp2parser_mod::sp2data_type::jobname`

Definition at line 27 of file `prg_sp2parser_mod.F90`.

10.12.2.5 integer `prg_sp2parser_mod::sp2data_type::maxsp2iter`

Definition at line 30 of file `prg_sp2parser_mod.F90`.

10.12.2.6 integer prg_sp2parser_mod::sp2data_type::mdim

Definition at line 34 of file prg_sp2parser_mod.F90.

10.12.2.7 integer prg_sp2parser_mod::sp2data_type::minsp2iter

Definition at line 29 of file prg_sp2parser_mod.F90.

10.12.2.8 integer prg_sp2parser_mod::sp2data_type::ndim

Definition at line 35 of file prg_sp2parser_mod.F90.

10.12.2.9 real(dp), dimension(3) prg_sp2parser_mod::sp2data_type::pdim

Definition at line 37 of file prg_sp2parser_mod.F90.

10.12.2.10 character, dimension(3) prg_sp2parser_mod::sp2data_type::sdim

Definition at line 36 of file prg_sp2parser_mod.F90.

10.12.2.11 character(10) prg_sp2parser_mod::sp2data_type::sp2conv

Definition at line 39 of file prg_sp2parser_mod.F90.

10.12.2.12 real(dp) prg_sp2parser_mod::sp2data_type::sp2tol

Definition at line 31 of file prg_sp2parser_mod.F90.

10.12.2.13 real(dp) prg_sp2parser_mod::sp2data_type::threshold

Definition at line 32 of file prg_sp2parser_mod.F90.

10.12.2.14 integer prg_sp2parser_mod::sp2data_type::verbose

Definition at line 28 of file prg_sp2parser_mod.F90.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_sp2parser_mod.F90](#)

10.13 prg_graph_mod::subgraph_t Type Reference

Subgraph type.

Private Attributes

- integer [part](#)
Partition number.
- integer [hsize](#)
Size of original matrix (h x h)
- integer [lsize](#)
Size of full subgraph (l x l)
- integer [llsize](#)
Size of core subgraph.
- integer, dimension(:), allocatable [core_halo_index](#)
Indeces from original matrix for subgraph core+halo extraction.
- integer, dimension(:), allocatable [nodeinpart](#)
Nodes in this partition.

10.13.1 Detailed Description

Subgraph type.

Definition at line 31 of file prg_graph_mod.F90.

10.13.2 Member Data Documentation

10.13.2.1 integer, dimension(:), allocatable `prg_graph_mod::subgraph_t::core_halo_index` `[private]`

Indeces from original matrix for subgraph core+halo extraction.

Definition at line 46 of file prg_graph_mod.F90.

10.13.2.2 integer `prg_graph_mod::subgraph_t::hsize` `[private]`

Size of original matrix (h x h)

Definition at line 37 of file prg_graph_mod.F90.

10.13.2.3 integer `prg_graph_mod::subgraph_t::llsize` `[private]`

Size of core subgraph.

Definition at line 43 of file prg_graph_mod.F90.

10.13.2.4 integer prg_graph_mod::subgraph_t::lsize [private]

Size of full subgraph (l x l)

Definition at line 40 of file prg_graph_mod.F90.

10.13.2.5 integer, dimension(:), allocatable prg_graph_mod::subgraph_t::nodeinpart [private]

Nodes in this partition.

Definition at line 49 of file prg_graph_mod.F90.

10.13.2.6 integer prg_graph_mod::subgraph_t::part [private]

Partition number.

Definition at line 34 of file prg_graph_mod.F90.

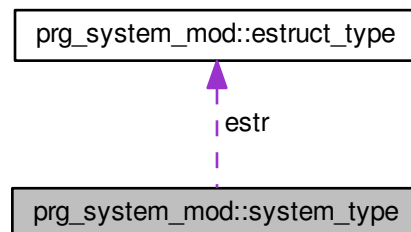
The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_graph_mod.F90](#)

10.14 prg_system_mod::system_type Type Reference

System type.

Collaboration diagram for prg_system_mod::system_type:



Public Attributes

- integer **nats**
Number of atoms of the system.
- character(2), dimension(:), allocatable **symbol**
Chemical Symbols for every atom of the system. Symbol can be recovered using ptable module and calling the following routine:
- integer, dimension(:), allocatable **atomic_number**
Atomic number for every atom in the system.
- real(**dp**), dimension(:, :), allocatable **coordinate**
Coordinates of every atom in the system. Allocation:
- real(**dp**), dimension(:, :), allocatable **velocity**
Velocities for every atom in the system. Allocation:
- real(**dp**), dimension(:, :), allocatable **force**
Forces acting on every atom in the system. Allocation:
- real(**dp**), dimension(:), allocatable **net_charge**
Charges of every atom in the system. Allocation:
- real(**dp**), dimension(:), allocatable **mass**
Mass of every atom in the system. These can be automatically loaded by using the structures of the ptable mod:
- real(**dp**), dimension(:, :), allocatable **lattice_vector**
Lattice vectors of the system. Use the prg_vectors_to_parameters and parameters_to_vector to transform from lattice vector to lattice parameters. Allocation:
- real(**dp**), dimension(:, :), allocatable **recip_vector**
Reciprocal vectors of the system. Allocation:
- real(**dp**) **volr**
Volume of the system (direct space).
- real(**dp**) **volk**
Volume of the system (direct space).
- integer **nsp**
Number of different species. Number of species or number of different atom types (symbols) in the system. This integer is always less or equal than the total number of atoms (nsp ≤ nats). This information can also be found in tparams structure and the following equality holds:
- integer, dimension(:), allocatable **spindex**
Species index. It gives the species index of a particular atom. Allocation:
- character(2), dimension(:), allocatable **splist**
Species symbol list. A list with the different species e.g. H, C, N, etc with the order corresponding to the appearance in systemsymbol. Allocation:
- integer, dimension(:), allocatable **spatnum**
Species atomic number list. A list with the atomic numbers for every species Allocation:
- real(**dp**), dimension(:), allocatable **spmass**
Species mass list. A list with the atomic mass for every species Allocation:
- real(**dp**), dimension(:), allocatable **userdef**
User define field.
- integer, dimension(:), allocatable **resindex**
Residue index.
- type(**estruct_type**) **estr**
Electronic structure.

10.14.1 Detailed Description

System type.

The molecular system type.

Definition at line 75 of file prg_system_mod.F90.

10.14.2 Member Data Documentation

10.14.2.1 integer, dimension(:), allocatable prg_system_mod::system_type::atomic_number

Atomic number for every atom in the system.

Definition at line 89 of file prg_system_mod.F90.

10.14.2.2 real(dp), dimension(:, :), allocatable prg_system_mod::system_type::coordinate

Coordinates of every atom in the system. Allocation:

```
coordinate(3,nats)
```

Definition at line 94 of file prg_system_mod.F90.

10.14.2.3 type(estruct_type) prg_system_mod::system_type::estr

Electronic structure.

Definition at line 187 of file prg_system_mod.F90.

10.14.2.4 real(dp), dimension(:, :), allocatable prg_system_mod::system_type::force

Forces acting on every atom in the system. Allocation:

```
force(3,nats)
```

Definition at line 104 of file prg_system_mod.F90.

10.14.2.5 real(dp), dimension(:, :), allocatable prg_system_mod::system_type::lattice_vector

Lattice vectors of the system. Use the prg_vectors_to_parameters and parameters_to_vector to transform from lattice vector to lattice parameters. Allocation:

```
lattice_vector(3,3)
```

```
v1 = lattice_vector(1,:)

```

```
v2 = lattice_vector(2,:)

```

```
v3 = lattice_vector(3,:)

```

Definition at line 126 of file prg_system_mod.F90.

10.14.2.6 real(dp), dimension(:), allocatable prg_system_mod::system_type::mass

Mass of every atom in the system. These can be automatically loaded by using the structures of the ptable mod:

```
system%mass(i) = mass(mystem%atomic_number(i))
```

Allocation:

```
mass(nats)
```

Definition at line 116 of file prg_system_mod.F90.

10.14.2.7 integer prg_system_mod::system_type::nats

Number of atoms of the system.

Definition at line 78 of file prg_system_mod.F90.

10.14.2.8 real(dp), dimension(:), allocatable prg_system_mod::system_type::net_charge

Charges of every atom in the system. Allocation:

```
net_charge(nats)
```

Definition at line 109 of file prg_system_mod.F90.

10.14.2.9 integer prg_system_mod::system_type::nsp

Number of different species. Number of species or number of different atom types (symbols) in the system. This integer is always less or equal than the total number of atoms ($nsp \leq nats$). This information can also be found in tbparams structure and the following equality holds:

```
system%nsp = tbparams%nsp
```

Definition at line 149 of file prg_system_mod.F90.

10.14.2.10 real(dp), dimension(:, :), allocatable prg_system_mod::system_type::recip_vector

Reciprocal vectors of the system. Allocation:

```
recip_vector(3,3)
```

```
v1 = recip_vector(1,:)
```

```
v2 = recip_vector(2,:)
```

```
v3 = recip_vector(3,:)
```

Definition at line 134 of file prg_system_mod.F90.

10.14.2.11 integer, dimension(:), allocatable prg_system_mod::system_type::resindex

Residue index.

Definition at line 184 of file prg_system_mod.F90.

10.14.2.12 integer, dimension(:), allocatable prg_system_mod::system_type::spatnum

Species atomic number list. A list with the atomic numbers for every species Allocation:

```
spatnum(nsp)
```

Definition at line 172 of file prg_system_mod.F90.

10.14.2.13 integer, dimension(:), allocatable prg_system_mod::system_type::spindex

Species index. It gives the species index of a particular atom. Allocation:

```
spindex(nats)
```

If we need the index of atom 30 then:

```
system%spindex(30)
```

Definition at line 157 of file prg_system_mod.F90.

10.14.2.14 character(2), dimension(:), allocatable prg_system_mod::system_type::splist

Species symbol list. A list with the different species e.g. H, C, N, etc with the order corresponding to the appearance in systemsymbol. Allocation:

```
splist(nsp)
```

Definition at line 165 of file prg_system_mod.F90.

10.14.2.15 real(dp), dimension(:), allocatable prg_system_mod::system_type::spmass

Species mass list. A list with the atomic mass for every species Allocation:

```
spmass(nsp)
```

Definition at line 178 of file prg_system_mod.F90.

10.14.2.16 `character(2), dimension(:), allocatable prg_system_mod::system_type::symbol`

Chemical Symbols for every atom of the system. Symbol can be recovered using ptable module and calling the following routine:

```
system%symbol(i) = element_symbol(system%atomic_number(i))
```

Allocation:

```
symbol(nats)
```

Definition at line 86 of file prg_system_mod.F90.

10.14.2.17 `real(dp), dimension(:), allocatable prg_system_mod::system_type::userdef`

User define field.

Definition at line 181 of file prg_system_mod.F90.

10.14.2.18 `real(dp), dimension(:, :), allocatable prg_system_mod::system_type::velocity`

Velocities for every atom in the system. Allocation:

```
velocity(3, nats)
```

Definition at line 99 of file prg_system_mod.F90.

10.14.2.19 `real(dp) prg_system_mod::system_type::volk`

Volume of the system (direct space).

Note

use prg_get_recip_vects in coulomb_latte_mod to compute this.

Definition at line 142 of file prg_system_mod.F90.

10.14.2.20 `real(dp) prg_system_mod::system_type::volr`

Volume of the system (direct space).

Note

use prg_get_recip_vects in coulomb_latte_mod to compute this.

Definition at line 138 of file prg_system_mod.F90.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_system_mod.F90](#)

10.15 prg_timer_mod::timer_status_t Type Reference

Timer status type.

Private Attributes

- character(len=20) [tname](#)
Timer name.
- integer [tstart](#)
Start time.
- integer [ttotal](#)
Current total time.
- integer [tcount](#)
Current call count.
- integer [minrank](#)
Rank with min value.
- integer [maxrank](#)
Rank with max value.
- real(dp) [tsum](#)
Sum time - total time in secs.
- real(dp) [minvalue](#)
Minimum value over all ranks.
- real(dp) [maxvalue](#)
Maximum value over all ranks.
- real(dp) [tavg](#)
Average value over all ranks.
- real(dp) [tstdev](#)
Stdev across all ranks.
- real(dp) [tpercent](#)
Percent of time across all timers.

10.15.1 Detailed Description

Timer status type.

Definition at line 54 of file prg_timer_mod.F90.

10.15.2 Member Data Documentation

10.15.2.1 integer prg_timer_mod::timer_status_t::maxrank [private]

Rank with max value.

Definition at line 72 of file prg_timer_mod.F90.

10.15.2.2 `real(dp) prg_timer_mod::timer_status_t::maxvalue` `[private]`

Maximum value over all ranks.

Definition at line 81 of file `prg_timer_mod.F90`.

10.15.2.3 `integer prg_timer_mod::timer_status_t::minrank` `[private]`

Rank with min value.

Definition at line 69 of file `prg_timer_mod.F90`.

10.15.2.4 `real(dp) prg_timer_mod::timer_status_t::minvalue` `[private]`

Minimum value over all ranks.

Definition at line 78 of file `prg_timer_mod.F90`.

10.15.2.5 `real(dp) prg_timer_mod::timer_status_t::tavg` `[private]`

Average value over all ranks.

Definition at line 84 of file `prg_timer_mod.F90`.

10.15.2.6 `integer prg_timer_mod::timer_status_t::tcount` `[private]`

Current call count.

Definition at line 66 of file `prg_timer_mod.F90`.

10.15.2.7 `character(len=20) prg_timer_mod::timer_status_t::tname` `[private]`

Timer name.

Definition at line 57 of file `prg_timer_mod.F90`.

10.15.2.8 `real(dp) prg_timer_mod::timer_status_t::tpercent` `[private]`

Percent of time across all timers.

Definition at line 90 of file `prg_timer_mod.F90`.

10.15.2.9 `integer prg_timer_mod::timer_status_t::tstart` `[private]`

Start time.

Definition at line 60 of file `prg_timer_mod.F90`.

10.15.2.10 `real(dp) prg_timer_mod::timer_status_t::tstdev` `[private]`

Stdev across all ranks.

Definition at line 87 of file `prg_timer_mod.F90`.

10.15.2.11 `real(dp) prg_timer_mod::timer_status_t::tsum` `[private]`

Sum time - total time in secs.

Definition at line 75 of file `prg_timer_mod.F90`.

10.15.2.12 `integer prg_timer_mod::timer_status_t::ttotal` `[private]`

Current total time.

Definition at line 63 of file `prg_timer_mod.F90`.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_timer_mod.F90](#)

10.16 prg_extras_mod::to_string Interface Reference

Private Member Functions

- `character(len=:)` function, allocatable [to_string_integer](#) (i)
Convert integer to string.
- `character(len=:)` function, allocatable [to_string_long_long](#) (i)
Convert integer to string.
- `character(len=:)` function, allocatable [to_string_double](#) (x)
Convert double to string.

10.16.1 Detailed Description

Definition at line 25 of file `prg_extras_mod.F90`.

10.16.2 Member Function/Subroutine Documentation

10.16.2.1 `character(len=:)` function, allocatable `prg_extras_mod::to_string::to_string_double` (`double precision, intent(in) x`)
`[private]`

Convert double to string.

Parameters

<i>x</i>	The double
----------	------------

Returns

The string

Definition at line 80 of file prg_extras_mod.F90.

10.16.2.2 `character(len=:)` function, allocatable `prg_extras_mod::to_string::to_string_integer (integer, intent(in) i)`
`[private]`

Convert integer to string.

Parameters

<i>i</i>	The integer
----------	-------------

Returns

The string

Definition at line 46 of file prg_extras_mod.F90.

10.16.2.3 `character(len=:)` function, allocatable `prg_extras_mod::to_string::to_string_long_long (integer(kind=c_long_long), intent(in) i)`
`[private]`

Convert integer to string.

Parameters

<i>i</i>	The integer
----------	-------------

Returns

The string

Definition at line 62 of file prg_extras_mod.F90.

The documentation for this interface was generated from the following file:

- [/home/christian/qmd-progress/src/prg_extras_mod.F90](#)

10.17 prg_xlbo_mod::xlbo_type Type Reference

General xlbo solver type.

Public Attributes

- character(20) `jobname`
- integer `verbose`
- integer `maxscfiter`
Max SCF iterations at every XLBO MD step.
- integer `maxscfinititer`
Max SCF iterations for the first minit steps.
- real(dp) `threshold`
- integer `minit`
Use SCF the first M_prg_init MD steps.
- real(dp) `cc`
Scaled prg_delta Kernel.

10.17.1 Detailed Description

General xlbo solver type.

Definition at line 33 of file prg_xlbo_mod.F90.

10.17.2 Member Data Documentation

10.17.2.1 real(dp) prg_xlbo_mod::xlbo_type::cc

Scaled prg_delta Kernel.

Definition at line 51 of file prg_xlbo_mod.F90.

10.17.2.2 character(20) prg_xlbo_mod::xlbo_type::jobname

Definition at line 35 of file prg_xlbo_mod.F90.

10.17.2.3 integer prg_xlbo_mod::xlbo_type::maxscfinititer

Max SCF iterations for the first minit steps.

Definition at line 43 of file prg_xlbo_mod.F90.

10.17.2.4 integer prg_xlbo_mod::xlbo_type::maxscfiter

Max SCF iterations at every XLBO MD step.

Definition at line 40 of file prg_xlbo_mod.F90.

10.17.2.5 integer prg_xlbo_mod::xlbo_type::init

Use SCF the first M_prg_init MD steps.

Definition at line 48 of file prg_xlbo_mod.F90.

10.17.2.6 real(dp) prg_xlbo_mod::xlbo_type::threshold

Definition at line 45 of file prg_xlbo_mod.F90.

10.17.2.7 integer prg_xlbo_mod::xlbo_type::verbose

Definition at line 37 of file prg_xlbo_mod.F90.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_xlbo_mod.F90](#)

10.18 prg_xlkernel_mod::xk_type Type Reference

Public Attributes

- character(20) [kerneltype](#)
Kernel type.
- integer [verbose](#)
Verbosity level.
- integer [nrank](#)
- real(dp) [scalecoeff](#)
Coefficient for mixing.

10.18.1 Detailed Description

Definition at line 17 of file prg_xlkernel_mod.F90.

10.18.2 Member Data Documentation

10.18.2.1 character(20) prg_xlkernel_mod::xk_type::kerneltype

Kernel type.

Definition at line 20 of file prg_xlkernel_mod.F90.

10.18.2.2 integer prg_xlkernel_mod::xk_type::nrank

Definition at line 23 of file prg_xlkernel_mod.F90.

10.18.2.3 real(dp) prg_xlkernel_mod::xlk_type::scalecoeff

Coefficient for mixing.

Definition at line 26 of file prg_xlkernel_mod.F90.

10.18.2.4 integer prg_xlkernel_mod::xlk_type::verbose

Verbosity level.

Definition at line 23 of file prg_xlkernel_mod.F90.

The documentation for this type was generated from the following file:

- [/home/christian/qmd-progress/src/prg_xlkernel_mod.F90](#)

Chapter 11

File Documentation

11.1 /home/christian/qmd-progress/README.md File Reference

11.2 /home/christian/qmd-progress/tests/README.md File Reference

11.3 /home/christian/qmd-progress/src/prg_charges_mod.F90 File Reference

Modules

- module [prg_charges_mod](#)
A module to compute the Mulliken charges of a chemical system.

Functions/Subroutines

- subroutine, public [prg_charges_mod::prg_get_charges](#) (rho_bml, over_bml, hindex, charges, numel, spindex, mdimin, threshold)
Constructs the charges from the density matrix.
- subroutine, public [prg_charges_mod::prg_get_hscf](#) (ham0_bml, over_bml, ham_bml, spindex, hindex, hubbard, charges, coulomb_pot_r, coulomb_pot_k, mdimin, threshold)
Constructs the SCF hamiltonian given H0, HubbardU and charges. This routine does: $H = \sum_i U_i q_i + V_i$, where U is the Hubbard parameter for every atom i . V is the coulombic potential for every atom i .

Variables

- integer, parameter [prg_charges_mod::dp](#) = kind(1.0d0)

11.4 /home/christian/qmd-progress/src/prg_chebyshev_mod.F90 File Reference

Data Types

- type [prg_chebyshev_mod::chebdata_type](#)
General Cheb solver type.

Modules

- module [prg_chebyshev_mod](#)

Module to obtain the density matrix by applying a Chebyshev polynomial expansion.

Functions/Subroutines

- subroutine, public [prg_chebyshev_mod::prg_parse_cheb](#) (chebdata, filename)
Chebyshev parser. This module is used to parse all the input variables for the cheb electronic structure solver. Adding a new input keyword to the parser:
- subroutine, public [prg_chebyshev_mod::prg_build_density_cheb](#) (ham_bml, rho_bml, athr, threshold, ncoeffs, kbt, ef, bndfil, jon, verbose)
Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion.
- subroutine, public [prg_chebyshev_mod::prg_build_density_cheb_fermi](#) (ham_bml, rho_bml, athr, threshold, ncoeffs, kbt, ef, bndfil, getef, fermitol, jon, npts, trkfunc, verbose)
Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion. In this case the self-consistent recursion is applied to converge to the correct number of electrons and obtain the Fermi level.
- real(dp) function [prg_chebyshev_mod::jackson](#) (ncoeffs, i, jon)
Evaluates the Jackson Kernel Coefficients.
- subroutine [prg_chebyshev_mod::prg_get_chebcoeffs](#) (npts, kbt, ef, ncoeffs, coeffs, emin, emax)
Gets the coefficients of the Chebyshev expansion.
- subroutine [prg_chebyshev_mod::prg_get_chebcoeffs_fermi_bs](#) (npts, kbt, ef, tracesT, ncoeffs, coeffs, emin, emax, bndfil, norb, tol, jon, verbose)
Gets the coefficients of the Chebyshev expansion with E_f computation.
- subroutine [prg_chebyshev_mod::prg_get_chebcoeffs_fermi_nt](#) (npts, kbt, ef, tracesT, ncoeffs, coeffs, emin, emax, bndfil, norb, tol, jon, verbose)
Gets the coefficients of the Chebyshev expansion with E_f computation.
- real(dp) function [prg_chebyshev_mod::tr](#) (r, x)
Chebyshev polynomial obtained by recursion.
- real(dp) function [prg_chebyshev_mod::fermi](#) (e, ef, kbt)
Gives the Fermi distribution value for energy e.
- real(dp) function [prg_chebyshev_mod::absmaxderivative](#) (func, de)
Gets the absolute maximum of the derivative of a function.

Variables

- integer, parameter [prg_chebyshev_mod::dp](#) = kind(1.0d0)
- real(dp), parameter [prg_chebyshev_mod::pi](#) = 3.14159265358979323846264338327950_dp

11.5 /home/christian/qmd-progress/src/prg_densitymatrix_mod.F90 File Reference

Modules

- module [prg_densitymatrix_mod](#)

Module to obtain the density matrix by diagonalizing an prg_orthogonalized Hamiltonian.

Functions/Subroutines

- subroutine, public `prg_densitymatrix_mod::prg_build_density_t0` (ham_bml, rho_bml, threshold, bndfil)
Builds the density matrix from H_0 for zero electronic temperature. $\rho = C\Theta(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. $\Theta()$ is the Heaviside function.
- subroutine, public `prg_densitymatrix_mod::prg_build_density_t` (ham_bml, rho_bml, threshold, bndfil, kbt, ef)
Builds the density matrix from H_0 for electronic temperature T . $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function.
- subroutine, public `prg_densitymatrix_mod::prg_build_density_t_fermi` (ham_bml, rho_bml, threshold, kbt, ef, verbose)
Builds the density matrix from H_0 for electronic temperature T . $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function. In this routine the Fermi level is passed as an argument.
- subroutine, public `prg_densitymatrix_mod::prg_build_atomic_density` (rhoat_bml, numel, hindex, spindex, norb, bml_type)
Builds the atomic density matrix. $\rho_{ii} = \text{mathcal{Z}}_{ii}$ Where, $\text{mathcal{Z}}_{ii}$ is the number of electrons for orbital i .
- subroutine, public `prg_densitymatrix_mod::prg_get_flevel` (eigenvalues, kbt, bndfil, tol, Ef)
Routine to compute the Fermi level given a set of eigenvalues and a temperature. It applies the Bisection method over the function: $g(\mu) = \sum_k 2f(\epsilon_k - \mu) - N = 0$ Where $f(\epsilon_k - \mu) = \frac{1}{1 + \exp((\epsilon_k - \mu)/(k_b T))}$.
- subroutine, public `prg_densitymatrix_mod::prg_get_eigenvalues` (ham_bml, eigenvalues, verbose)
Gets the eigenvalues of the Orthogonalized Hamiltonian.
- subroutine, public `prg_densitymatrix_mod::prg_check_idempotency` (mat_bml, threshold, idempotency)
To check the idempotency error of a matrix. This is calculated as the Frobenius norm of $(A - A^2)$.
- real(dp) function `prg_densitymatrix_mod::fermi` (e, ef, kbt)
Gives the Fermi distribution value for energy e .

Variables

- integer, parameter `prg_densitymatrix_mod::dp` = kind(1.0d0)

11.6 /home/christian/qmd-progress/src/prg_dos_mod.F90 File Reference

Modules

- module `prg_dos_mod`
A module to compute the Density of state (DOS) and IDOS.

Functions/Subroutines

- subroutine, public `prg_dos_mod::prg_write_tdos` (eigenvals, gamma, npts, emin, emax, filename)
Writes the total DOS into a file. $DOS(\epsilon) = \sum_k L(\epsilon - \epsilon_k)$ Where $\int_{-\infty}^{\infty} DOS(\epsilon) = Nstates$.
- real(dp) function `prg_dos_mod::lorentz` (energy, eigenvals, loads, Gamma)
Lorentzian Function.

Variables

- integer, parameter `prg_dos_mod::dp` = kind(1.0d0)

11.7 /home/christian/qmd-progress/src/prg_doxy_mod.F90 File Reference

11.8 /home/christian/qmd-progress/src/prg_extras_mod.F90 File Reference

Data Types

- interface [prg_extras_mod::prg_memory_consumption](#)
- interface [prg_extras_mod::to_string](#)

Modules

- module [prg_extras_mod](#)

Extra routines:

Functions/Subroutines

- character(len=:) function, allocatable [prg_extras_mod::to_string_integer](#) (i)
Convert integer to string.
- character(len=:) function, allocatable [prg_extras_mod::to_string_long_long](#) (i)
Convert integer to string.
- character(len=:) function, allocatable [prg_extras_mod::to_string_double](#) (x)
Convert double to string.
- subroutine, public [prg_extras_mod::prg_print_matrix](#) (matname, amat, i1, i2, j1, j2)
To write a dense matrix to screen.
- real(dp) function, public [prg_extras_mod::mls](#) ()
To get the actual time in milliseconds.
- subroutine, public [prg_extras_mod::prg_delta](#) (x, s, nn, dta)
Delta function $\|X^tSX - I\|$. CFAN, March 2015.
- subroutine, public [prg_extras_mod::prg_get_mem](#) (procname, tag)
Get proc memory.
- subroutine [prg_extras_mod::prg_twonorm](#) (a, nn, norm2)

Variables

- integer, parameter [prg_extras_mod::dp](#) = kind(1.0d0)

11.9 /home/christian/qmd-progress/src/prg_genz_mod.F90 File Reference

Data Types

- type [prg_genz_mod::genzspinp](#)
Input for the genz driver.
- type [prg_genz_mod::genzspdata](#)
contains the data for the genZ driver.

Modules

- module [prg_genz_mod](#)

To produce a matrix Z which is needed to orthogonalize H .

Functions/Subroutines

- subroutine, public [prg_genz_mod::prg_parse_zsp](#) (input, filename)

The parser for genz solver.

- subroutine [prg_genz_mod::prg_init](#) (self, input)

Initializes the genz input variables.

- subroutine [prg_genz_mod::prg_allocatezspmat](#) (self, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, norb, bml_type)

Allocates the matrices for the XI integration of Z .

- subroutine, public [prg_genz_mod::prg_init_zspmat](#) (igenz, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, norb, bml_type)

Initiates the matrices for the XI integration of Z .

- subroutine [prg_genz_mod::prg_generate](#) (self, over_bml, zmat_bml, igenz, mdim, bml_type, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml)

Generates the Z matrix.

- subroutine, public [prg_genz_mod::prg_buildzdiag](#) (smat_bml, zmat_bml, threshold, mdimin, bml_type, verbose)

Usual subroutine involving diagonalization.

- subroutine, public [prg_genz_mod::prg_buildzsparse](#) (smat_bml, zmat_bml, igenz, mdim, bml_type, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, nfirst, nrefi, nreff, thresholdi, thresholdf, integration, verbose)

Inverse factorization using niklasson's algorithm.

- subroutine, public [prg_genz_mod::prg_genz_sp_initialz0](#) (smat_bml, zmat_bml, norb, mdim, bml_type, f, threshold)

- subroutine, public [prg_genz_mod::prg_genz_sp_initial_zmat](#) (smat_bml, zmat_bml, norb, mdim, bml_type, f, threshold)

Estimate Z matrix.

- subroutine [prg_genz_mod::prg_genz_sp_int](#) (zmat_bml, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, igenz, norb, bml_type, threshold)

- subroutine, public [prg_genz_mod::prg_genz_sp_ref](#) (smat_bml, zmat_bml, nref, norb, bml_type, threshold)

Variables

- integer, parameter [prg_genz_mod::dp](#) = kind(1.0d0)

11.10 /home/christian/qmd-progress/src/prg_graph_mod.F90 File Reference

Data Types

- type [prg_graph_mod::subgraph_t](#)

Subgraph type.

- type [prg_graph_mod::graph_partitioning_t](#)

Trace per iteration.

Modules

- module [prg_graph_mod](#)
The graph module.

Functions/Subroutines

- subroutine, public [prg_graph_mod::prg_initsubgraph](#) (sg, pnum, hsize)
Initialize subgraph.
- subroutine, public [prg_graph_mod::prg_destroysubgraph](#) (sg)
Destroy subgraph.
- subroutine, public [prg_graph_mod::prg_initgraphpartitioning](#) (gp, pname, np, nnodes, nnodes2)
Initialize graph partitioning.
- subroutine, public [prg_graph_mod::prg_destroygraphpartitioning](#) (gp)
Destroy graph partitioning.
- subroutine, public [prg_graph_mod::prg_printgraphpartitioning](#) (gp)
Print graph partitioning structure data.
- subroutine, public [prg_graph_mod::prg_equalpartition](#) (gp, nodesPerPart, nnodes)
Create equal graph partitions, based on number of rows/orbitals.
- subroutine, public [prg_graph_mod::prg_equalitygrouppartitioning](#) (gp, hindex, ngroup, nodesPerPart, nnodes)
Create equal group graph partitions, based on number of atoms/groups.
- subroutine, public [prg_graph_mod::prg_filepartition](#) (gp, partFile)
Read graph partitions from a file, based on number of rows/orbitals.
- subroutine [prg_graph_mod::prg_readpart](#) (gp, partFile)
Read parts (core) from part file.
- subroutine, public [prg_graph_mod::prg_fnormgraph](#) (gp)
Accumulate trace norm across all subgraphs.

Variables

- integer, parameter [prg_graph_mod::dp](#) = kind(1.0d0)

11.11 /home/christian/qmd-progress/src/prg_graphsp2parser_mod.F90 File Reference

Data Types

- type [prg_graphsp2parser_mod::gsp2data_type](#)
General SP2 solver type.

Modules

- module [prg_graphsp2parser_mod](#)
Graph partitioning SP2 parser.

Functions/Subroutines

- subroutine, public [prg_graphsp2parser_mod::prg_parse_gsp2](#) (gsp2data, filename)
The parser for SP2 solver.

Variables

- integer, parameter `prg_graphsp2parser_mod::dp` = kind(1.0d0)

11.12 /home/christian/qmd-progress/src/prg_homolumo_mod.F90 File Reference

Modules

- module `prg_homolumo_mod`

The homolumo module.

Functions/Subroutines

- subroutine, public `prg_homolumo_mod::prg_homolumogap` (vv, imax, pp, mineval, maxeval, ehomo, elumo, egap, verbose)
- subroutine, public `prg_homolumo_mod::prg_sp2sequence` (pp, imax, mineval, maxeval, ehomo, elumo, erlimit, verbose)

Variables

- integer, parameter `prg_homolumo_mod::dp` = kind(1.0d0)

11.13 /home/christian/qmd-progress/src/prg_implicit_fermi_mod.F90 File Reference

Modules

- module `prg_implicit_fermi_mod`

Functions/Subroutines

- subroutine, public `prg_implicit_fermi_mod::prg_implicit_fermi` (h_bml, xi0_bml, p_bml, nsteps, nocc, mu, beta, osteps, occErrLimit, threshold)

Recursive Implicit Fermi Dirac.

Variables

- integer, parameter `prg_implicit_fermi_mod::dp` = kind(1.0d0)

11.14 /home/christian/qmd-progress/src/prg_initmatrices_mod.F90 File Reference

Modules

- module `prg_initmatrices_mod`

Initialization module.

Functions/Subroutines

- subroutine, public [prg_initmatrices_mod::prg_init_hsmat](#) (ham_bml, over_bml, bml_type, mdim, norb)
Initialize Hamiltonian and Overlap Matrix.
- subroutine, public [prg_initmatrices_mod::prg_init_pzmat](#) (rho_bml, zmat_bml, bml_type, mdim, norb)
Initialize Density matrix and Inverse square root Overlap.
- subroutine, public [prg_initmatrices_mod::prg_init_ortho](#) (orthoh_bml, orthop_bml, bml_type, mdim, norb)
Initialize The orthogonal versions of Hamiltonian and Density Matrix.

Variables

- integer, parameter [prg_initmatrices_mod::dp](#) = kind(1.0d0)

11.15 /home/christian/qmd-progress/src/prg_kernelparser_mod.F90 File Reference

Modules

- module [prg_kernelparser_mod](#)
Some general parsing functions.

Functions/Subroutines

- subroutine, public [prg_kernelparser_mod::prg_parsing_kernel](#) (keyvector_char, valvector_char, keyvector_↵int, valvector_int, keyvector_re, valvector_re, keyvector_log, valvector_log, filename, startstop)
The general parsing function. It is used to vectorize a set of "keywords" "value" pairs as included in a general input file.
- subroutine [prg_kernelparser_mod::prg_check_valid](#) (invalidc)
Check for valid keywords (checks for an = sign)

Variables

- integer, parameter [prg_kernelparser_mod::dp](#) = kind(1.0d0)

11.16 /home/christian/qmd-progress/src/prg_nonortho_mod.F90 File Reference

Modules

- module [prg_nonortho_mod](#)
Module to prg_orthogonalize and prg_deorthogonalize any operator.

Functions/Subroutines

- subroutine, public [prg_nonortho_mod::prg_orthogonalize](#) (A_bml, zmat_bml, orthoA_bml, threshold, bml_↵type, verbose)
This routine performs: $A_{ortho} = Z^\dagger A Z$.
- subroutine, public [prg_nonortho_mod::prg_deorthogonalize](#) (orthoA_bml, zmat_bml, a_bml, threshold, bml_↵_type, verbose)
This routine performs: $A = Z A_{ortho} Z^\dagger$.

Variables

- integer, parameter `prg_nonortho_mod::dp` = kind(1.0d0)

11.17 /home/christian/qmd-progress/src/prg_normalize_mod.F90 File Reference

Modules

- module `prg_normalize_mod`
The prg_normalize module.

Functions/Subroutines

- subroutine, public `prg_normalize_mod::prg_normalize` (h_bml)
Normalize a Hamiltonian matrix prior to running the SP2 algorithm.
- subroutine, public `prg_normalize_mod::prg_normalize_fermi` (h_bml, h1, hN, mu)
Normalize a Hamiltonian matrix prior to running the truncated SP2 algorithm.
- subroutine, public `prg_normalize_mod::prg_normalize_implicit_fermi` (h_bml, cnst, mu)
Normalize a Hamiltonian matrix prior to running the implicit fermi dirac algorithm.
- subroutine, public `prg_normalize_mod::prg_gershgorinreduction` (gp)
Determine gershgorin bounds across all parts, local and distributed.
- subroutine, public `prg_normalize_mod::prg_normalize_cheb` (h_bml, mu, emin, emax, alpha, scaledmu)
Normalize a Hamiltonian matrix prior to running the Chebyshev algorithm.

Variables

- integer, parameter `prg_normalize_mod::dp` = kind(1.0d0)

11.18 /home/christian/qmd-progress/src/prg_openfiles_mod.F90 File Reference

Modules

- module `prg_openfiles_mod`
Module to handle input output files for the PROGRESS lib.

Functions/Subroutines

- integer function, public `prg_openfiles_mod::get_file_unit` (io_max)
Returns a unit number that is not in use.
- subroutine, public `prg_openfiles_mod::prg_open_file` (io, name)
Opens a file to write.
- subroutine, public `prg_openfiles_mod::prg_open_file_to_read` (io, name)
Opens a file to read.

11.19 /home/christian/qmd-progress/src/prg_parallel_mod.F90 File Reference

Data Types

- type [prg_parallel_mod::rankreducedata_t](#)
Data structure for rection over MPI ranks.

Modules

- module [prg_parallel_mod](#)
The parallel module.

Functions/Subroutines

- integer function, public [prg_parallel_mod::getnranks](#) ()
- integer function, public [prg_parallel_mod::getmyrank](#) ()
- integer function, public [prg_parallel_mod::printrank](#) ()
- subroutine, public [prg_parallel_mod::prg_initparallel](#) ()
- subroutine, public [prg_parallel_mod::prg_shutdownparallel](#) ()
- integer function [prg_parallel_mod::saverequest](#) (irequest)
- subroutine, public [prg_parallel_mod::prg_barrierparallel](#) ()
- subroutine, public [prg_parallel_mod::sendreceiveparallel](#) (sendBuf, sendLen, dest, recvBuf, recvLen, source, nreceived)
- subroutine, public [prg_parallel_mod::isendparallel](#) (sendBuf, sendLen, dest)
- subroutine, public [prg_parallel_mod::sendparallel](#) (sendBuf, sendLen, dest)
- subroutine, public [prg_parallel_mod::prg_iprg_recvparallel](#) (recvBuf, recvLen, rind)
- subroutine, public [prg_parallel_mod::prg_recvparallel](#) (recvBuf, recvLen)
- subroutine, public [prg_parallel_mod::sumintparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [prg_parallel_mod::sumrealparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [prg_parallel_mod::maxintparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [prg_parallel_mod::maxrealparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [prg_parallel_mod::minintparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [prg_parallel_mod::minrealparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [prg_parallel_mod::prg_minrealreduce](#) (rvalue)
- subroutine, public [prg_parallel_mod::prg_maxrealreduce](#) (rvalue)
- subroutine, public [prg_parallel_mod::prg_maxintreduce2](#) (value1, value2)
- subroutine, public [prg_parallel_mod::prg_sumintreduce2](#) (value1, value2)
- subroutine, public [prg_parallel_mod::prg_sumrealreduce](#) (value1)
- subroutine, public [prg_parallel_mod::prg_sumrealreduce2](#) (value1, value2)
- subroutine, public [prg_parallel_mod::prg_sumrealreduce3](#) (value1, value2, value3)
- subroutine, public [prg_parallel_mod::prg_sumrealreducen](#) (valueVec, N)
- subroutine, public [prg_parallel_mod::prg_sumintreducen](#) (valueVec, N)
- subroutine, public [prg_parallel_mod::minrankrealparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [prg_parallel_mod::maxrankrealparallel](#) (sendBuf, recvBuf, icount)
- subroutine, public [prg_parallel_mod::prg_bcastparallel](#) (buf, blen, root)
- subroutine, public [prg_parallel_mod::allgatherrealparallel](#) (sendBuf, sendLen, recvBuf, recvLen)
- subroutine, public [prg_parallel_mod::allgatherintparallel](#) (sendBuf, sendLen, recvBuf, recvLen)
- subroutine, public [prg_parallel_mod::allgathervrealparallel](#) (sendBuf, sendLen, recvBuf, recvLen, recvDispl)
- subroutine, public [prg_parallel_mod::allgathervintparallel](#) (sendBuf, sendLen, recvBuf, recvLen, recvDispl)
- subroutine, public [prg_parallel_mod::prg_allsumrealreduceparallel](#) (buf, buflen)
- subroutine, public [prg_parallel_mod::prg_allsumintreduceparallel](#) (buf, buflen)
- subroutine, public [prg_parallel_mod::prg_allgatherparallel](#) (a)
- subroutine, public [prg_parallel_mod::prg_wait](#) ()

Variables

- integer, parameter `prg_parallel_mod::dp` = kind(1.0d0)
- integer `prg_parallel_mod::myrank`
- integer `prg_parallel_mod::nranks`
- integer `prg_parallel_mod::ierr`
- integer `prg_parallel_mod::reqcount`
- integer, dimension(:), allocatable `prg_parallel_mod::requestlist`
- integer, dimension(:), allocatable `prg_parallel_mod::rused`

11.20 /home/christian/qmd-progress/src/prg_partition_mod.F90 File Reference

Modules

- module `prg_partition_mod`

The partition module.

Functions/Subroutines

- subroutine, public `prg_partition_mod::prg_metispartition` (gp, ngroups, nnodes, xadj, adjncy, nparts, part, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)
Create graph partitions minizing number of cut edges.
- subroutine, public `prg_partition_mod::prg_costpartition` (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)
Compute cost of a partition.
- subroutine, public `prg_partition_mod::update_prg_costpartition` (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, node, new_part)
Update cost of partition and the different parameters node is moves into new_part For each neighbor of node, the following cases hold: Case 1: neighbor is in old_part Case 2: neighbor is in new_part Case 3: neighbor is neither in old_ or new_part.
- subroutine `prg_partition_mod::prg_accept_prob` (it, prg_delta, r)
Compute acceptance probability for simulated annealing.
- subroutine `prg_partition_mod::prg_costindex` (cost, sumCubes, maxCH, smooth_maxCH, obj_fun)
Choose objective function to work with.
- subroutine `prg_partition_mod::prg_rand_node` (gp, node, seed)
Pick a random node.
- subroutine, public `prg_partition_mod::prg_simannealing` (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, niter, seed)
Graph partitioning based on Simulated Annealing.
- subroutine, public `prg_partition_mod::prg_kernlin` (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, nconverg, seed)
Graph partitioning based on inspired by Kernighan-Lin Review METIS manual for description of k-way implementation of KL Pick a core together with its halos Place free vertices on a priority queue with (key, value) =(prg_delta, best_↔ part),with prg_delta = change in obj_value Dequeue and allow hill climbing.
- subroutine, public `prg_partition_mod::prg_update_gp` (gp, partNumber, core_count)
- subroutine `prg_partition_mod::prg_rand_shuffle` (array, seed)
Randomly shuffle array.
- subroutine, public `prg_partition_mod::prg_check_arrays` (gp, core_count, CH_count, Halo_count)
Error checking Checking that core_count, CH_count, Halo_count match.
- subroutine, public `prg_partition_mod::prg_kernlin_queue` (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)

Greedy algorithm. At each step it chooses the (vertex, new_part) pair with highest gain. Currently implementation is very slow.

- subroutine [prg_partition_mod::prg_find_best_move](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, best_node, best_part)

For kerlin_queue to find (vertex, new_part) pair with highest gain.

- subroutine, public [prg_partition_mod::prg_kernlin2](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)
- subroutine [prg_partition_mod::prg_get_largest_hedge_in_part](#) (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, search_part, largest_Hedge)
- subroutine, public [prg_partition_mod::prg_simannealing_old](#) (gp, xadj, adjncy, partNumber, core_count, C↔H_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, niter, seed)

Variables

- integer, parameter [prg_partition_mod::dp](#) = kind(1.0d0)
- integer, parameter [prg_partition_mod::metis_index_kind](#) = METIS_INDEX_KIND
From /usr/include/metis.h.
- integer, parameter [prg_partition_mod::metis_real_kind](#) = kind(METIS_REAL_KIND)
From /usr/include/metis.h.

11.21 /home/christian/qmd-progress/src/prg_progress_mod.F90 File Reference

Modules

- module [prg_progress_mod](#)
The progress module.

Functions/Subroutines

- subroutine, public [prg_progress_mod::prg_progress_init](#) ()
Initialize progress.
- subroutine, public [prg_progress_mod::prg_progress_shutdown](#) ()
Shutdown progress.

Variables

- integer, parameter [prg_progress_mod::dp](#) = kind(1.0d0)

11.22 /home/christian/qmd-progress/src/prg_ptable_mod.F90 File Reference

Modules

- module [prg_ptable_mod](#)
Periodic table of elements.

Functions/Subroutines

- integer function, public [prg_ptable_mod::element_atomic_number](#) (symbol)
- integer function [prg_ptable_mod::element_atomic_number_upper](#) (symbol)

Variables

- integer, parameter [prg_ptable_mod::nz](#) = 103
- integer, parameter, private [prg_ptable_mod::dp](#) = kind(1.0d0)
- character(2), dimension(nz), parameter [prg_ptable_mod::element_symbol](#) = [character(2) :: "H", "He", "Li", "Be", "B", "C", "N", "O", "F", "Ne", "Na", "Mg", "Al", "Si", "P", "S", "Cl", "Ar", "K", "Ca", "Sc", "Ti", "V", "Cr", "Mn", "Fe", "Co", "Ni", "Cu", "Zn", "Ga", "Ge", "As", "Se", "Br", "Kr", "Rb", "Sr", "Y", "Zr", "Nb", "Mo", "Tc", "Ru", "Rh", "Pd", "Ag", "Cd", "In", "Sn", "Sb", "Te", "I", "Xe", "Cs", "Ba", "La", "Ce", "Pr", "Nd", "Pm", "Sm", "Eu", "Gd", "Tb", "Dy", "Ho", "Er", "Tm", "Yb", "Lu", "Hf", "Ta", "W", "Re", "Os", "Ir", "Pt", "Au", "Hg", "Tl", "Pb", "Bi", "Po", "At", "Rn", "Fr", "Ra", "Ac", "Th", "Pa", "U", "Np", "Pu", "Am", "Cm", "Bk", "Cf", "Es", "Fm", "Md", "No", "Lr"]

Element symbol.

- character(2), dimension(nz), parameter [prg_ptable_mod::element_symbol_upper](#) = [character(2) :: "H", "HE", "LI", "BE", "B", "C", "N", "O", "F", "NE", "NA", "MG", "AL", "SI", "P", "S", "CL", "AR", "K", "CA", "SC", "TI", "V", "CR", "MN", "FE", "CO", "NI", "CU", "ZN", "GA", "GE", "AS", "SE", "BR", "KR", "RB", "SR", "Y", "ZR", "NB", "MO", "TC", "RU", "RH", "PD", "AG", "CD", "IN", "SN", "SB", "TE", "I", "XE", "CS", "BA", "LA", "CE", "PR", "ND", "PM", "SM", "EU", "GD", "TB", "DY", "HO", "ER", "TM", "YB", "LU", "HF", "TA", "W", "RE", "OS", "IR", "PT", "AU", "HG", "TL", "PB", "BI", "PO", "AT", "RN", "FR", "RA", "AC", "TH", "PA", "U", "NP", "PU", "AM", "CM", "BK", "CF", "ES", "FM", "MD", "NO", "LR"]

Element symbol upper.

- character(20), dimension(nz), parameter [prg_ptable_mod::element_name](#) = [character(20) :: "Hydrogen", "Helium", "Lithium", "Beryllium", "Boron", "Carbon", "Nitrogen", "Oxygen", "Fluorine", "Neon", "Sodium", "Magnesium", "Aluminium", "Silicon", "Phosphorus", "Sulfur", "Chlorine", "Argon", "Potassium", "Calcium", "Scandium", "Titanium", "Vanadium", "Chromium", "Manganese", "Iron", "Cobalt", "Nickel", "Copper", "Zinc", "Gallium", "Germanium", "Arsenic", "Selenium", "Bromine", "Krypton", "Rubidium", "Strontium", "Yttrium", "Zirconium", "Niobium", "Molybdenum", "Technetium", "Ruthenium", "Rhodium", "Palladium", "Silver", "Cadmium", "Indium", "Tin", "Antimony", "Tellurium", "Iodine", "Xenon", "Caesium", "Barium", "Lanthanum", "Cerium", "Praseodymium", "Neodymium", "Promethium", "Samarium", "Europium", "Gadolinium", "Terbium", "Dysprosium", "Holmium", "Erbium", "Thulium", "Ytterbium", "Lutetium", "Hafnium", "Tantalum", "Tungsten", "Rhenium", "Osmium", "Iridium", "Platinum", "Gold", "Mercury", "Thallium", "Lead", "Bismuth", "Polonium", "Astatine", "Radon", "Francium", "Radium", "Actinium", "Thorium", "Protactinium", "Uranium", "Neptunium", "Plutonium", "Americium", "Curium", "Berkelium", "Californium", "Einsteinium", "Fermium", "Mendelevium", "Nobelium", "Lawrencium"]

Element name.

- real(dp), dimension(nz), parameter [prg_ptable_mod::element_mass](#) = (/ 1.007825032, 4.002603254, 7.01600455, 9.0121822, 11.0093054, 12.0, 14.003074005, 15.99491462, 18.99840322, 19.992440175, 22.989769281, 23.9850417, 26.98153863, 27.976926532, 30.97376163, 31.972071, 34.96885268, 39.962383123, 38.96370668, 39.96259098, 44.9559119, 47.9479463, 50.9439595, 51.9405075, 54.9380451, 55.9349375, 58.933195, 57.9353429, 62.9295975, 63.929142, 68.925573, 73.921177, 74.921596, 79.916521, 78.918337, 83.911507, 84.911789, 87.905612, 88.905848, 89.904704, 92.906378, 97.905408, 97.907216, 101.904349, 102.905504, 105.903486, 106.905097, 113.903358, 114.903878, 119.902194, 120.903815, 129.906224, 126.904473, 131.904153, 132.905451, 137.905247, 138.906353, 139.905438, 140.907652, 141.907723, 144.912749, 151.919732, 152.92123, 157.924103, 158.925346, 163.929174, 164.930322, 165.930293, 168.934213, 173.938862, 174.940771, 179.94655, 180.947995, 183.950931, 186.955753, 191.96148, 192.962926, 194.964791, 196.966568, 201.970643, 204.974427, 207.976652, 208.980398, 208.98243, 209.987148, 222.017577, 223.019735, 226.025409, 227.027752, 232.038055, 231.035884, 238.050788, 237.048173, 244.064204, 243.061381, 247.070354, 247.070307, 251.079587, 252.08298, 257.095105, 258.098431, 259.10103, 262.10963 /)

Element mass in atomic mass units (1.66 x 10⁻²⁷ kg)

- character(50), dimension(nz), parameter `prg_ptable_mod::element_econf` = [character(50) :: "1s", "1s2", "1s22s", "1s22s2", "1s22s22p", "1s22s22p2", "1s22s22p3", "1s22s22p4", "1s22s22p5", "1s22s22p6", "[Ne]3s", "[Ne]3s2", "[Ne]3s23p", "[Ne]3s23p2", "[Ne]3s23p3", "[Ne]3s23p4", "[Ne]3s23p5", "[Ne]3s23p6", "[Ar]4s", "[Ar]4s2", "[Ar]3d4s2", "[Ar]3d24s2", "[Ar]3d34s2", "[Ar]3d54s", "[Ar]3d54s2", "[Ar]3d64s2", "[Ar]3d74s2", "[Ar]3d84s2", "[Ar]3d104s", "[Ar]3d104s2", "[Ar]3d104s24p", "[Ar]3d104s24p2", "[Ar]3d104s24p3", "[Ar]3d104s24p4", "[Ar]3d104s24p5", "[Ar]3d104s24p6", "[Kr]5s", "[Kr]5s2", "[Kr]4d5s2", "[Kr]4d25s2", "[Kr]4d45s", "[Kr]4d55s", "[Kr]4d55s2", "[Kr]4d75s", "[Kr]4d85s", "[Kr]4d10", "[Kr]4d105s", "[Kr]4d105s2", "[Cd]5p", "[Cd]5p2", "[Cd]5p3", "[Cd]5p4", "[Cd]5p5", "[Cd]5p6", "[Xe]6s", "[Xe]6s2", "[Xe]5d6s2", "[Xe]4f5d6s2", "[Xe]4f36s2", "[Xe]4f46s2", "[Xe]4f56s2", "[Xe]4f66s2", "[Xe]4f76s2", "[Xe]4f75d6s2", "[Xe]4f96s2", "[Xe]4f106s2", "[Xe]4f116s2", "[Xe]4f126s2", "[Xe]4f136s2", "[Xe]4f146s2", "[Xe]4f145d6s2", "[Xe]4f145d26s2", "[Xe]4f145d36s2", "[Xe]4f145d46s2", "[Xe]4f145d56s2", "[Xe]4f145d66s2", "[Xe]4f145d76s2", "[Xe]4f145d96s", "[Xe]4f145d106s", "[Xe]4f145d106s2", "[Hg]6p", "[Hg]6p2", "[Hg]6p3", "[Hg]6p4", "[Hg]6p5", "[Hg]6p6", "[Rn]7s", "[Rn]7s2", "[Rn]6d7s2", "[Rn]6d27s2", "[Rn]5f26d7s2", "[Rn]5f36d7s2", "[Rn]5f46d7s2", "[Rn]5f67s2", "[Rn]5f77s2", "[Rn]5f76d7s2", "[Rn]5f97s2", "[Rn]5f107s2", "[Rn]5f117s2", "[Rn]5f127s2", "[Rn]5f137s2", "[Rn]5f147s2", "[Rn]5f147s27p"]

The electronic configuration.

11.23 /home/christian/qmd-progress/src/prg_pulaycomponent_mod.F90 File Reference

Modules

- module `prg_pulaycomponent_mod`
Produces a matrix to get the Pulay Component of the forces.

Functions/Subroutines

- subroutine, public `prg_pulaycomponent_mod::prg_pulaycomponent0` (rho_bml, ham_bml, pcm_bml, threshold, M, bml_type, verbose)
 $At\ T = 0K, P = \rho H \rho.$
- subroutine, public `prg_pulaycomponent_mod::prg_pulaycomponentt` (rho_bml, ham_bml, zmat_bml, pcm_bml, threshold, M, bml_type, verbose)
 $At\ T > 0K, P = \rho H S^{-1} + S^{-1} H \rho.$
- subroutine, public `prg_pulaycomponent_mod::prg_get_pulayforce` (nats, zmat_bml, ham_bml, rho_bml, dSx_bml, dSy_bml, dSz_bml, hindex, FPUL, threshold)
Pulay Force FPUL from $2Tr[Z Z' H D \frac{dS}{dR}]$.

Variables

- integer, parameter `prg_pulaycomponent_mod::dp` = kind(1.0d0)

11.24 /home/christian/qmd-progress/src/prg_pulaymixer_mod.F90 File Reference

Data Types

- type `prg_pulaymixer_mod::mx_type`

Modules

- module [prg_pulaymixer_mod](#)

Pulay mixer mode.

Functions/Subroutines

- subroutine, public [prg_pulaymixer_mod::prg_parse_mixer](#) (input, filename)
The parser for the mixer routines.
- subroutine, public [prg_pulaymixer_mod::prg_qmixer](#) (charges, oldcharges, dqin, dqout, scferror, piter, pulay-coef, mpulay, verbose)
Mixing the charges to accelerate scf convergence.
- subroutine, public [prg_pulaymixer_mod::prg_linearmixer](#) (charges, oldcharges, scferror, linmixcoef, verbose)
Routine to perform linear mixing.

Variables

- integer, parameter [prg_pulaymixer_mod::dp](#) = kind(1.0d0)

11.25 /home/christian/qmd-progress/src/prg_quantumdynamics_mod.F90 File Reference

Modules

- module [prg_quantumdynamics_mod](#)

A module to add in common quantum dynamical operations.

Functions/Subroutines

- subroutine, public [prg_quantumdynamics_mod::prg_kick_density](#) (kick_dirac, kick_mag, dens, norbs, mdim, S, SINV, which_atom, r, bmltype, thresh)
Provides perturbation to initial density matrix in the form of an electric field kick. This routine does: $\rho_{kick} = \exp \frac{-i}{\hbar} \hat{V} \hat{\rho} \hat{S} \exp \frac{i}{\hbar} \hat{V} \hat{S}^{-1}$ where \hat{V} is the field disturbance.
- subroutine, public [prg_quantumdynamics_mod::prg_get_sparsity_cplxmat](#) (matrix_type, element_type, thresh, a_dense)
This computes the sparsity of a complex matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ where f is the sparsity, N_0 is the number of values less than the threshold, and N_{tot} is the total number of values. The sparsity and threshold are printed to the screen.
- subroutine, public [prg_quantumdynamics_mod::prg_get_sparsity_realmat](#) (matrix_type, element_type, thresh, a_dense)
This computes the sparsity of a real matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ where f is the sparsity, N_0 is the number of values less than the threshold, and N_{tot} is the total number of values. The sparsity and threshold are printed to the screen.
- subroutine, public [prg_quantumdynamics_mod::prg_kick_density_bml](#) (kick_dirac, kick_mag, rho_bml, s_bml, sinv_bml, mdim, which_atom, r, matrix_type, thresh)
Provides perturbation to initial density matrix in the form of an electric field kick given input matrices in BML format. This routine does: $\rho_{kick} = \exp \frac{-i}{\hbar} \hat{V} \hat{\rho} \hat{S} \exp \frac{i}{\hbar} \hat{V} \hat{S}^{-1}$ where \hat{V} is the field disturbance.
- subroutine, public [prg_quantumdynamics_mod::prg_lvni_bml](#) (h1_bml, sinv_bml, dt, hbar, rhoold_bml, rho_bml, aux_bml, matrix_type, mdim, thresh)

Performs Liouville-von Neumann integration using leap-frog method. This routine does: $\hat{\rho}(t + \Delta t) = \hat{\rho}(t - \Delta t) + 2\Delta t \frac{\partial \hat{\rho}(t)}{\partial t}$ where the time derivative of the density matrix is defined as follows: $\frac{\partial \hat{\rho}(t)}{\partial t} = \frac{-i}{\hbar} \left(S^{-1} \hat{H}(t) \hat{\rho}(t) - \hat{\rho}(t) \hat{H}(t) S^{-1} \right)$.

- subroutine, public [prg_quantumdynamics_mod::prg_getcharge](#) (rho_bml, s_bml, charges, aux_bml, z, spin-dex, N, nats, thresh)

Constructs the charges from the density matrix.

- subroutine, public [prg_quantumdynamics_mod::prg_getdipole](#) (charges, r, mu)

This routine computes the dipole moment of the system with units determined by the units of the coordinate matrix and charges given.

- subroutine, public [prg_quantumdynamics_mod::prg_excitation](#) (fill_mat, orbit_orig, orbit_exc)

Produce an excitation in the initially calculated density matrix to.

Variables

- integer, parameter [prg_quantumdynamics_mod::dp](#) = kind(1.0d0)

11.26 /home/christian/qmd-progress/src/prg_response_mod.F90 File Reference

Data Types

- type [prg_response_mod::respdata_type](#)

Modules

- module [prg_response_mod](#)

Module to compute the density matrix response and related quantities.

Functions/Subroutines

- subroutine, public [prg_response_mod::prg_parse_response](#) (RespData, filename)

The parser for the calculation of the DM response.

- subroutine, public [prg_response_mod::prg_compute_dipole](#) (charges, coordinate, dipoleMoment, factor, verbose)

To compute the dipole moment of the system. The units of the dipole moment are determined by the units of the coordinates and charges that are given.

- subroutine, public [prg_response_mod::prg_write_dipole_tcl](#) (dipoleMoment, file, factor, verbose)

To visualize a dipole moment using VMD. This will prg_generate a .tcl script that could be run using VMD To visualize with VMD: \$ vmd -e dipole.tcl.

- subroutine, public [prg_response_mod::prg_compute_polarizability](#) (rsp_bml, prt_bml, polarizability, factor, verbose)

To compute the polarizability of the system. The units of the directional polarizability are determined by the units of the perturbation and Hamiltonian. This equation can be found in [5] equation 4a. Note that in equation 4a of the reference there is a 2 that account for the double occupancy which is not present in this case cause the density matrix construction is done by taking the occupancy into account.

- subroutine, public [prg_response_mod::prg_pert_from_file](#) (prt_bml, norb)

Read perturbation from file.

- subroutine, public [prg_response_mod::prg_compute_response_rs](#) (ham_bml, prt_bml, rsp_bml, lambda, bndfil, threshold, verbose)

Computes the first order response density matrix using Rayleigh Schrodinger Perturbation theory The transformation hereby performed are:

- subroutine, public `prg_response_mod::prg_compute_response_fd` (ham_bml, prt_bml, rsp_bml, prg_delta, bndfil, threshold, verbose)

Computes the first order response density matrix using finite differences. The transformation hereby performed are:

- subroutine, public `prg_response_mod::prg_pert_constant_field` (field, intensity, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)

Apply a constant field perturbation through the dipole moment operator ($\hat{\mu} = e\hat{r}$). In the matrix representation, this is: $H^{(1)} = \lambda \frac{1}{2} (S e\mathbf{r} \cdot \mathbf{E} + e\mathbf{r} \cdot \mathbf{E} S)$. The symmetrization is done in order to preserve the Hermiticity of H . In this case the whole system will be affected by the field. In a latter version we will add the possibility of applying this field to a region of the system. In this implementation $e = 1$ and units can be transformed by using the parameter λ .

- subroutine, public `prg_response_mod::prg_pert_sin_pot` (direction, lx, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)

Apply a sinusoidal length dependent potential ($\sin(\tilde{r}_x)$) where \mathbf{r}_x is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2}\lambda(S \sin(\tilde{r}_x) + \sin(\tilde{r}_x)S)$. $\tilde{r}_x = 2\pi(\mathbf{r}/l_x) - \pi$. The symmetrization is done in order to preserve the Hermiticity of H . Units can be transformed by using the parameter λ .

- subroutine, public `prg_response_mod::prg_pert_cos_pot` (direction, lx, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)

Apply a cosine length dependent potential ($\cos(\tilde{r}_x)$) where \mathbf{r}_x is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2}\lambda(S \cos(\tilde{r}_x) + \cos(\tilde{r}_x)S)$. $\tilde{r}_x = 2\pi(\mathbf{r}/l_x) - \pi$. The symmetrization is done in order to preserve the Hermiticity of H . Units can be transformed by using the parameter λ .

- subroutine, public `prg_response_mod::prg_compute_response_sp2` (ham_bml, prt_bml, rsp_bml, rho_bml, lambda, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtoll, threshold, verbose)

Finds the first order response matrix from a Hamiltonian matrix.

- subroutine, public `prg_response_mod::prg_project_response` (rsp_bml, over_bml, spindex, norbi, coordinates, rspfunc, verbose)

Project the response onto atomic positions. First order response to the perturbation ($\rho^{(1)}$) projected onto the atomic position. Basically: $rsp(i) = \sum_{\alpha \in i} \rho_{\alpha\alpha}^{(1)}$, where orbital α belong to atom i .

Variables

- integer, parameter `prg_response_mod::dp` = kind(1.0d0)
- real(dp), parameter `prg_response_mod::pi` = 3.14159265358979323846264338327950_dp

11.27 /home/christian/qmd-progress/src/prg_sp2_fermi_mod.F90 File Reference

Modules

- module `prg_sp2_fermi_mod`

The SP2 Fermi module.

Functions/Subroutines

- subroutine, public `prg_sp2_fermi_mod::prg_sp2_fermi_init` (h_bml, nsteps, nocc, tscale, threshold, occErrLimit, traceLimit, x_bml, mu, beta, h1, hN, sgnlist)

Truncated SP2 prg_initialization.

- subroutine, public `prg_sp2_fermi_mod::prg_sp2_fermi_init_norecs` (h_bml, nsteps, nocc, tscale, threshold, occErrLimit, traceLimit, x_bml, mu, beta, h1, hN, sgnlist, verbose)

Truncated SP2 prg_initialization. This routine also gives back the Number of SP2 recursive steps that gets a Pseudo-Fermi distribution with a temperature close to the target temperature which is entered using parameter $\beta = (1/KbT)$.

- subroutine, public [prg_sp2_fermi_mod::prg_sp2_fermi](#) (h_bml, osteps, nsteps, nocc, mu, beta, h1, hN, sgnlist, threshold, eps, traceLimit, x_bml)
Calculate Truncated SP2.
- subroutine, public [prg_sp2_fermi_mod::prg_sp2_entropy_function](#) (mu, h1, hN, nsteps, sgnlist, GG, ee)
Calculate SP2 entropy function using gaussian quadrature. Note that GG and ee are allocated and returned from this routine.
- real(dp) function, public [prg_sp2_fermi_mod::sp2_entropy_ts](#) (D0_bml, GG, ee)
Test SP2 entropy. Get the entropy contribution TS to the total free energy.
- real(dp) function, public [prg_sp2_fermi_mod::sp2_inverse](#) (f, mu, h1, hN, nsteps, sgnlist)
Calculate the SP2 inverse.
- real(dp) function [prg_sp2_fermi_mod::absmaxderivative](#) (func, de)
Gets the absolute maximum of the derivative of a function.

Variables

- integer, parameter [prg_sp2_fermi_mod::dp](#) = kind(1.0d0)

11.28 /home/christian/qmd-progress/src/prg_sp2_mod.F90 File Reference

Modules

- module [prg_sp2_mod](#)
The SP2 module.

Functions/Subroutines

- subroutine, public [prg_sp2_mod::prg_sp2_basic](#) (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemt看, verbose)
Calculates the density matrix from a Hamiltonian matrix by purification. The method implemented here is the very first verion of the SP2 method.
- subroutine, public [prg_sp2_mod::prg_sp2_alg2](#) (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemt看, verbose)
- subroutine, public [prg_sp2_mod::prg_sp2_alg2_genseq](#) (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemt看, pp, icount, vv, verbose)
- subroutine, public [prg_sp2_mod::prg_sp2_alg2_seq](#) (h_bml, rho_bml, threshold, pp, icount, vv, verbose)
- subroutine, public [prg_sp2_mod::prg_prg_sp2_alg2_seq_inplace](#) (rho_bml, threshold, pp, icount, vv, mineval, maxeval, verbose)
- subroutine, public [prg_sp2_mod::prg_sp2_alg1](#) (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemt看, verbose)
- subroutine, public [prg_sp2_mod::prg_sp2_alg1_genseq](#) (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemt看, pp, icount, vv)
- subroutine, public [prg_sp2_mod::prg_sp2_alg1_seq](#) (h_bml, rho_bml, threshold, pp, icount, vv)
- subroutine, public [prg_sp2_mod::prg_prg_sp2_alg1_seq_inplace](#) (rho_bml, threshold, pp, icount, vv, mineval, maxeval)
- subroutine, public [prg_sp2_mod::prg_sp2_submatrix](#) (ham_bml, rho_bml, threshold, pp, icount, vv, mineval, maxeval, core_size)
Perform SP2 algorithm using sequence and calculate norm for a submatrix.
- subroutine, public [prg_sp2_mod::prg_sp2_submatrix_inplace](#) (rho_bml, threshold, pp, icount, vv, mineval, maxeval, core_size)

Variables

- integer, parameter `prg_sp2_mod::dp` = kind(1.0d0)

11.29 /home/christian/qmd-progress/src/prg_sp2parser_mod.F90 File Reference

Data Types

- type `prg_sp2parser_mod::sp2data_type`
General SP2 solver type.

Modules

- module `prg_sp2parser_mod`
SP2 parser.

Functions/Subroutines

- subroutine, public `prg_sp2parser_mod::prg_parse_sp2` (sp2data, filename)
The parser for SP2 solver.

Variables

- integer, parameter `prg_sp2parser_mod::dp` = kind(1.0d0)

11.30 /home/christian/qmd-progress/src/prg_subgraphloop_mod.F90 File Reference

Modules

- module `prg_subgraphloop_mod`
The subgraphloop module.

Functions/Subroutines

- subroutine, public `prg_subgraphloop_mod::prg_subgraphsp2loop` (h_bml, g_bml, rho_bml, gp, threshold)
- subroutine, public `prg_subgraphloop_mod::prg_collectmatrixfromparts` (gp, rho_bml)
Collect distributed parts into same matrix.
- subroutine, public `prg_subgraphloop_mod::prg_balanceparts` (gp)
- subroutine, public `prg_subgraphloop_mod::prg_partordering` (gp)
Set row ordering bases on parts.
- subroutine, public `prg_subgraphloop_mod::prg_getgrouppartitionhalosfromgraph` (gp, g_bml, hnode, djflag)
Get core+halo indeces for all partitions only using the graph.
- subroutine, public `prg_subgraphloop_mod::prg_getpartitionhalosfromgraph` (gp, g_bml, djflag)
Get core+halo indeces for all partitions only using the graph.

Variables

- integer, parameter `prg_subgraphloop_mod::dp` = kind(1.0d0)

11.31 /home/christian/qmd-progress/src/prg_syrotation_mod.F90 File Reference

Data Types

- type `prg_syrotation_mod::rotation_type`
Rotation type.

Modules

- module `prg_syrotation_mod`
A module to rotate the coordinates of a sybsystem in chemical systems.

Functions/Subroutines

- subroutine, public `prg_syrotation_mod::prg_parse_rotation` (rot, filename)
The parser for rotation.
- subroutine, public `prg_syrotation_mod::prg_rotate` (rot, r, verbose)
Rotation routine.

Variables

- integer, parameter `prg_syrotation_mod::dp` = kind(1.0d0)

11.32 /home/christian/qmd-progress/src/prg_system_mod.F90 File Reference

Data Types

- type `prg_system_mod::estruct_type`
Electronic structure type.
- type `prg_system_mod::system_type`
System type.

Modules

- module `prg_system_mod`
A module to read and handle chemical systems.

Functions/Subroutines

- subroutine, public [prg_system_mod::prg_get_nameandext](#) (fullfilename, filename, ext)
Get the name and extension of a file.
- subroutine, public [prg_system_mod::prg_parse_system](#) (system, filename, extin)
The parser for the chemical system.
- subroutine, public [prg_system_mod::prg_write_system](#) (system, filename, extension)
Write system in .xyz, .dat or pdb file.
- subroutine, public [prg_system_mod::prg_write_trajectory](#) (system, iter, each, prg_deltat, filename, extension)
Write trajectory in .xyz, .dat or pdb file.
- subroutine, public [prg_system_mod::prg_write_trajectoryandproperty](#) (system, iter, each, prg_deltat, scalarprop, filename, extension)
Write trajectory and atomic properties. Only pdb file.
- subroutine, public [prg_system_mod::prg_make_random_system](#) (system, nats, seed, lx, ly, lz)
Make random Xx system.
- subroutine [prg_system_mod::prg_parameters_to_vectors](#) (abc_angles, lattice_vector)
Transforms the lattice parameters into lattice vectors.
- subroutine [prg_system_mod::prg_vectors_to_parameters](#) (lattice_vector, abc_angles)
Transforms the lattice vectors into lattice parameters.
- subroutine, public [prg_system_mod::prg_get_origin](#) (coords, origin)
Get the origin of the coordinates.
- subroutine, public [prg_system_mod::prg_get_distancematrix](#) (coords, dmat)
Get the distance matrix.
- subroutine, public [prg_system_mod::prg_translateandfoldtobox](#) (coords, lattice_vectors, origin, verbose)
Translate and fold to box.
- subroutine, public [prg_system_mod::prg_centeratbox](#) (coords, lattice_vectors, verbose)
Translate geometric center to the center of the box.
- subroutine, public [prg_system_mod::prg_wraparound](#) (coords, lattice_vectors, index, verbose)
Wrap around atom i using pbc.
- subroutine, public [prg_system_mod::prg_translatetogeomcandfoldtobox](#) (coords, lattice_vectors, origin)
Translate to geometric center.
- subroutine, public [prg_system_mod::prg_replicate](#) (coords, symbols, lattice_vectors, nx, ny, nz)
Extend/replicate system along lattice vectors.
- subroutine, public [prg_system_mod::prg_get_recip_vects](#) (lattice_vectors, recip_vectors, volr, volk)
Get the volume of the cell and the reciprocal vectors: This subroutine computes:
- subroutine, public [prg_system_mod::prg_get_dihedral](#) (coords, id1, id2, id3, id4, dihedral)
Get the dihedral angle given four atomic positions.
- subroutine, public [prg_system_mod::prg_get_covgraph](#) (sy, nnStructMindist, nnStruct, nrrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)
Get the covalency graph in bml format.
- subroutine [prg_system_mod::prg_get_covgraph_int](#) (sy, nnStructMindist, nnStruct, nrrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)
- subroutine, public [prg_system_mod::prg_get_covgraph_h](#) (sy, nnStructMindist, nnStruct, nrrnnstruct, rcut, graph_h, mdimin, verbose)
Get the covanlency graph.
- subroutine, public [prg_system_mod::prg_get_subsystem](#) (sy, lsize, indices, sbsy, verbose)
Get a subsystem out of the total system.
- subroutine, public [prg_system_mod::prg_destroy_subsystems](#) (sbsy, verbose)
Destroy allocated subsystem.
- subroutine, public [prg_system_mod::prg_molpartition](#) (sy, npart, nnStructMindist, nnStruct, nrrnnstruct, het-atm, gp, verbose)
Partition by molecule.

- subroutine, public `prg_system_mod::prg_get_partial_atomgraph` (rho_bml, hindex, gch_bml, threshold, verbose)
Get partial subgraph based on the Density matrix.
- subroutine, public `prg_system_mod::prg_collect_graph_p` (rho_bml, nc, nats, hindex, chindex, graph_p, threshold, mdimin, verbose)
Collect the small graph to build the full graph.
- subroutine, public `prg_system_mod::prg_merge_graph` (graph_p, graph_h)
Get partial subgraph based on the Density matrix.
- subroutine, public `prg_system_mod::prg_merge_graph_adj` (graph_p, graph_h, xadj, adjncy)
Get partial subgraph based on the Density matrix.
- subroutine, public `prg_system_mod::prg_adj2bml` (xadj, adjncy, bml_type, g_bml)
prg_adj2bml
- subroutine, public `prg_system_mod::prg_graph2bml` (graph, bml_type, g_bml)
Graph2bml.
- subroutine, public `prg_system_mod::prg_graph2vector` (graph, vector, maxnz)
Vectorize graph.
- subroutine, public `prg_system_mod::prg_vector2graph` (vector, graph, maxnz)
Back to graph.
- subroutine, public `prg_system_mod::prg_sortadj` (xadj, adjncy)
Sort adj NOTE: this might not be needed anymore since the bml_get_adj routine is sorting the values.

Variables

- integer, parameter `prg_system_mod::dp` = kind(1.0d0)

11.33 /home/christian/qmd-progress/src/prg_timer_mod.F90 File Reference

Data Types

- type `prg_timer_mod::timer_status_t`
Timer status type.

Modules

- module `prg_timer_mod`
The timer module.

Functions/Subroutines

- subroutine, public `prg_timer_mod::timer_prg_init` ()
Initialize timers.
- subroutine `prg_timer_mod::prg_timer_getid` ()
Get timer id.
- subroutine, public `prg_timer_mod::prg_timer_shutdown` ()
Done with timers.
- subroutine, public `prg_timer_mod::prg_timer_start` (itimer, tag)
Start Timing.
- subroutine, public `prg_timer_mod::prg_timer_stop` (itimer, verbose)
Stop timing.
- subroutine, public `prg_timer_mod::prg_timer_collect` ()
- subroutine, public `prg_timer_mod::prg_timer_results` ()
- real(8) function, public `prg_timer_mod::time2milliseconds` ()
- subroutine, public `prg_timer_mod::prg_print_date_and_time` (tag)
- character(2) function, private `prg_timer_mod::int2char` (ival)

Variables

- integer, parameter `prg_timer_mod::dp` = kind(1.0d0)
- integer, public `prg_timer_mod::loop_timer`
- integer, public `prg_timer_mod::sp2_timer`
- integer, public `prg_timer_mod::genx_timer`
- integer, public `prg_timer_mod::part_timer`
- integer, public `prg_timer_mod::subgraph_timer`
- integer, public `prg_timer_mod::deortho_timer`
- integer, public `prg_timer_mod::ortho_timer`
- integer, public `prg_timer_mod::zdiag_timer`
- integer, public `prg_timer_mod::graphsp2_timer`
- integer, public `prg_timer_mod::subind_timer`
- integer, public `prg_timer_mod::subext_timer`
- integer, public `prg_timer_mod::subsp2_timer`
- integer, public `prg_timer_mod::suball_timer`
- integer, public `prg_timer_mod::bmult_timer`
- integer, public `prg_timer_mod::badd_timer`
- integer, public `prg_timer_mod::dyn_timer`
- integer, public `prg_timer_mod::mdloop_timer`
- integer, public `prg_timer_mod::buildz_timer`
- integer, public `prg_timer_mod::realcoul_timer`
- integer, public `prg_timer_mod::recipcoul_timer`
- integer, public `prg_timer_mod::pairpot_timer`
- integer, public `prg_timer_mod::halfverlet_timer`
- integer, public `prg_timer_mod::pos_timer`
- integer, public `prg_timer_mod::nlist_timer`
- integer `prg_timer_mod::tstart_clock`
- integer `prg_timer_mod::tstop_clock`
- integer `prg_timer_mod::tclock_rate`
- integer `prg_timer_mod::tclock_max`
- integer `prg_timer_mod::num_timers`
- type(timer_status_t), dimension(:), allocatable `prg_timer_mod::ptimer`

11.34 /home/christian/qmd-progress/src/prg_xlbo_mod.F90 File Reference

Data Types

- type `prg_xlbo_mod::xlbo_type`
General xlbo solver type.

Modules

- module `prg_xlbo_mod`
A module to perform XLBO integration.

Functions/Subroutines

- subroutine, public [prg_xlbo_mod::prg_parse_xlbo](#) (xlbo, filename)
The parser for XLBO parser.
- subroutine, public [prg_xlbo_mod::prg_xlbo_nint](#) (charges, n, n_0, n_1, n_2, n_3, n_4, n_5, mdstep, xl)
This routine integrates the dynamical variable "n".
- subroutine, public [prg_xlbo_mod::prg_xlbo_fcoulupdate](#) (fcoul, charges, n)
Adjust forces for the linearized XLBOMD functional.

Variables

- integer, parameter [prg_xlbo_mod::dp](#) = kind(1.0d0)
- real(dp), parameter [prg_xlbo_mod::c0](#) = -6.0_dp
Coefficients for modified Verlet integration.
- real(dp), parameter [prg_xlbo_mod::c1](#) = 14.0_dp
- real(dp), parameter [prg_xlbo_mod::c2](#) = -8.0_dp
- real(dp), parameter [prg_xlbo_mod::c3](#) = -3.0_dp
- real(dp), parameter [prg_xlbo_mod::c4](#) = 4.0_dp
- real(dp), parameter [prg_xlbo_mod::c5](#) = -1.0_dp
- real(dp), parameter [prg_xlbo_mod::kappa](#) = 1.82_dp
Coefficients for modified Verlet integration.
- real(dp), parameter [prg_xlbo_mod::alpha](#) = 0.018_dp
- real(dp), parameter [prg_xlbo_mod::cc](#) = 0.9_dp

11.35 /home/christian/qmd-progress/src/prg_xlkernel_mod.F90 File Reference

Data Types

- type [prg_xlkernel_mod::xlk_type](#)

Modules

- module [prg_xlkernel_mod](#)
Add name.

Functions/Subroutines

- subroutine, public [prg_xlkernel_mod::prg_parse_xlkernel](#) (input, filename)
The parser for the mixer routines.
- subroutine, public [prg_xlkernel_mod::prg_fermi](#) (D0, QQ, ee, gap, Fe_vec, mu0, H, Z, Nocc, T, OccErrLim, MaxIt, HDIM)
- subroutine, public [prg_xlkernel_mod::prg_kernel_fermi_full](#) (KK, JJ, D0, mu0, mu1, T, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_atoms, MaxIt, eps, m, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERAT←IO, nnRx, nnRy, nnRz, nrrnlist, nnType, H_INDEX_START, H_INDEX_END, H, S, Z, Nocc, Znuc, QQ, ee, Fe_vec)
- subroutine, public [prg_xlkernel_mod::prg_v_kernel_fermi](#) (D0, dq_dv, v, mu0, mu1, T, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_atoms, MaxIt, eps, m, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERAT←IO, nnRx, nnRy, nnRz, nrrnlist, nnType, H_INDEX_START, H_INDEX_END, H, S, Z, Nocc, Znuc, QQ, ee, Fe_vec)
- subroutine, private [prg_xlkernel_mod::prg_get_deriv_finite_temp](#) (P1, H0, H1, Nocc, T, Q, ev, fe, mu0, eps, HDIM)
- subroutine, private [prg_xlkernel_mod::prg_mmult](#) (alpha, A, B, beta, C, TA, TB, HDIM)
- subroutine, private [prg_xlkernel_mod::prg_eig](#) (A, Q, ee, type, HDIM)
- subroutine, private [prg_xlkernel_mod::prg_inv](#) (X, XI, HDIM)
- subroutine, public [prg_xlkernel_mod::prg_rank1](#) (verbose)
Rank1 kernel

Variables

- integer, parameter `prg_xlkernel_mod::dp = kind(1.0d0)`

Bibliography

- [1] S. M. Mniszewski, M. J. Cawkwell, M. E. Wall, J. Mohd-Yusof, N. Bock, T. C. Germann, and A. M. N. Niklasson. Efficient parallel linear scaling construction of the density matrix for born–oppenheimer molecular dynamics. *Journal of Chemical Theory and Computation*, 11(10):4644–4654, 2015. PMID: 26574255. [63](#), [64](#), [189](#), [190](#)
- [2] Christian F. A. Negre, Susan M. Mniszewski, Marc J. Cawkwell, Nicolas Bock, Michael E. Wall, and Anders M. N. Niklasson. Recursive factorization of the inverse overlap matrix in linear-scaling quantum molecular dynamics simulations. *Journal of Chemical Theory and Computation*, 12(7):3063–3073, 2016. PMID: 27267207. [45](#)
- [3] Anders M. N. Niklasson. A note on the pulay force at finite electronic temperatures. *The Journal of Chemical Physics*, 129(24), 2008. [117](#)
- [4] Anders M. N. Niklasson, Valéry Weber, and Matt Challacombe. Nonorthogonal density-matrix perturbation theory. *The Journal of Chemical Physics*, 123(4), 2005. [126](#)
- [5] Valéry Weber, Anders M. N. Niklasson, and Matt Challacombe. Higher-order response in $\mathcal{O}(n)$ by perturbed projection. *The Journal of Chemical Physics*, 123(4), 2005. [125](#), [127](#), [239](#)

Index

- (EXTERNAL related routines), [20](#)
- (High-level codes using PROGRESS/LATTE modules), [21](#)
- (LATTE related routines), [17](#)
- (PROGRESS related routines), [18](#)
- /home/christian/qmd-progress/README.md, [223](#)
- /home/christian/qmd-progress/src/prg_charges_mod.↔
F90, [223](#)
- /home/christian/qmd-progress/src/prg_chebyshev_↔
mod.F90, [223](#)
- /home/christian/qmd-progress/src/prg_densitymatrix_↔
mod.F90, [224](#)
- /home/christian/qmd-progress/src/prg_dos_mod.F90,
[225](#)
- /home/christian/qmd-progress/src/prg_doxy_mod.F90,
[226](#)
- /home/christian/qmd-progress/src/prg_extras_mod.F90,
[226](#)
- /home/christian/qmd-progress/src/prg_genz_mod.F90,
[226](#)
- /home/christian/qmd-progress/src/prg_graph_mod.F90,
[227](#)
- /home/christian/qmd-progress/src/prg_graphsp2parser_↔
_mod.F90, [228](#)
- /home/christian/qmd-progress/src/prg_homolumo_↔
mod.F90, [229](#)
- /home/christian/qmd-progress/src/prg_implicit_fermi_↔
mod.F90, [229](#)
- /home/christian/qmd-progress/src/prg_initmatrices_↔
mod.F90, [229](#)
- /home/christian/qmd-progress/src/prg_kernelparser_↔
mod.F90, [230](#)
- /home/christian/qmd-progress/src/prg_nonortho_↔
mod.F90, [230](#)
- /home/christian/qmd-progress/src/prg_normalize_↔
mod.F90, [231](#)
- /home/christian/qmd-progress/src/prg_openfiles_↔
mod.F90, [231](#)
- /home/christian/qmd-progress/src/prg_parallel_mod.↔
F90, [232](#)
- /home/christian/qmd-progress/src/prg_partition_mod.↔
F90, [233](#)
- /home/christian/qmd-progress/src/prg_progress_↔
mod.F90, [234](#)
- /home/christian/qmd-progress/src/prg_ptable_mod.F90,
[234](#)
- /home/christian/qmd-progress/src/prg_pulaycomponent_↔
_mod.F90, [237](#)
- /home/christian/qmd-progress/src/prg_pulaymixer_↔
mod.F90, [237](#)
- /home/christian/qmd-progress/src/prg_quantumdynamics_↔
_mod.F90, [238](#)
- /home/christian/qmd-progress/src/prg_response_↔
mod.F90, [239](#)
- /home/christian/qmd-progress/src/prg_sp2_fermi_↔
mod.F90, [240](#)
- /home/christian/qmd-progress/src/prg_sp2_mod.F90,
[241](#)
- /home/christian/qmd-progress/src/prg_sp2parser_↔
mod.F90, [242](#)
- /home/christian/qmd-progress/src/prg_subgraphloop_↔
mod.F90, [242](#)
- /home/christian/qmd-progress/src/prg_syrotation_↔
mod.F90, [243](#)
- /home/christian/qmd-progress/src/prg_system_mod.↔
F90, [243](#)
- /home/christian/qmd-progress/src/prg_timer_mod.F90,
[245](#)
- /home/christian/qmd-progress/src/prg_xlbo_mod.F90,
[246](#)
- /home/christian/qmd-progress/src/prg_xlkernel_mod.↔
F90, [247](#)
- /home/christian/qmd-progress/tests/README.md, [223](#)
- absmaxderivative
 - prg_chebyshev_mod, [26](#)
 - prg_sp2_fermi_mod, [132](#)
- allgatherintparallel
 - prg_parallel_mod, [78](#)
- allgatherrealparallel
 - prg_parallel_mod, [78](#)
- allgathervintparallel
 - prg_parallel_mod, [79](#)
- allgathervrealparallel
 - prg_parallel_mod, [79](#)
- alpha
 - prg_xlbo_mod, [175](#)
- atom_en
 - prg_ptable_mod, [113](#)
- atomic_number
 - prg_system_mod::system_type, [211](#)
- atr
 - prg_chebyshev_mod::chebdata_type, [182](#)
- badd_timer
 - prg_timer_mod, [170](#)
- bml_type
 - prg_chebyshev_mod::chebdata_type, [182](#)
 - prg_genz_mod::genzspinp, [189](#)

- prg_graphsp2parser_mod::gsp2data_type, 197
- prg_sp2parser_mod::sp2data_type, 206
- bmltype
 - prg_response_mod::respdata_type, 202
- bmult_timer
 - prg_timer_mod, 170
- bndfil
 - prg_chebyshev_mod::chebdata_type, 182
 - prg_graphsp2parser_mod::gsp2data_type, 197
 - prg_sp2parser_mod::sp2data_type, 206
- buildz_timer
 - prg_timer_mod, 171
- c0
 - prg_xlbo_mod, 176
- c1
 - prg_xlbo_mod, 176
- c2
 - prg_xlbo_mod, 176
- c3
 - prg_xlbo_mod, 176
- c4
 - prg_xlbo_mod, 176
- c5
 - prg_xlbo_mod, 176
- catom
 - prg_syrotation_mod::rotation_type, 204
- catom2
 - prg_syrotation_mod::rotation_type, 204
- cc
 - prg_xlbo_mod, 176
 - prg_xlbo_mod::xlbo_type, 219
- computedipole
 - prg_response_mod::respdata_type, 202
- coordinate
 - prg_system_mod::system_type, 211
- core_halo_index
 - prg_graph_mod::subgraph_t, 208
- coul_pot_k
 - prg_system_mod::estruct_type, 184
- coul_pot_r
 - prg_system_mod::estruct_type, 184
- covgfact
 - prg_graphsp2parser_mod::gsp2data_type, 197
- deortho_timer
 - prg_timer_mod, 171
- double_jump
 - prg_graphsp2parser_mod::gsp2data_type, 197
- dp
 - prg_charges_mod, 25
 - prg_chebyshev_mod, 33
 - prg_densitymatrix_mod, 38
 - prg_dos_mod, 40
 - prg_extras_mod, 44
 - prg_genz_mod, 50
 - prg_graph_mod, 59
 - prg_graphsp2parser_mod, 60
 - prg_homolumo_mod, 61
 - prg_implicit_fermi_mod, 62
 - prg_initmatrices_mod, 64
 - prg_kernelparser_mod, 67
 - prg_nonortho_mod, 69
 - prg_normalize_mod, 73
 - prg_parallel_mod, 92
 - prg_partition_mod, 108
 - prg_progress_mod, 110
 - prg_ptable_mod, 113
 - prg_pulaycomponent_mod, 118
 - prg_pulaymixer_mod, 120
 - prg_quantumdynamics_mod, 125
 - prg_response_mod, 131
 - prg_sp2_fermi_mod, 137
 - prg_sp2_mod, 143
 - prg_sp2parser_mod, 144
 - prg_subgraphloop_mod, 148
 - prg_syrotation_mod, 150
 - prg_system_mod, 165
 - prg_timer_mod, 171
 - prg_xlbo_mod, 176
 - prg_xlkernel_mod, 180
- dyn_timer
 - prg_timer_mod, 171
- eband
 - prg_system_mod::estruct_type, 185
- ef
 - prg_chebyshev_mod::chebdata_type, 182
- ehomo
 - prg_graph_mod::graph_partitioning_t, 192
- element_atomic_number
 - prg_ptable_mod, 113
- element_atomic_number_upper
 - prg_ptable_mod, 113
- element_covr
 - prg_ptable_mod, 114
- element_ea
 - prg_ptable_mod, 114
- element_econf
 - prg_ptable_mod, 114
- element_ip
 - prg_ptable_mod, 114
- element_mass
 - prg_ptable_mod, 115
- element_maxbonds
 - prg_ptable_mod, 115
- element_name
 - prg_ptable_mod, 115
- element_numel
 - prg_ptable_mod, 115
- element_symbol
 - prg_ptable_mod, 116
- element_symbol_upper
 - prg_ptable_mod, 116
- element_vdwr
 - prg_ptable_mod, 116
- elumo
 - prg_graph_mod::graph_partitioning_t, 192

- errlimit
 - prg_graphsp2parser_mod::gsp2data_type, 197
- estep
 - prg_chebyshev_mod::chebdata_type, 182
- estr
 - prg_system_mod::system_type, 211
- fermi
 - prg_chebyshev_mod, 26
 - prg_densitymatrix_mod, 34
- fermitol
 - prg_chebyshev_mod::chebdata_type, 182
- field
 - prg_response_mod::respdata_type, 202
- fieldintensity
 - prg_response_mod::respdata_type, 202
- flavor
 - prg_chebyshev_mod::chebdata_type, 182
 - prg_sp2parser_mod::sp2data_type, 206
- force
 - prg_system_mod::system_type, 211
- fpul
 - prg_system_mod::estruct_type, 185
- fscoul
 - prg_system_mod::estruct_type, 185
- genx_timer
 - prg_timer_mod, 171
- get_file_unit
 - prg_openfiles_mod, 74
- getef
 - prg_chebyshev_mod::chebdata_type, 182
- getmyrank
 - prg_parallel_mod, 79
- getnranks
 - prg_parallel_mod, 79
- getresponse
 - prg_response_mod::respdata_type, 202
- globalpartextent
 - prg_graph_mod::graph_partitioning_t, 193
- globalpartmax
 - prg_graph_mod::graph_partitioning_t, 193
- globalpartmin
 - prg_graph_mod::graph_partitioning_t, 193
- graph_element
 - prg_graphsp2parser_mod::gsp2data_type, 197
- graphsp2_timer
 - prg_timer_mod, 171
- gthreshold
 - prg_graphsp2parser_mod::gsp2data_type, 197
- halfverlet_timer
 - prg_timer_mod, 171
- ham
 - prg_system_mod::estruct_type, 185
- ham0
 - prg_system_mod::estruct_type, 185
- hamfile
 - prg_graphsp2parser_mod::gsp2data_type, 197
- hindex
 - prg_system_mod::estruct_type, 185
- hsize
 - prg_graph_mod::subgraph_t, 208
- ierr
 - prg_parallel_mod, 92
- igenz
 - prg_genz_mod::genzspinp, 189
- int2char
 - prg_timer_mod, 166
- integration
 - prg_genz_mod::genzspdata, 187
 - prg_genz_mod::genzspinp, 189
- isendparallel
 - prg_parallel_mod, 80
- jackson
 - prg_chebyshev_mod, 27
- jobname
 - prg_chebyshev_mod::chebdata_type, 182
 - prg_graphsp2parser_mod::gsp2data_type, 197
 - prg_sp2parser_mod::sp2data_type, 206
 - prg_syrotation_mod::rotation_type, 204
 - prg_xlbo_mod::xlbo_type, 219
- jon
 - prg_chebyshev_mod::chebdata_type, 182
- kappa
 - prg_xlbo_mod, 176
- kbt
 - prg_chebyshev_mod::chebdata_type, 182
- kerneltype
 - prg_xlkernel_mod::xlk_type, 220
- lattice_vector
 - prg_system_mod::system_type, 211
- llsize
 - prg_graph_mod::subgraph_t, 208
- localpartextent
 - prg_graph_mod::graph_partitioning_t, 193
- localpartmax
 - prg_graph_mod::graph_partitioning_t, 193
- localpartmin
 - prg_graph_mod::graph_partitioning_t, 193
- loop_timer
 - prg_timer_mod, 171
- lorentz
 - prg_dos_mod, 39
- lsize
 - prg_graph_mod::subgraph_t, 208
- mass
 - prg_system_mod::system_type, 211
- maxeval
 - prg_graph_mod::graph_partitioning_t, 193
- maxintparallel
 - prg_parallel_mod, 80
- maxiter

- prg_graph_mod::graph_partitioning_t, 193
- maxrank
 - prg_timer_mod::timer_status_t, 215
- maxrankrealparallel
 - prg_parallel_mod, 81
- maxrealparallel
 - prg_parallel_mod, 81
- maxscfinititer
 - prg_xlbo_mod::xlbo_type, 219
- maxscfiter
 - prg_xlbo_mod::xlbo_type, 219
- maxsp2iter
 - prg_graphsp2parser_mod::gsp2data_type, 197
 - prg_sp2parser_mod::sp2data_type, 206
- maxvalue
 - prg_timer_mod::timer_status_t, 215
- mdim
 - prg_chebyshev_mod::chebdata_type, 183
 - prg_genz_mod::genzspinp, 190
 - prg_graphsp2parser_mod::gsp2data_type, 197
 - prg_response_mod::respdata_type, 203
 - prg_sp2parser_mod::sp2data_type, 206
- mdloop_timer
 - prg_timer_mod, 171
- metis_index_kind
 - prg_partition_mod, 108
- metis_real_kind
 - prg_partition_mod, 108
- mineval
 - prg_graph_mod::graph_partitioning_t, 194
- minintparallel
 - prg_parallel_mod, 81
- minit
 - prg_xlbo_mod::xlbo_type, 219
- minrank
 - prg_timer_mod::timer_status_t, 216
- minrankrealparallel
 - prg_parallel_mod, 81
- minrealparallel
 - prg_parallel_mod, 82
- minsp2iter
 - prg_graphsp2parser_mod::gsp2data_type, 198
 - prg_sp2parser_mod::sp2data_type, 207
- minvalue
 - prg_timer_mod::timer_status_t, 216
- mixcoeff
 - prg_pulaymixer_mod::mx_type, 200
- mixeron
 - prg_pulaymixer_mod::mx_type, 200
- mixertype
 - prg_pulaymixer_mod::mx_type, 200
- mls
 - prg_extras_mod, 41
- mpulay
 - prg_pulaymixer_mod::mx_type, 200
- myrank
 - prg_graph_mod::graph_partitioning_t, 194
 - prg_parallel_mod, 92
- natoms
 - prg_graphsp2parser_mod::gsp2data_type, 198
- nats
 - prg_system_mod::system_type, 212
- ncoeffs
 - prg_chebyshev_mod::chebdata_type, 183
- ndim
 - prg_chebyshev_mod::chebdata_type, 183
 - prg_graphsp2parser_mod::gsp2data_type, 198
 - prg_sp2parser_mod::sp2data_type, 207
- nel
 - prg_system_mod::estruct_type, 185
- net_charge
 - prg_system_mod::system_type, 212
- nfirsr
 - prg_genz_mod::genzspdata, 187
 - prg_genz_mod::genzspinp, 190
- nlgcut
 - prg_graphsp2parser_mod::gsp2data_type, 198
- nlist_timer
 - prg_timer_mod, 171
- nnodesinpart
 - prg_graph_mod::graph_partitioning_t, 194
- nnodesinpartall
 - prg_graph_mod::graph_partitioning_t, 194
- nodeinpart
 - prg_graph_mod::subgraph_t, 209
- nodesperpart
 - prg_graphsp2parser_mod::gsp2data_type, 198
- norbs
 - prg_system_mod::estruct_type, 185
- nparts
 - prg_graph_mod::graph_partitioning_t, 194
- npts
 - prg_chebyshev_mod::chebdata_type, 183
- nrank
 - prg_xlkernel_mod::xlk_type, 220
- nrank
 - prg_parallel_mod, 92
- nref
 - prg_genz_mod::genzspdata, 188
 - prg_genz_mod::genzspinp, 190
- nrefi
 - prg_genz_mod::genzspdata, 188
 - prg_genz_mod::genzspinp, 190
- nsp
 - prg_system_mod::system_type, 212
- num_timers
 - prg_timer_mod, 172
- numthresf
 - prg_genz_mod::genzspdata, 188
 - prg_genz_mod::genzspinp, 190
- numthresh
 - prg_response_mod::respdata_type, 203
- numthresi
 - prg_genz_mod::genzspdata, 188
 - prg_genz_mod::genzspinp, 190
- nz

- prg_ptable_mod, 116
- oham
 - prg_system_mod::estruct_type, 186
- order
 - prg_graph_mod::graph_partitioning_t, 194
- orho
 - prg_system_mod::estruct_type, 186
- ortho_timer
 - prg_timer_mod, 172
- over
 - prg_system_mod::estruct_type, 186
- pairpot_timer
 - prg_timer_mod, 172
- part
 - prg_graph_mod::subgraph_t, 209
- part_timer
 - prg_timer_mod, 172
- parteach
 - prg_graphsp2parser_mod::gsp2data_type, 198
- partition_count
 - prg_graphsp2parser_mod::gsp2data_type, 198
- partition_refinement
 - prg_graphsp2parser_mod::gsp2data_type, 198
- partition_type
 - prg_graphsp2parser_mod::gsp2data_type, 198
- patom1
 - prg_syrotation_mod::rotation_type, 204
- patom2
 - prg_syrotation_mod::rotation_type, 204
- pdim
 - prg_graphsp2parser_mod::gsp2data_type, 198
 - prg_sp2parser_mod::sp2data_type, 207
- pi
 - prg_chebyshev_mod, 33
 - prg_response_mod, 131
- pname
 - prg_graph_mod::graph_partitioning_t, 194
- pos_timer
 - prg_timer_mod, 172
- pp
 - prg_graph_mod::graph_partitioning_t, 194
- pq1
 - prg_syrotation_mod::rotation_type, 204
- pq2
 - prg_syrotation_mod::rotation_type, 204
- prg_accept_prob
 - prg_partition_mod, 94
- prg_adj2bml
 - prg_system_mod, 152
- prg_allgatherparallel
 - prg_parallel_mod, 82
- prg_allocatezspmat
 - prg_genz_mod, 45
 - prg_genz_mod::genzspdata, 187
- prg_allsumintreduceparallel
 - prg_parallel_mod, 83
- prg_allsumrealreduceparallel
 - prg_parallel_mod, 83
- prg_balanceparts
 - prg_subgraphloop_mod, 145
- prg_barrierparallel
 - prg_parallel_mod, 83
- prg_bcastparallel
 - prg_parallel_mod, 84
- prg_build_atomic_density
 - prg_densitymatrix_mod, 35
- prg_build_density_cheb
 - prg_chebyshev_mod, 27
- prg_build_density_cheb_fermi
 - prg_chebyshev_mod, 28
- prg_build_density_t
 - prg_densitymatrix_mod, 35
- prg_build_density_t0
 - prg_densitymatrix_mod, 36
- prg_build_density_t_fermi
 - prg_densitymatrix_mod, 36
- prg_buildzdiag
 - prg_genz_mod, 46
- prg_buildzsparse
 - prg_genz_mod, 46
- prg_centeratbox
 - prg_system_mod, 152
- prg_charges_mod, 23
 - dp, 25
 - prg_get_charges, 23
 - prg_get_hscf, 24
- prg_chebyshev_mod, 25
 - absmaxderivative, 26
 - dp, 33
 - fermi, 26
 - jackson, 27
 - pi, 33
 - prg_build_density_cheb, 27
 - prg_build_density_cheb_fermi, 28
 - prg_get_chebcoeffs, 29
 - prg_get_chebcoeffs_fermi_bs, 30
 - prg_get_chebcoeffs_fermi_nt, 31
 - prg_parse_cheb, 32
 - tr, 32
- prg_chebyshev_mod::chebdata_type, 181
 - atr, 182
 - bml_type, 182
 - bndfil, 182
 - ef, 182
 - estep, 182
 - fermitol, 182
 - flavor, 182
 - getef, 182
 - jobname, 182
 - jon, 182
 - kbt, 182
 - mdim, 183
 - ncoeffs, 183
 - ndim, 183
 - npts, 183

- threshold, 183
- trkfunc, 183
- verbose, 183
- prg_check_arrays
 - prg_partition_mod, 95
- prg_check_idempotency
 - prg_densitymatrix_mod, 37
- prg_check_valid
 - prg_kernelparser_mod, 65
- prg_collect_graph_p
 - prg_system_mod, 152
- prg_collectmatrixfromparts
 - prg_subgraphloop_mod, 145
- prg_compute_dipole
 - prg_response_mod, 126
- prg_compute_polarizability
 - prg_response_mod, 126
- prg_compute_response_fd
 - prg_response_mod, 127
- prg_compute_response_rs
 - prg_response_mod, 127
- prg_compute_response_sp2
 - prg_response_mod, 127
- prg_costindex
 - prg_partition_mod, 95
- prg_costpartition
 - prg_partition_mod, 96
- prg_delta
 - prg_extras_mod, 42
- prg_densitymatrix_mod, 33
 - dp, 38
 - fermi, 34
 - prg_build_atomic_density, 35
 - prg_build_density_t, 35
 - prg_build_density_t0, 36
 - prg_build_density_t_fermi, 36
 - prg_check_idempotency, 37
 - prg_get_eigenvalues, 37
 - prg_get_flevel, 38
- prg_deorthogonalize
 - prg_nonortho_mod, 68
- prg_destroy_subsystems
 - prg_system_mod, 153
- prg_destroygraphpartitioning
 - prg_graph_mod, 51
- prg_destroysubgraph
 - prg_graph_mod, 52
- prg_dos_mod, 39
 - dp, 40
 - lorentz, 39
 - prg_write_tdos, 40
- prg_eig
 - prg_xlkernel_mod, 177
- prg_equalgrouppartition
 - prg_graph_mod, 53
- prg_equalpartition
 - prg_graph_mod, 54
- prg_excitation
 - prg_quantumdynamics_mod, 122
- prg_extras_mod, 41
 - dp, 44
 - mls, 41
 - prg_delta, 42
 - prg_get_mem, 42
 - prg_print_matrix, 43
 - prg_twonorm, 43
 - to_string_double, 43
 - to_string_integer, 44
 - to_string_long_long, 44
- prg_extras_mod::prg_memory_consumption, 200
 - prg_memory_consumption, 201
- prg_extras_mod::to_string, 217
 - to_string_double, 217
 - to_string_integer, 218
 - to_string_long_long, 218
- prg_fermi
 - prg_xlkernel_mod, 177
- prg_filepartition
 - prg_graph_mod, 54
- prg_find_best_move
 - prg_partition_mod, 97
- prg_fnormgraph
 - prg_graph_mod, 55
- prg_generate
 - prg_genz_mod, 47
 - prg_genz_mod::genzspdata, 187
- prg_genz_mod, 45
 - dp, 50
 - prg_allocatezspmat, 45
 - prg_buildzdiag, 46
 - prg_buildzsparse, 46
 - prg_generate, 47
 - prg_genz_sp_initial_zmat, 48
 - prg_genz_sp_initialz0, 48
 - prg_genz_sp_int, 48
 - prg_genz_sp_ref, 49
 - prg_init, 49
 - prg_init_zspmat, 50
 - prg_parse_zsp, 50
- prg_genz_mod::genzspdata, 187
 - integration, 187
 - nfirst, 187
 - nreff, 188
 - nrefi, 188
 - numthresf, 188
 - numthresi, 188
 - prg_allocatezspmat, 187
 - prg_generate, 187
 - prg_init, 187
 - verbose, 188
- prg_genz_mod::genzspinp, 188
 - bml_type, 189
 - igenz, 189
 - integration, 189
 - mdim, 190
 - nfirst, 190

- nreff, 190
- nrefi, 190
- numthresf, 190
- numthresi, 190
- verbose, 190
- zsp, 190
- prg_genz_sp_initial_zmat
 - prg_genz_mod, 48
- prg_genz_sp_initialz0
 - prg_genz_mod, 48
- prg_genz_sp_int
 - prg_genz_mod, 48
- prg_genz_sp_ref
 - prg_genz_mod, 49
- prg_gershgorinreduction
 - prg_normalize_mod, 70
- prg_get_charges
 - prg_charges_mod, 23
- prg_get_chebcoeffs
 - prg_chebyshev_mod, 29
- prg_get_chebcoeffs_fermi_bs
 - prg_chebyshev_mod, 30
- prg_get_chebcoeffs_fermi_nt
 - prg_chebyshev_mod, 31
- prg_get_covgraph
 - prg_system_mod, 153
- prg_get_covgraph_h
 - prg_system_mod, 154
- prg_get_covgraph_int
 - prg_system_mod, 154
- prg_get_deriv_finite_temp
 - prg_xlkernel_mod, 178
- prg_get_dihedral
 - prg_system_mod, 154
- prg_get_distancematrix
 - prg_system_mod, 154
- prg_get_eigenvalues
 - prg_densitymatrix_mod, 37
- prg_get_flevel
 - prg_densitymatrix_mod, 38
- prg_get_hscf
 - prg_charges_mod, 24
- prg_get_largest_hedge_in_part
 - prg_partition_mod, 98
- prg_get_mem
 - prg_extras_mod, 42
- prg_get_nameandext
 - prg_system_mod, 155
- prg_get_origin
 - prg_system_mod, 155
- prg_get_partial_atomgraph
 - prg_system_mod, 156
- prg_get_pulayforce
 - prg_pulaycomponent_mod, 117
- prg_get_recip_vects
 - prg_system_mod, 156
- prg_get_sparsity_cplxmat
 - prg_quantumdynamics_mod, 122
- prg_get_sparsity_realmat
 - prg_quantumdynamics_mod, 122
- prg_get_subsystem
 - prg_system_mod, 156
- prg_getcharge
 - prg_quantumdynamics_mod, 122
- prg_getdipole
 - prg_quantumdynamics_mod, 123
- prg_getgrouppartitionhalosfromgraph
 - prg_subgraphloop_mod, 146
- prg_getpartitionhalosfromgraph
 - prg_subgraphloop_mod, 147
- prg_graph2bml
 - prg_system_mod, 157
- prg_graph2vector
 - prg_system_mod, 157
- prg_graph_mod, 51
 - dp, 59
 - prg_destroygraphpartitioning, 51
 - prg_destroysubgraph, 52
 - prg_equalgrouppartition, 53
 - prg_equalpartition, 54
 - prg_filepartition, 54
 - prg_fnormgraph, 55
 - prg_initgraphpartitioning, 55
 - prg_initsubgraph, 57
 - prg_printgraphpartitioning, 58
 - prg_readpart, 58
- prg_graph_mod::graph_partitioning_t, 191
 - ehomo, 192
 - elumo, 192
 - globalpartextent, 193
 - globalpartmax, 193
 - globalpartmin, 193
 - localpartextent, 193
 - localpartmax, 193
 - localpartmin, 193
 - maxeval, 193
 - maxiter, 193
 - mineval, 194
 - myrank, 194
 - nnodesinpart, 194
 - nnodesinpartall, 194
 - nparts, 194
 - order, 194
 - pname, 194
 - pp, 194
 - reorder, 195
 - sgraph, 195
 - totalnodes, 195
 - totalnodes2, 195
 - totalparts, 195
 - totalprocs, 195
 - vv, 195
- prg_graph_mod::subgraph_t, 208
 - core_halo_index, 208
 - hsize, 208
 - llsize, 208

- lsize, 208
- nodeinpart, 209
- part, 209
- prg_graphsp2parser_mod, 59
 - dp, 60
 - prg_parse_gsp2, 60
- prg_graphsp2parser_mod::gsp2data_type, 196
 - bml_type, 197
 - bndfil, 197
 - covgfact, 197
 - double_jump, 197
 - errlimit, 197
 - graph_element, 197
 - gthreshold, 197
 - hamfile, 197
 - jobname, 197
 - maxsp2iter, 197
 - mdim, 197
 - minsp2iter, 198
 - natoms, 198
 - ndim, 198
 - nlgcut, 198
 - nodesperpart, 198
 - parteach, 198
 - partition_count, 198
 - partition_refinement, 198
 - partition_type, 198
 - pdim, 198
 - sdim, 199
 - sp2conv, 199
 - sp2tol, 199
 - threshold, 199
 - verbose, 199
- prg_homolumo_mod, 61
 - dp, 61
 - prg_homolumogap, 61
 - prg_sp2sequence, 61
- prg_homolumogap
 - prg_homolumo_mod, 61
- prg_implicit_fermi
 - prg_implicit_fermi_mod, 62
- prg_implicit_fermi_mod, 61
 - dp, 62
 - prg_implicit_fermi, 62
- prg_init
 - prg_genz_mod, 49
 - prg_genz_mod::genzspdata, 187
- prg_init_hsmat
 - prg_initmatrices_mod, 63
- prg_init_ortho
 - prg_initmatrices_mod, 63
- prg_init_pzmat
 - prg_initmatrices_mod, 64
- prg_init_zspmat
 - prg_genz_mod, 50
- prg_initgraphpartitioning
 - prg_graph_mod, 55
- prg_initmatrices_mod, 62
 - dp, 64
 - prg_init_hsmat, 63
 - prg_init_ortho, 63
 - prg_init_pzmat, 64
- prg_initparallel
 - prg_parallel_mod, 84
- prg_initsubgraph
 - prg_graph_mod, 57
- prg_inv
 - prg_xlkernel_mod, 178
- prg_iprg_recvparallel
 - prg_parallel_mod, 84
- prg_kernel_fermi_full
 - prg_xlkernel_mod, 178
- prg_kernelparser_mod, 64
 - dp, 67
 - prg_check_valid, 65
 - prg_parsing_kernel, 66
- prg_kernlin
 - prg_partition_mod, 98
- prg_kernlin2
 - prg_partition_mod, 100
- prg_kernlin_queue
 - prg_partition_mod, 101
- prg_kick_density
 - prg_quantumdynamics_mod, 123
- prg_kick_density_bml
 - prg_quantumdynamics_mod, 124
- prg_linearmixer
 - prg_pulaymixer_mod, 119
- prg_lvni_bml
 - prg_quantumdynamics_mod, 124
- prg_make_random_system
 - prg_system_mod, 157
- prg_maxintreduce2
 - prg_parallel_mod, 84
- prg_maxrealreduce
 - prg_parallel_mod, 85
- prg_memory_consumption
 - prg_extras_mod::prg_memory_consumption, 201
- prg_merge_graph
 - prg_system_mod, 158
- prg_merge_graph_adj
 - prg_system_mod, 158
- prg_metispartition
 - prg_partition_mod, 102
- prg_minrealreduce
 - prg_parallel_mod, 85
- prg_mmult
 - prg_xlkernel_mod, 178
- prg_molpartition
 - prg_system_mod, 158
- prg_nonortho_mod, 67
 - dp, 69
 - prg_deorthogonalize, 68
 - prg_orthogonalize, 69
- prg_normalize
 - prg_normalize_mod, 70

- prg_normalize_cheb
 - prg_normalize_mod, 71
- prg_normalize_fermi
 - prg_normalize_mod, 72
- prg_normalize_implicit_fermi
 - prg_normalize_mod, 73
- prg_normalize_mod, 69
 - dp, 73
 - prg_gershgorinreduction, 70
 - prg_normalize, 70
 - prg_normalize_cheb, 71
 - prg_normalize_fermi, 72
 - prg_normalize_implicit_fermi, 73
- prg_open_file
 - prg_openfiles_mod, 75
- prg_open_file_to_read
 - prg_openfiles_mod, 76
- prg_openfiles_mod, 74
 - get_file_unit, 74
 - prg_open_file, 75
 - prg_open_file_to_read, 76
- prg_orthogonalize
 - prg_nonortho_mod, 69
- prg_parallel_mod, 77
 - allgatherintparallel, 78
 - allgatherrealparallel, 78
 - allgathervintparallel, 79
 - allgathervrealparallel, 79
 - dp, 92
 - getmyrank, 79
 - getnranks, 79
 - ierr, 92
 - isendparallel, 80
 - maxintparallel, 80
 - maxrankrealparallel, 81
 - maxrealparallel, 81
 - minintparallel, 81
 - minrankrealparallel, 81
 - minrealparallel, 82
 - myrank, 92
 - nranks, 92
 - prg_allgatherparallel, 82
 - prg_allsumintreduceparallel, 83
 - prg_allsumrealreduceparallel, 83
 - prg_barrierparallel, 83
 - prg_bcastparallel, 84
 - prg_initparallel, 84
 - prg_iprg_recvparallel, 84
 - prg_maxintreduce2, 84
 - prg_maxrealreduce, 85
 - prg_minrealreduce, 85
 - prg_recvparallel, 86
 - prg_shutdownparallel, 86
 - prg_sumintreduce2, 86
 - prg_sumintreducen, 87
 - prg_sumrealreduce, 87
 - prg_sumrealreduce2, 87
 - prg_sumrealreduce3, 88
 - prg_sumrealreducen, 89
 - prg_wait, 89
 - printrank, 89
 - reqcount, 92
 - requestlist, 93
 - rused, 93
 - saverequest, 90
 - sendparallel, 91
 - sendreceiveparallel, 91
 - sumintparallel, 91
 - sumrealparallel, 91
- prg_parallel_mod::rankreducedata_t, 201
 - rank, 201
 - val, 201
- prg_parameters_to_vectors
 - prg_system_mod, 159
- prg_parse_cheb
 - prg_chebyshev_mod, 32
- prg_parse_gsp2
 - prg_graphsp2parser_mod, 60
- prg_parse_mixer
 - prg_pulaymixer_mod, 120
- prg_parse_response
 - prg_response_mod, 128
- prg_parse_rotation
 - prg_syrotation_mod, 149
- prg_parse_sp2
 - prg_sp2parser_mod, 144
- prg_parse_system
 - prg_system_mod, 159
- prg_parse_xlbo
 - prg_xlbo_mod, 175
- prg_parse_xlkernel
 - prg_xlkernel_mod, 178
- prg_parse_zsp
 - prg_genz_mod, 50
- prg_parsing_kernel
 - prg_kernelparser_mod, 66
- prg_partition_mod, 93
 - dp, 108
 - metis_index_kind, 108
 - metis_real_kind, 108
 - prg_accept_prob, 94
 - prg_check_arrays, 95
 - prg_costindex, 95
 - prg_costpartition, 96
 - prg_find_best_move, 97
 - prg_get_largest_hedge_in_part, 98
 - prg_kernlin, 98
 - prg_kernlin2, 100
 - prg_kernlin_queue, 101
 - prg_metispartition, 102
 - prg_rand_node, 103
 - prg_rand_shuffle, 103
 - prg_simannealing, 104
 - prg_simannealing_old, 105
 - prg_update_gp, 106
 - update_prg_costpartition, 107

- prg_partordering
 - prg_subgraphloop_mod, 147
- prg_pert_constant_field
 - prg_response_mod, 128
- prg_pert_cos_pot
 - prg_response_mod, 129
- prg_pert_from_file
 - prg_response_mod, 129
- prg_pert_sin_pot
 - prg_response_mod, 129
- prg_prg_sp2_alg1_seq_inplace
 - prg_sp2_mod, 138
- prg_prg_sp2_alg2_seq_inplace
 - prg_sp2_mod, 138
- prg_print_date_and_time
 - prg_timer_mod, 166
- prg_print_matrix
 - prg_extras_mod, 43
- prg_printgraphpartitioning
 - prg_graph_mod, 58
- prg_progress_init
 - prg_progress_mod, 109
- prg_progress_mod, 108
 - dp, 110
 - prg_progress_init, 109
 - prg_progress_shutdown, 109
- prg_progress_shutdown
 - prg_progress_mod, 109
- prg_project_response
 - prg_response_mod, 130
- prg_ptable_mod, 110
 - atom_en, 113
 - dp, 113
 - element_atomic_number, 113
 - element_atomic_number_upper, 113
 - element_covr, 114
 - element_ea, 114
 - element_econf, 114
 - element_ip, 114
 - element_mass, 115
 - element_maxbonds, 115
 - element_name, 115
 - element_numel, 115
 - element_symbol, 116
 - element_symbol_upper, 116
 - element_vdwr, 116
 - nz, 116
- prg_pulaycomponent0
 - prg_pulaycomponent_mod, 117
- prg_pulaycomponent_mod, 117
 - dp, 118
 - prg_get_pulayforce, 117
 - prg_pulaycomponent0, 117
 - prg_pulaycomponentt, 118
- prg_pulaycomponentt
 - prg_pulaycomponent_mod, 118
- prg_pulaymixer_mod, 119
 - dp, 120
 - prg_linearmixer, 119
 - prg_parse_mixer, 120
 - prg_qmixer, 120
- prg_pulaymixer_mod::mx_type, 199
 - mixcoeff, 200
 - mixeron, 200
 - mixertype, 200
 - mpulay, 200
 - verbose, 200
- prg_qmixer
 - prg_pulaymixer_mod, 120
- prg_quantumdynamics_mod, 121
 - dp, 125
 - prg_excitation, 122
 - prg_get_sparsity_cplxmat, 122
 - prg_get_sparsity_realmat, 122
 - prg_getcharge, 122
 - prg_getdipole, 123
 - prg_kick_density, 123
 - prg_kick_density_bml, 124
 - prg_lvni_bml, 124
- prg_rand_node
 - prg_partition_mod, 103
- prg_rand_shuffle
 - prg_partition_mod, 103
- prg_rank1
 - prg_xlkernel_mod, 179
- prg_readpart
 - prg_graph_mod, 58
- prg_recvparallel
 - prg_parallel_mod, 86
- prg_replicate
 - prg_system_mod, 160
- prg_response_mod, 125
 - dp, 131
 - pi, 131
 - prg_compute_dipole, 126
 - prg_compute_polarizability, 126
 - prg_compute_response_fd, 127
 - prg_compute_response_rs, 127
 - prg_compute_response_sp2, 127
 - prg_parse_response, 128
 - prg_pert_constant_field, 128
 - prg_pert_cos_pot, 129
 - prg_pert_from_file, 129
 - prg_pert_sin_pot, 129
 - prg_project_response, 130
 - prg_write_dipole_tcl, 130
- prg_response_mod::respdata_type, 202
 - bmltype, 202
 - computedipole, 202
 - field, 202
 - fieldintensity, 202
 - getresponse, 202
 - mdim, 203
 - numthresh, 203
 - respmode, 203
 - typeofpert, 203

- prg_rotate
 - prg_syrotation_mod, 149
- prg_shutdownparallel
 - prg_parallel_mod, 86
- prg_simannealing
 - prg_partition_mod, 104
- prg_simannealing_old
 - prg_partition_mod, 105
- prg_sortadj
 - prg_system_mod, 161
- prg_sp2_alg1
 - prg_sp2_mod, 138
- prg_sp2_alg1_genseq
 - prg_sp2_mod, 139
- prg_sp2_alg1_seq
 - prg_sp2_mod, 139
- prg_sp2_alg2
 - prg_sp2_mod, 140
- prg_sp2_alg2_genseq
 - prg_sp2_mod, 140
- prg_sp2_alg2_seq
 - prg_sp2_mod, 141
- prg_sp2_basic
 - prg_sp2_mod, 141
- prg_sp2_entropy_function
 - prg_sp2_fermi_mod, 133
- prg_sp2_fermi
 - prg_sp2_fermi_mod, 133
- prg_sp2_fermi_init
 - prg_sp2_fermi_mod, 134
- prg_sp2_fermi_init_norecs
 - prg_sp2_fermi_mod, 135
- prg_sp2_fermi_mod, 131
 - absmaxderivative, 132
 - dp, 137
 - prg_sp2_entropy_function, 133
 - prg_sp2_fermi, 133
 - prg_sp2_fermi_init, 134
 - prg_sp2_fermi_init_norecs, 135
 - sp2_entropy_ts, 136
 - sp2_inverse, 136
- prg_sp2_mod, 137
 - dp, 143
 - prg_prg_sp2_alg1_seq_inplace, 138
 - prg_prg_sp2_alg2_seq_inplace, 138
 - prg_sp2_alg1, 138
 - prg_sp2_alg1_genseq, 139
 - prg_sp2_alg1_seq, 139
 - prg_sp2_alg2, 140
 - prg_sp2_alg2_genseq, 140
 - prg_sp2_alg2_seq, 141
 - prg_sp2_basic, 141
 - prg_sp2_submatrix, 142
 - prg_sp2_submatrix_inplace, 143
- prg_sp2_submatrix
 - prg_sp2_mod, 142
- prg_sp2_submatrix_inplace
 - prg_sp2_mod, 143
- prg_sp2parser_mod, 143
 - dp, 144
 - prg_parse_sp2, 144
- prg_sp2parser_mod::sp2data_type, 205
 - bml_type, 206
 - bndfil, 206
 - flavor, 206
 - jobname, 206
 - maxsp2iter, 206
 - mdim, 206
 - minsp2iter, 207
 - ndim, 207
 - pdim, 207
 - sdim, 207
 - sp2conv, 207
 - sp2tol, 207
 - threshold, 207
 - verbose, 207
- prg_sp2sequence
 - prg_homolumo_mod, 61
- prg_subgraphloop_mod, 144
 - dp, 148
 - prg_balanceparts, 145
 - prg_collectmatrixfromparts, 145
 - prg_getgrouppartitionhalosfromgraph, 146
 - prg_getpartitionhalosfromgraph, 147
 - prg_partordering, 147
 - prg_subgraphsp2loop, 147
- prg_subgraphsp2loop
 - prg_subgraphloop_mod, 147
- prg_sumintreduce2
 - prg_parallel_mod, 86
- prg_sumintreducen
 - prg_parallel_mod, 87
- prg_sumrealreduce
 - prg_parallel_mod, 87
- prg_sumrealreduce2
 - prg_parallel_mod, 87
- prg_sumrealreduce3
 - prg_parallel_mod, 88
- prg_sumrealreducen
 - prg_parallel_mod, 89
- prg_syrotation_mod, 148
 - dp, 150
 - prg_parse_rotation, 149
 - prg_rotate, 149
- prg_syrotation_mod::rotation_type, 203
 - catom, 204
 - catom2, 204
 - jobname, 204
 - patom1, 204
 - patom2, 204
 - pq1, 204
 - pq2, 204
 - rotate_atoms, 205
 - typeofrot, 205
 - v1, 205
 - v2, 205

- vq, 205
- prg_system_mod, 150
 - dp, 165
 - prg_adj2bml, 152
 - prg_centeratbox, 152
 - prg_collect_graph_p, 152
 - prg_destroy_subsystems, 153
 - prg_get_covgraph, 153
 - prg_get_covgraph_h, 154
 - prg_get_covgraph_int, 154
 - prg_get_dihedral, 154
 - prg_get_distancematrix, 154
 - prg_get_nameandext, 155
 - prg_get_origin, 155
 - prg_get_partial_atomgraph, 156
 - prg_get_recip_vects, 156
 - prg_get_subsystem, 156
 - prg_graph2bml, 157
 - prg_graph2vector, 157
 - prg_make_random_system, 157
 - prg_merge_graph, 158
 - prg_merge_graph_adj, 158
 - prg_molpartition, 158
 - prg_parameters_to_vectors, 159
 - prg_parse_system, 159
 - prg_replicate, 160
 - prg_sortadj, 161
 - prg_translateandfoldtobox, 161
 - prg_translatetogeomcandfoldtobox, 161
 - prg_vector2graph, 161
 - prg_vectors_to_parameters, 162
 - prg_wraparound, 162
 - prg_write_system, 162
 - prg_write_trajectory, 163
 - prg_write_trajectoryandproperty, 164
- prg_system_mod::estruct_type, 183
 - coul_pot_k, 184
 - coul_pot_r, 184
 - eband, 185
 - fpul, 185
 - fscoul, 185
 - ham, 185
 - ham0, 185
 - hindex, 185
 - nel, 185
 - norbs, 185
 - oham, 186
 - orho, 186
 - over, 186
 - rho, 186
 - skforce, 186
 - zmat, 186
- prg_system_mod::system_type, 209
 - atomic_number, 211
 - coordinate, 211
 - estr, 211
 - force, 211
 - lattice_vector, 211
 - mass, 211
 - nats, 212
 - net_charge, 212
 - nsp, 212
 - recip_vector, 212
 - resindex, 212
 - spatnum, 213
 - spindex, 213
 - splist, 213
 - spmass, 213
 - symbol, 213
 - userdef, 214
 - velocity, 214
 - volk, 214
 - volr, 214
- prg_timer_collect
 - prg_timer_mod, 167
- prg_timer_getid
 - prg_timer_mod, 167
- prg_timer_mod, 165
 - badd_timer, 170
 - bmult_timer, 170
 - buildz_timer, 171
 - deortho_timer, 171
 - dp, 171
 - dyn_timer, 171
 - genx_timer, 171
 - graphsp2_timer, 171
 - halfverlet_timer, 171
 - int2char, 166
 - loop_timer, 171
 - mdloop_timer, 171
 - nlist_timer, 171
 - num_timers, 172
 - ortho_timer, 172
 - pairpot_timer, 172
 - part_timer, 172
 - pos_timer, 172
 - prg_print_date_and_time, 166
 - prg_timer_collect, 167
 - prg_timer_getid, 167
 - prg_timer_results, 168
 - prg_timer_shutdown, 168
 - prg_timer_start, 169
 - prg_timer_stop, 169
 - ptimer, 172
 - realcoul_timer, 172
 - recipcoul_timer, 172
 - sp2_timer, 172
 - suball_timer, 172
 - subext_timer, 173
 - subgraph_timer, 173
 - subind_timer, 173
 - subsp2_timer, 173
 - tclock_max, 173
 - tclock_rate, 173
 - time2milliseconds, 170
 - timer_prg_init, 170

- tstart_clock, [173](#)
- tstop_clock, [173](#)
- zdiag_timer, [173](#)
- prg_timer_mod::timer_status_t, [215](#)
 - maxrank, [215](#)
 - maxvalue, [215](#)
 - minrank, [216](#)
 - minvalue, [216](#)
 - tavg, [216](#)
 - tcount, [216](#)
 - tname, [216](#)
 - tpercent, [216](#)
 - tstart, [216](#)
 - tstdev, [216](#)
 - tsum, [217](#)
 - ttotal, [217](#)
- prg_timer_results
 - prg_timer_mod, [168](#)
- prg_timer_shutdown
 - prg_timer_mod, [168](#)
- prg_timer_start
 - prg_timer_mod, [169](#)
- prg_timer_stop
 - prg_timer_mod, [169](#)
- prg_translateandfoldtobox
 - prg_system_mod, [161](#)
- prg_translatetogeomcandfoldtobox
 - prg_system_mod, [161](#)
- prg_twonorm
 - prg_extras_mod, [43](#)
- prg_update_gp
 - prg_partition_mod, [106](#)
- prg_v_kernel_fermi
 - prg_xlkernel_mod, [179](#)
- prg_vector2graph
 - prg_system_mod, [161](#)
- prg_vectors_to_parameters
 - prg_system_mod, [162](#)
- prg_wait
 - prg_parallel_mod, [89](#)
- prg_wraparound
 - prg_system_mod, [162](#)
- prg_write_dipole_tcl
 - prg_response_mod, [130](#)
- prg_write_system
 - prg_system_mod, [162](#)
- prg_write_tdos
 - prg_dos_mod, [40](#)
- prg_write_trajectory
 - prg_system_mod, [163](#)
- prg_write_trajectoryandproperty
 - prg_system_mod, [164](#)
- prg_xlbo_fcoulupdate
 - prg_xlbo_mod, [175](#)
- prg_xlbo_mod, [174](#)
 - alpha, [175](#)
 - c0, [176](#)
 - c1, [176](#)
 - c2, [176](#)
 - c3, [176](#)
 - c4, [176](#)
 - c5, [176](#)
 - cc, [176](#)
 - dp, [176](#)
 - kappa, [176](#)
 - prg_parse_xlbo, [175](#)
 - prg_xlbo_fcoulupdate, [175](#)
 - prg_xlbo_nint, [175](#)
- prg_xlbo_mod::xlbo_type, [218](#)
 - cc, [219](#)
 - jobname, [219](#)
 - maxscfinititer, [219](#)
 - maxscfiter, [219](#)
 - minit, [219](#)
 - threshold, [220](#)
 - verbose, [220](#)
- prg_xlbo_nint
 - prg_xlbo_mod, [175](#)
- prg_xlkernel_mod, [177](#)
 - dp, [180](#)
 - prg_eig, [177](#)
 - prg_fermi, [177](#)
 - prg_get_deriv_finite_temp, [178](#)
 - prg_inv, [178](#)
 - prg_kernel_fermi_full, [178](#)
 - prg_mmult, [178](#)
 - prg_parse_xlkernel, [178](#)
 - prg_rank1, [179](#)
 - prg_v_kernel_fermi, [179](#)
- prg_xlkernel_mod::xlk_type, [220](#)
 - kerneltype, [220](#)
 - nrank, [220](#)
 - scalecoeff, [220](#)
 - verbose, [221](#)
- printrank
 - prg_parallel_mod, [89](#)
- ptimer
 - prg_timer_mod, [172](#)
- rank
 - prg_parallel_mod::rankreducedata_t, [201](#)
- realcoul_timer
 - prg_timer_mod, [172](#)
- recip_vector
 - prg_system_mod::system_type, [212](#)
- recipcoul_timer
 - prg_timer_mod, [172](#)
- reorder
 - prg_graph_mod::graph_partitioning_t, [195](#)
- reqcount
 - prg_parallel_mod, [92](#)
- requestlist
 - prg_parallel_mod, [93](#)
- resindex
 - prg_system_mod::system_type, [212](#)
- respmode
 - prg_response_mod::respdata_type, [203](#)

- rho
 - prg_system_mod::estruct_type, 186
- rotate_atoms
 - prg_syrotation_mod::rotation_type, 205
- rused
 - prg_parallel_mod, 93
- saverequest
 - prg_parallel_mod, 90
- scalecoeff
 - prg_xlkernel_mod::xlk_type, 220
- sdim
 - prg_graphsp2parser_mod::gsp2data_type, 199
 - prg_sp2parser_mod::sp2data_type, 207
- sendparallel
 - prg_parallel_mod, 91
- sendreceiveparallel
 - prg_parallel_mod, 91
- sgraph
 - prg_graph_mod::graph_partitioning_t, 195
- skforce
 - prg_system_mod::estruct_type, 186
- sp2_entropy_ts
 - prg_sp2_fermi_mod, 136
- sp2_inverse
 - prg_sp2_fermi_mod, 136
- sp2_timer
 - prg_timer_mod, 172
- sp2conv
 - prg_graphsp2parser_mod::gsp2data_type, 199
 - prg_sp2parser_mod::sp2data_type, 207
- sp2tol
 - prg_graphsp2parser_mod::gsp2data_type, 199
 - prg_sp2parser_mod::sp2data_type, 207
- spatnum
 - prg_system_mod::system_type, 213
- spindex
 - prg_system_mod::system_type, 213
- splist
 - prg_system_mod::system_type, 213
- spmass
 - prg_system_mod::system_type, 213
- suball_timer
 - prg_timer_mod, 172
- subext_timer
 - prg_timer_mod, 173
- subgraph_timer
 - prg_timer_mod, 173
- subind_timer
 - prg_timer_mod, 173
- subsp2_timer
 - prg_timer_mod, 173
- sumintparallel
 - prg_parallel_mod, 91
- sumrealparallel
 - prg_parallel_mod, 91
- symbol
 - prg_system_mod::system_type, 213
- tavg
 - prg_timer_mod::timer_status_t, 216
- tclock_max
 - prg_timer_mod, 173
- tclock_rate
 - prg_timer_mod, 173
- tcount
 - prg_timer_mod::timer_status_t, 216
- threshold
 - prg_chebyshev_mod::chebdata_type, 183
 - prg_graphsp2parser_mod::gsp2data_type, 199
 - prg_sp2parser_mod::sp2data_type, 207
 - prg_xlbo_mod::xlbo_type, 220
- time2milliseconds
 - prg_timer_mod, 170
- timer_prg_init
 - prg_timer_mod, 170
- tname
 - prg_timer_mod::timer_status_t, 216
- to_string_double
 - prg_extras_mod, 43
 - prg_extras_mod::to_string, 217
- to_string_integer
 - prg_extras_mod, 44
 - prg_extras_mod::to_string, 218
- to_string_long_long
 - prg_extras_mod, 44
 - prg_extras_mod::to_string, 218
- totalnodes
 - prg_graph_mod::graph_partitioning_t, 195
- totalnodes2
 - prg_graph_mod::graph_partitioning_t, 195
- totalparts
 - prg_graph_mod::graph_partitioning_t, 195
- totalprocs
 - prg_graph_mod::graph_partitioning_t, 195
- tpercent
 - prg_timer_mod::timer_status_t, 216
- tr
 - prg_chebyshev_mod, 32
- trkfunc
 - prg_chebyshev_mod::chebdata_type, 183
- tstart
 - prg_timer_mod::timer_status_t, 216
- tstart_clock
 - prg_timer_mod, 173
- tstdev
 - prg_timer_mod::timer_status_t, 216
- tstop_clock
 - prg_timer_mod, 173
- tsum
 - prg_timer_mod::timer_status_t, 217
- ttotal
 - prg_timer_mod::timer_status_t, 217
- typeofpert
 - prg_response_mod::respdata_type, 203
- typeofrot
 - prg_syrotation_mod::rotation_type, 205

- update_prg_costpartition
 - prg_partition_mod, [107](#)
- userdef
 - prg_system_mod::system_type, [214](#)
- v1
 - prg_syrotation_mod::rotation_type, [205](#)
- v2
 - prg_syrotation_mod::rotation_type, [205](#)
- val
 - prg_parallel_mod::rankreducedata_t, [201](#)
- velocity
 - prg_system_mod::system_type, [214](#)
- verbose
 - prg_chebyshev_mod::chebdata_type, [183](#)
 - prg_genz_mod::genzspdata, [188](#)
 - prg_genz_mod::genzspinp, [190](#)
 - prg_graphsp2parser_mod::gsp2data_type, [199](#)
 - prg_pulaymixer_mod::mx_type, [200](#)
 - prg_sp2parser_mod::sp2data_type, [207](#)
 - prg_xlbo_mod::xlbo_type, [220](#)
 - prg_xlkernel_mod::xlk_type, [221](#)
- volk
 - prg_system_mod::system_type, [214](#)
- volr
 - prg_system_mod::system_type, [214](#)
- vq
 - prg_syrotation_mod::rotation_type, [205](#)
- vv
 - prg_graph_mod::graph_partitioning_t, [195](#)
- zdiag_timer
 - prg_timer_mod, [173](#)
- zmat
 - prg_system_mod::estruct_type, [186](#)
- zsp
 - prg_genz_mod::genzspinp, [190](#)