

DRS
1.7.0

Generated by Doxygen 1.6.1

Sat Feb 28 16:13:49 2015

Contents

1	Todo List	1
2	Namespace Index	3
2.1	Namespace List	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	CrankNicholson Namespace Reference	9
5.1.1	Detailed Description	10
5.1.2	Function Documentation	10
5.1.2.1	CrankNicholson_init	10
5.1.2.2	updateCrankNicholson_matrices	10
5.1.3	Variable Documentation	10
5.1.3.1	field_lap_inv_pol	10
5.1.3.2	field_lap_inv_tor	10
5.1.3.3	flow_lap_inv_pol	11
5.1.3.4	flow_lap_inv_tor	11
5.1.3.5	pinv	11
5.1.3.6	temp_lap_inv	11
5.2	drs_Chebyshev Namespace Reference	12
5.2.1	Detailed Description	12
5.2.2	Function Documentation	13
5.2.2.1	Cheb_compute_dx_ddx_n2x	13
5.2.2.2	Cheb_compute_dx_ddx_x2x	13
5.2.2.3	Cheb_compute_dx_n2n	14

5.2.2.4	Chebyshev_cleanup	14
5.2.2.5	Chebyshev_init	14
5.2.2.6	Chebyshev_n2x	15
5.2.2.7	Chebyshev_x2n	15
5.2.3	Variable Documentation	15
5.2.3.1	Cheb_x	15
5.2.3.2	Chebyshev	15
5.2.3.3	Chebyshev_ddx	15
5.2.3.4	Chebyshev_dx	16
5.2.3.5	ct_buffer	16
5.2.3.6	Nx	16
5.2.3.7	Nx_s	16
5.2.3.8	pi	16
5.2.3.9	plan_x	16
5.3	drs_comp Namespace Reference	17
5.3.1	Detailed Description	17
5.3.2	Function Documentation	18
5.3.2.1	apply_comp_BC	18
5.3.2.2	calc_comp	18
5.3.2.3	compProfName	18
5.3.2.4	drs_comp_allocation	18
5.3.2.5	drs_comp_init	18
5.3.2.6	drs_comp_randomize	19
5.3.2.7	drs_comp_reset	19
5.3.3	Variable Documentation	19
5.3.3.1	comp	19
5.3.3.2	comp_avg	19
5.3.3.3	comp_ddr	19
5.3.3.4	comp_dr	19
5.3.3.5	comp_dr_avg	19
5.3.3.6	comp_profile	20
5.3.3.7	comp_profile_dr	20
5.4	drs_debug Namespace Reference	21
5.4.1	Detailed Description	21
5.4.2	Function Documentation	21
5.4.2.1	save_lmr_quantity	21

5.4.2.2	save_tpr_quantity	21
5.5	drs_dims Namespace Reference	22
5.5.1	Detailed Description	22
5.5.2	Function Documentation	22
5.5.2.1	check_dims	22
5.5.2.2	drs_dims_init	23
5.5.3	Variable Documentation	23
5.5.3.1	lsymm	23
5.5.3.2	m0	23
5.5.3.3	Np	23
5.5.3.4	Np_s	23
5.5.3.5	Nr	24
5.5.3.6	Nr_s	24
5.5.3.7	Nt	24
5.5.3.8	Nt_s	25
5.5.3.9	usr_dims	25
5.6	drs_fftw3 Namespace Reference	26
5.6.1	Detailed Description	26
5.6.2	Function Documentation	27
5.6.2.1	cos_r2r_1_n2r	27
5.6.2.2	cos_r2r_1_r2n	27
5.6.2.3	dft_backward	27
5.6.2.4	dft_forward	27
5.6.2.5	drs_fftw3_cleanup	27
5.6.2.6	drs_fftw3_init	27
5.6.2.7	remesh	28
5.6.3	Variable Documentation	28
5.6.3.1	drs_fftw3_Np	28
5.6.3.2	drs_fftw3_Nr	28
5.6.3.3	drs_fftw3_Nt	28
5.6.3.4	in_p	28
5.6.3.5	inout_r	28
5.6.3.6	plan_pb	28
5.6.3.7	plan_pf	29
5.6.3.8	plan_r	29
5.7	drs_field Namespace Reference	30

5.7.1	Function Documentation	30
5.7.1.1	apply_field_pol_BC	30
5.7.1.2	apply_field_tor_BC	31
5.7.1.3	calc_B	31
5.7.1.4	calc_field	31
5.7.1.5	calc_field_lspec	32
5.7.1.6	calc_field_mspec	32
5.7.1.7	calc_field_nspec	32
5.7.1.8	calc_rot_field	32
5.7.1.9	drs_field_allocation	32
5.7.1.10	drs_field_init	33
5.7.1.11	drs_field_random_init	33
5.7.1.12	update_field_pol_lap	33
5.7.1.13	update_field_tor_lap	33
5.7.2	Variable Documentation	33
5.7.2.1	field_pol	33
5.7.2.2	field_pol_avg	34
5.7.2.3	field_pol_ddr	34
5.7.2.4	field_pol_dr	34
5.7.2.5	field_pol_lap	34
5.7.2.6	field_tor	34
5.7.2.7	field_tor_avg	34
5.7.2.8	field_tor_ddr	34
5.7.2.9	field_tor_dr	34
5.7.2.10	field_tor_lap	35
5.8	drs_flow Namespace Reference	36
5.8.1	Function Documentation	37
5.8.1.1	apply_flow_pol_BC	37
5.8.1.2	apply_flow_tor_BC	37
5.8.1.3	calc_flow	37
5.8.1.4	calc_flow_lspec	37
5.8.1.5	calc_flow_mspec	38
5.8.1.6	calc_flow_nspec	38
5.8.1.7	calc_rot_flow	38
5.8.1.8	calc_u	39
5.8.1.9	drs_flow_allocation	39

5.8.1.10	drs_flow_init	39
5.8.1.11	update_flow_pol_lap	39
5.8.1.12	update_flow_tor_lap	40
5.8.2	Variable Documentation	40
5.8.2.1	flow_pol	40
5.8.2.2	flow_pol_avg	40
5.8.2.3	flow_pol_ddr	40
5.8.2.4	flow_pol_dr	40
5.8.2.5	flow_pol_lap	40
5.8.2.6	flow_tor	41
5.8.2.7	flow_tor_avg	41
5.8.2.8	flow_tor_ddr	41
5.8.2.9	flow_tor_dr	41
5.8.2.10	flow_tor_lap	41
5.9	drs_hypDiff Namespace Reference	42
5.9.1	Function Documentation	42
5.9.1.1	drs_hypDiff_init	42
5.9.2	Variable Documentation	42
5.9.2.1	drs_want_hypDiff	42
5.9.2.2	hypDiff	42
5.10	drs_io Namespace Reference	43
5.10.1	Detailed Description	43
5.10.2	Function Documentation	43
5.10.2.1	drs_load_state	43
5.10.2.2	drs_open_output	44
5.10.2.3	dump_state	44
5.10.2.4	save_l_spec	45
5.10.2.5	save_m_spec	46
5.10.2.6	save_n_spec	46
5.10.2.7	save_state	46
5.10.3	Variable Documentation	47
5.10.3.1	deflate	47
5.10.3.2	inflate	47
5.10.3.3	io_calc_file_in	47
5.10.3.4	io_calc_file_out	47
5.11	drs_io_DX Namespace Reference	48

5.11.1	Function Documentation	48
5.11.1.1	save2DXscalar	48
5.11.1.2	save2DXvector	49
5.11.1.3	saveDXmeridional	49
5.11.1.4	saveDXmeridional3DVec	50
5.11.1.5	saveDXvolume	50
5.11.1.6	saveDXvolume3DVec	50
5.11.1.7	saveDXvolume_v2	51
5.11.2	Variable Documentation	51
5.11.2.1	cut_phi	51
5.11.2.2	cut_type	51
5.11.2.3	cut_z	51
5.11.2.4	where_to_cut	51
5.12	drs_io_par Namespace Reference	52
5.12.1	Detailed Description	53
5.12.2	Function Documentation	53
5.12.2.1	drs_read_conf	53
5.12.2.2	drs_read_conf_v2	53
5.12.2.3	read_input_par	53
5.12.2.4	write_parp	54
5.12.3	Variable Documentation	54
5.12.3.1	commenti	54
5.12.3.2	drifti	54
5.12.3.3	drs_calc_typei	54
5.12.3.4	etai	54
5.12.3.5	flowBCi	54
5.12.3.6	hi	54
5.12.3.7	lformi	54
5.12.3.8	lsymmi	54
5.12.3.9	m0i	54
5.12.3.10	magBCi	55
5.12.3.11	magic	55
5.12.3.12	MAGICC1	55
5.12.3.13	MAGICC2	55
5.12.3.14	MAGICC3	55
5.12.3.15	MAGICC4	55

5.12.3.16	MAGICC5	55
5.12.3.17	MAGICC6	55
5.12.3.18	MAGICC7	55
5.12.3.19	MAGICC9	55
5.12.3.20	magici	55
5.12.3.21	noise	55
5.12.3.22	Npi	55
5.12.3.23	Npi_s	55
5.12.3.24	Nri	56
5.12.3.25	Nri_s	56
5.12.3.26	Nti	56
5.12.3.27	Nti_s	56
5.12.3.28	Pmi	56
5.12.3.29	Pti	56
5.12.3.30	Ra_ti	56
5.12.3.31	sampling_ratei	56
5.12.3.32	stepmaxi	56
5.12.3.33	Tai	56
5.12.3.34	tempBCi	56
5.12.3.35	transienti	57
5.12.3.36	usr_dimsi	57
5.13	drs_io_units Namespace Reference	58
5.13.1	Detailed Description	58
5.13.2	Variable Documentation	58
5.13.2.1	unit_adv	58
5.13.2.2	unit_am	58
5.13.2.3	unit_cfl	58
5.13.2.4	unit_dissB	58
5.13.2.5	unit_dissu	59
5.13.2.6	unit_eb	59
5.13.2.7	unit_ek	59
5.13.2.8	unit_evp	59
5.13.2.9	unit_evt	59
5.13.2.10	unit_koeb	59
5.13.2.11	unit_koeu	59
5.13.2.12	unit_lspec	59

5.13.2.13	unit_mspec	59
5.13.2.14	unit_nspec	59
5.13.2.15	unit_nu	59
5.13.2.16	unit_t	60
5.13.2.17	unit_u_mid	60
5.13.2.18	unit_uaz	60
5.13.2.19	unit_ur	60
5.13.2.20	unit_uzon	60
5.14	drs_legendre Namespace Reference	61
5.14.1	Function Documentation	62
5.14.1.1	drs_legendre_allocation	62
5.14.1.2	drs_legendre_init	62
5.14.1.3	gauleg	62
5.14.1.4	initNormalization	62
5.14.1.5	legendre_init_new	63
5.14.2	Variable Documentation	63
5.14.2.1	costheta	63
5.14.2.2	dleg	63
5.14.2.3	leg_neg	63
5.14.2.4	leg_sin	63
5.14.2.5	legendre	64
5.14.2.6	llp1	64
5.14.2.7	pi	64
5.14.2.8	plmfac	64
5.14.2.9	sintheta	64
5.14.2.10	w	64
5.15	drs_lock Namespace Reference	65
5.15.1	Detailed Description	65
5.15.2	Function Documentation	65
5.15.2.1	add_lock	65
5.15.2.2	drs_lock_init	65
5.15.2.3	lockExists	66
5.15.2.4	rm_lock	66
5.15.3	Variable Documentation	66
5.15.3.1	lockFileName	66
5.15.3.2	lockFileUnit	66

5.16	drs_mpi Namespace Reference	67
5.16.1	Detailed Description	69
5.16.2	Function Documentation	69
5.16.2.1	blk_ts_start_init	69
5.16.2.2	distribute_in_m	69
5.16.2.3	drs_abort	69
5.16.2.4	drs_bcast_dble	69
5.16.2.5	drs_bcast_dble_scal	70
5.16.2.6	drs_bcast_int	70
5.16.2.7	drs_bcast_int_scal	70
5.16.2.8	drs_bcast_logical_scal	70
5.16.2.9	drs_gather_vars	70
5.16.2.10	drs_maximize_dble	70
5.16.2.11	drs_maximize_dble_scal	70
5.16.2.12	drs_minimize_dble	70
5.16.2.13	drs_minimize_dble_scal	70
5.16.2.14	drs_mpi_init	70
5.16.2.15	gather_from_m	71
5.16.2.16	mpi_cleanup	71
5.16.2.17	mpi_dims_init	71
5.16.2.18	sum_over_all_cpus_scal	71
5.16.2.19	sum_over_all_cpus_vect	71
5.16.2.20	transpos_phi2theta	71
5.16.2.21	transpos_theta2phi	72
5.16.2.22	wait_for_everyone	72
5.16.3	Variable Documentation	72
5.16.3.1	blk_ps_max_size	72
5.16.3.2	blk_ps_size	72
5.16.3.3	blk_ps_start	73
5.16.3.4	blk_t_max_size	73
5.16.3.5	blk_t_size	73
5.16.3.6	blk_t_start	73
5.16.3.7	blk_ts_start	73
5.16.3.8	mm	73
5.16.3.9	mpi_rank	74
5.16.3.10	mpi_size	74

5.17	drs_nonlinear Namespace Reference	75
5.17.1	Detailed Description	75
5.17.2	Function Documentation	76
5.17.2.1	drs_nonlinear_init	76
5.17.2.2	evaluate_real_space	76
5.17.2.3	rhs	76
5.17.2.4	save_stuff	77
5.17.3	Variable Documentation	78
5.17.3.1	cfl	78
5.17.3.2	field_p_t	78
5.17.3.3	field_r_t	78
5.17.3.4	field_t_t	78
5.17.3.5	flow_p_t	78
5.17.3.6	flow_r_t	79
5.17.3.7	flow_t_t	79
5.17.3.8	ncfl	79
5.17.3.9	rhs_IE_pol	79
5.17.3.10	rhs_IE_tor	79
5.17.3.11	rhs_NS_pol	79
5.17.3.12	rhs_NS_tor	79
5.17.3.13	rhs_TE	79
5.17.3.14	rot_field_p_t	79
5.17.3.15	rot_field_r_t	79
5.17.3.16	rot_field_t_t	79
5.17.3.17	rot_flow_p_t	80
5.17.3.18	rot_flow_r_t	80
5.17.3.19	rot_flow_t_t	80
5.17.3.20	temp_t	80
5.18	drs_probes Namespace Reference	81
5.18.1	Detailed Description	83
5.18.2	Function Documentation	83
5.18.2.1	average_unnormalised_field_l_spectrum	83
5.18.2.2	average_unnormalised_flow_l_spectrum	83
5.18.2.3	average_unnormalised_scalar_l_spectrum	83
5.18.2.4	c_integrate_r	83
5.18.2.5	check_resolution_Hartman	84

5.18.2.6	compute_advection	84
5.18.2.7	compute_helicities	84
5.18.2.8	drs_probes_allocation	84
5.18.2.9	drs_probes_init	84
5.18.2.10	energy	85
5.18.2.11	integrate_power_surf	85
5.18.2.12	integrate_r	85
5.18.2.13	l_spec_of_scalar_field	85
5.18.2.14	m_spec_of_scalar_field	86
5.18.2.15	measure	86
5.18.2.16	measure_lm	87
5.18.2.17	n_spec_of_scalar_field	87
5.18.2.18	nusselt	87
5.18.2.19	save_angular_momentum	88
5.18.2.20	save_field_coeffs	88
5.18.2.21	save_flow_coeffs	88
5.18.2.22	save_flow_dissipation	89
5.18.2.23	save_magnetic_dissipation	89
5.18.3	Variable Documentation	89
5.18.3.1	adv_avg	89
5.18.3.2	Bdrkea	90
5.18.3.3	Bdrkes	90
5.18.3.4	Betora	90
5.18.3.5	Betors	90
5.18.3.6	Bmckea	90
5.18.3.7	Bmckes	90
5.18.3.8	Bnkea	90
5.18.3.9	Bnkes	90
5.18.3.10	Brspec_avg	90
5.18.3.11	dOmega	90
5.18.3.12	drkea	90
5.18.3.13	drkes	91
5.18.3.14	EB	91
5.18.3.15	Ekin	91
5.18.3.16	etora	91
5.18.3.17	etors	91

5.18.3.18	groth	91
5.18.3.19	mckea	91
5.18.3.20	mckes	91
5.18.3.21	nkea	91
5.18.3.22	nkes	91
5.18.3.23	Rm	91
5.18.3.24	t2_avg	92
5.18.3.25	tspec_avg	92
5.18.3.26	up2	92
5.18.3.27	up_avg	92
5.18.3.28	ur_avg	92
5.18.3.29	urspec_avg	92
5.18.3.30	ut2	92
5.18.3.31	ut_avg	92
5.19	drs_radial Namespace Reference	93
5.19.1	Detailed Description	94
5.19.2	Function Documentation	94
5.19.2.1	drs_radial_init	94
5.19.2.2	radial_derivative_r2r	94
5.19.2.3	radial_dr_ddr_1D_n2r	94
5.19.2.4	radial_dr_ddr_1D_r2r	95
5.19.2.5	radial_dr_ddr_3D_n2r	95
5.19.2.6	radial_dr_ddr_3D_r2r	96
5.19.3	Variable Documentation	96
5.19.3.1	b	96
5.19.3.2	drcoll	97
5.19.3.3	poly	97
5.19.3.4	poly_ddr	97
5.19.3.5	poly_dr	97
5.19.3.6	rcoll	97
5.19.3.7	rcoll2	97
5.20	drs_renderers Namespace Reference	98
5.20.1	Function Documentation	99
5.20.1.1	drs_renderers_allocation	99
5.20.1.2	render	99
5.20.1.3	render_B	100

5.20.1.4	render_B_outside	101
5.20.1.5	render_Bp	101
5.20.1.6	render_Br	101
5.20.1.7	render_Bt	101
5.20.1.8	render_Bz	102
5.20.1.9	render_helicity	102
5.20.1.10	render_poloidal_streamlines	102
5.20.1.11	render_radial_streamfunction	103
5.20.1.12	render_rotu	103
5.20.1.13	render_rotu_p	103
5.20.1.14	render_rotu_r	104
5.20.1.15	render_rotu_t	104
5.20.1.16	render_rotu_z	104
5.20.1.17	render_streamlines_t	104
5.20.1.18	render_temperature	105
5.20.1.19	render_temperature_perturbation	105
5.20.1.20	render_temprature_grad_r	105
5.20.1.21	render_u	105
5.20.1.22	render_up	106
5.20.1.23	render_ur	106
5.20.1.24	render_ut	106
5.20.1.25	render_uz	107
5.20.2	Variable Documentation	107
5.20.2.1	render_out	107
5.20.2.2	XX	107
5.20.2.3	YY	107
5.20.2.4	ZZ	107
5.21	drs_temp Namespace Reference	108
5.21.1	Detailed Description	108
5.21.2	Function Documentation	109
5.21.2.1	apply_temp_BC_RHS	109
5.21.2.2	calc_temp	109
5.21.2.3	drs_temp_allocation	109
5.21.2.4	drs_temp_init	109
5.21.2.5	drs_temp_randomize	110
5.21.2.6	drs_temp_reset	110

5.21.2.7	tempProfName	110
5.21.2.8	update_temp_lap	110
5.21.3	Variable Documentation	110
5.21.3.1	temp	110
5.21.3.2	temp_avg	111
5.21.3.3	temp_ddr	111
5.21.3.4	temp_dr	111
5.21.3.5	temp_dr_avg	111
5.21.3.6	temp_lap	111
5.21.3.7	temp_profile	111
5.21.3.8	temp_profile_dr	111
5.22	drs_time Namespace Reference	112
5.22.1	Detailed Description	113
5.22.2	Function Documentation	113
5.22.2.1	drs_time_init	113
5.22.2.2	drs_time_update	113
5.22.2.3	update_time_last_sample	114
5.22.2.4	update_timestep	114
5.22.3	Variable Documentation	114
5.22.3.1	cpu_max_time	114
5.22.3.2	cpu_time_first_step	114
5.22.3.3	cpu_time_now	114
5.22.3.4	cpu_time_start	114
5.22.3.5	drift	114
5.22.3.6	dtimestep	115
5.22.3.7	h	115
5.22.3.8	h_old	115
5.22.3.9	imeasure	115
5.22.3.10	max_time	115
5.22.3.11	nsample	115
5.22.3.12	sampling_rate	115
5.22.3.13	stepmax	115
5.22.3.14	steps	115
5.22.3.15	stepstart	116
5.22.3.16	time	116
5.22.3.17	time_last_sample	116

5.22.3.18	time_since_last_sample	116
5.22.3.19	time_start	116
5.22.3.20	transient	116
5.22.3.21	variable_h	116
5.23	drs_transforms Namespace Reference	118
5.23.1	Detailed Description	118
5.23.2	Function Documentation	119
5.23.2.1	m2phi_2D	119
5.23.2.2	my_div	119
5.23.2.3	my_rot	119
5.23.2.4	my_rotrot	120
5.23.2.5	PolTor_common2PolTor_field	120
5.23.2.6	PolTor_common2PolTor_flow	120
5.23.2.7	vectorField2Divergence	120
5.23.2.8	vectorField2PolTor_common	121
5.23.2.9	ylmb	121
5.23.2.10	ylmt	122
5.23.2.11	ylmt_3D	122
5.23.3	Variable Documentation	122
5.23.3.1	pi	122
5.24	parser Namespace Reference	123
5.24.1	Detailed Description	123
5.24.2	Function Documentation	123
5.24.2.1	parse	123
6	Class Documentation	125
6.1	drs_mpi::drs_bcast Interface Reference	125
6.1.1	Detailed Description	125
6.1.2	Member Function Documentation	125
6.1.2.1	drs_bcast_dble	125
6.1.2.2	drs_bcast_dble_scal	125
6.1.2.3	drs_bcast_int	125
6.1.2.4	drs_bcast_int_scal	125
6.1.2.5	drs_bcast_logical_scal	125
6.2	drs_mpi::drs_maximize Interface Reference	126
6.2.1	Detailed Description	126
6.2.2	Member Function Documentation	126

6.2.2.1	drs_maximize_dble	126
6.2.2.2	drs_maximize_dble_scal	126
6.3	drs_mpi::drs_minimize Interface Reference	127
6.3.1	Detailed Description	127
6.3.2	Member Function Documentation	127
6.3.2.1	drs_minimize_dble	127
6.3.2.2	drs_minimize_dble_scal	127
6.4	drs_legendre::interface Interface Reference	128
6.4.1	Member Function Documentation	128
6.4.1.1	PLegendreA_d1	128
6.4.1.2	PlmBar_d1	128
6.4.1.3	PlmIndex	128
6.5	drs_io_DX::save2DX Interface Reference	129
6.5.1	Member Function Documentation	129
6.5.1.1	save2DXscalar	129
6.5.1.2	save2DXvector	129
6.6	drs_mpi::sum_over_all_cpus Interface Reference	130
6.6.1	Detailed Description	130
6.6.2	Member Function Documentation	130
6.6.2.1	sum_over_all_cpus_scal	130
6.6.2.2	sum_over_all_cpus_vect	130
7	File Documentation	131
7.1	CrankNicholson.f90 File Reference	131
7.2	drs.f90 File Reference	133
7.2.1	Function Documentation	133
7.2.1.1	applyGreen	133
7.2.1.2	drs	134
7.2.1.3	drs_init	136
7.2.1.4	kd_grothrate	137
7.2.1.5	mk_green	138
7.2.1.6	need_to_step	138
7.2.1.7	update_field	138
7.2.1.8	update_flow	139
7.2.1.9	update_Green_functions	140
7.2.1.10	update_temp	140
7.3	drs_Chebyshev.f90 File Reference	142

7.4	drs_comp.f90 File Reference	143
7.5	drs_debug.f90 File Reference	145
7.6	drs_dims.f90 File Reference	146
7.7	drs_fftw3.f90 File Reference	148
7.8	drs_field.f90 File Reference	149
7.9	drs_flow.f90 File Reference	151
7.10	drs_hypDiff.f90 File Reference	153
7.11	drs_io.f90 File Reference	154
7.12	drs_io_par.f90 File Reference	155
7.13	drs_io_units.f90 File Reference	157
7.14	drs_legendre.f90 File Reference	158
7.15	drs_lock.f90 File Reference	160
7.16	drs_mpi.f90 File Reference	161
7.17	drs_nonlinear.f90 File Reference	164
7.18	drs_params.f90 File Reference	166
7.19	drs_probes.f90 File Reference	167
7.20	drs_radial.f90 File Reference	170
7.21	drs_temp.f90 File Reference	172
7.22	drs_time.f90 File Reference	174
7.23	drs_transforms.f90 File Reference	176
7.24	parser/parser.f90 File Reference	178
7.25	SHTOOLS/PLegendreA_d1.f90 File Reference	179
7.25.1	Function Documentation	179
7.25.1.1	PLegendreA_d1	179
7.26	SHTOOLS/PlmBar_d1.f90 File Reference	180
7.26.1	Function Documentation	180
7.26.1.1	PlmBar_d1	180
7.27	SHTOOLS/PlmIndex.f90 File Reference	181
7.27.1	Function Documentation	181
7.27.1.1	PlmIndex	181
7.28	tests/test_drs_Chebyshev.f90 File Reference	182
7.28.1	Function Documentation	182
7.28.1.1	test_drs_radial	182
7.29	tests/test_drs_fftw-r2r.f90 File Reference	183
7.29.1	Function Documentation	183
7.29.1.1	test_drs_fftw	183

7.30	tests/test_drs_fftw.f90 File Reference	184
7.30.1	Function Documentation	184
7.30.1.1	test_drs_fftw	184
7.31	tests/test_drs_radial.f90 File Reference	185
7.31.1	Function Documentation	185
7.31.1.1	test_drs_radial	185
7.31.1.2	test_radial_colocation_points	185
7.31.1.3	test_radial_derivative_r2r	185
7.31.1.4	test_radial_dr_ddr_1D_r2r	186
7.32	tests/test_logFeature.f90 File Reference	187
7.32.1	Function Documentation	187
7.32.1.1	testLogFeature	187
7.33	tests/test_remesh-r2r.f90 File Reference	188
7.33.1	Function Documentation	188
7.33.1.1	test_drs_fftw	188
7.34	tests/test_saveDXmeridional.f90 File Reference	189
7.34.1	Function Documentation	189
7.34.1.1	saveDXmeridional	189
7.34.1.2	test_saveDXMer	189
7.35	tests/test_vectorField2Divergence.f90 File Reference	190
7.35.1	Function Documentation	190
7.35.1.1	init	190
7.35.1.2	test_vectorField2Divergence	190
7.36	utilities/Benchmark-gen.f90 File Reference	192
7.36.1	Function Documentation	192
7.36.1.1	Benchmarkv1	192
7.36.1.2	init	192
7.37	utilities/Benchmark-v2.f90 File Reference	194
7.37.1	Function Documentation	194
7.37.1.1	Benchmarkv2	194
7.37.1.2	cacheTemperatureProfile	195
7.37.1.3	init	195
7.37.1.4	selectEquatorMidShell	196
7.37.1.5	volume	196
7.38	utilities/Benchmark.f90 File Reference	197
7.38.1	Function Documentation	197

7.38.1.1	Benchmarkv1	197
7.38.1.2	cacheTemperatureProfile	198
7.38.1.3	init	198
7.38.1.4	selectEquatorMidShell	199
7.38.1.5	volume	199
7.39	utilities/drs-spectra.f90 File Reference	200
7.39.1	Function Documentation	200
7.39.1.1	init	200
7.39.1.2	StateAverage	201
7.40	utilities/drs-version.f90 File Reference	203
7.40.1	Function Documentation	203
7.40.1.1	drsVersion	203
7.41	utilities/drs2dx.f90 File Reference	204
7.41.1	Function Documentation	204
7.41.1.1	drs2dx	204
7.41.1.2	init	205
7.41.1.3	parse_drs2dx	206
7.41.1.4	redefine_radial_coordinate	206
7.42	utilities/drs_io_DX.f90 File Reference	207
7.43	utilities/drs_renderers.f90 File Reference	209
7.43.1	Define Documentation	210
7.43.1.1	ERR_UNSUPPORTED_OPTION	210
7.44	utilities/getProfile.f90 File Reference	211
7.44.1	Function Documentation	211
7.44.1.1	getProfile	211
7.44.1.2	init	212
7.44.1.3	parseConfig	213
7.44.1.4	setWhatName	213
7.45	utilities/state-average.f90 File Reference	214
7.45.1	Function Documentation	214
7.45.1.1	init	214
7.45.1.2	StateAverage	215
7.46	utilities/Yokoi-plots-rs.f90 File Reference	216
7.46.1	Function Documentation	216
7.46.1.1	computeAlpha	216
7.46.1.2	computeAndSaveAverage	216

7.46.1.3	computeBeta	217
7.46.1.4	computeEMF	217
7.46.1.5	computeGamma	217
7.46.1.6	init	217
7.46.1.7	saveIDLmeridional	218
7.46.1.8	StateAverage	218
7.47	utilities/Yokoi-plots.f90 File Reference	220
7.47.1	Function Documentation	220
7.47.1.1	computeAlpha	220
7.47.1.2	computeAndSaveAverage	220
7.47.1.3	computeBeta	221
7.47.1.4	computeEMF	221
7.47.1.5	computeGamma	221
7.47.1.6	init	221
7.47.1.7	saveDXmeridional	222
7.47.1.8	YokoiPlots	222

Chapter 1

Todo List

Member `drs_io::save_state` (p. 46)() Describe the format of the following files.

Member `drs_io_par::read_input_par` (p. 53)(`unit_in`) Restore read states with other magic numbers.

Member `drs_probes::nusselt` (p. 87)(`r`) It only works for serial runs. Needs to be parallelized.

Member `drs_probes::save_angular_momentum` (p. 88)(`u_t`, `u_p`) Split the saving part and move it to io.

Member `drs_probes::save_field_coeffs` (p. 88)() Should take a list of l's and m's and reply with a list of values
Writing should be moved to io.

Member `drs_probes::save_flow_coeffs` (p. 88)() Should take a list of l's and m's and reply with a list of values
Writing should be moved to io.

Member `drs_probes::save_flow_dissipation` (p. 89)(`mmax`) Separate computing from writing.

Member `drs_temp::apply_temp_BC_RHS` (p. 109)(`val`) make this value depend on l and m when we can specify the full 2D anomaly at the boundaries.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

CrankNicholson (Provides routines that compute the Crank-Nicholson inverse operators and variables to store them)	9
drs_Chebyshev (Module containing the implementation of the Chebyshev polynomials)	12
drs_comp (Module dealing with the composition)	17
drs_debug (Module with helper subroutines for debug)	21
drs_dims (Provides variables to store the real space and spectral space dimensions of the problem)	22
drs_fftw3 (This module abstracts the computation of Fourier and cosinus transforms)	26
drs_field	30
drs_flow	36
drs_hypDiff	42
drs_io (Deals with input and output of state files and derived quantities)	43
drs_io_DX	48
drs_io_par (Module to read and write parameter and configuration files)	52
drs_io_units (Manages the I/O units of DRS)	58
drs_legendre	61
drs_lock (This module provides a locking mechanism for the dynamo code)	65
drs_mpi (Provides initialisation and variables to be used with the mpi implementation)	67
drs_nonlinear (Takes care of constructing the nonlinear terms of all equations and other quantities in real space)	75
drs_probes (This module implements some prbing facilities for the running models)	81
drs_radial (This module implements the radial domain and operations in it)	93
drs_renderers	98
drs_temp (Temperature related operations)	108
drs_time (Module to deal with time. It deals with both wall time and simulation time. It also deals wit the time-stepping)	112
drs_transforms (Spherical transforms)	118
parser (This module provides a simple parser (p. 123) for input files of the type 'key = val' eventually separated by '[Sections]')	123

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

drs_mpi::drs_bcast (Encapsulates broadcast of several types and ranks)	125
drs_mpi::drs_maximize (Encapsulates maximization of several types and ranks)	126
drs_mpi::drs_minimize (Encapsulates minimization of several types and ranks)	127
drs_legendre::interface	128
drs_io_DX::save2DX	129
drs_mpi::sum_over_all_cpus (Encapsulates sums of several types and ranks)	130

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

CrankNicholson.f90	131
drs.f90	133
drs_Chebyshev.f90	142
drs_comp.f90	143
drs_debug.f90	145
drs_dims.f90	146
drs_fftw3.f90	148
drs_field.f90	149
drs_flow.f90	151
drs_hypDiff.f90	153
drs_io.f90	154
drs_io_par.f90	155
drs_io_units.f90	157
drs_legendre.f90	158
drs_lock.f90	160
drs_mpi.f90	161
drs_nonlinear.f90	164
drs_params.f90	166
drs_probes.f90	167
drs_radial.f90	170
drs_temp.f90	172
drs_time.f90	174
drs_transforms.f90	176
parser/ parser.f90	178
SHTOOLS/ PLegendreA_d1.f90	179
SHTOOLS/ PlmBar_d1.f90	180
SHTOOLS/ PlmIndex.f90	181
tests/ test_drs_Chebyshev.f90	182
tests/ test_drs_fftw-r2r.f90	183
tests/ test_drs_fftw.f90	184
tests/ test_drs_radial.f90	185
tests/ test_logFeature.f90	187
tests/ test_remesh-r2r.f90	188

tests/ test_saveDXmeridional.f90	189
tests/ test_vectorField2Divergence.f90	190
utilities/ Benchmark-gen.f90	192
utilities/ Benchmark-v2.f90	194
utilities/ Benchmark.f90	197
utilities/ drs-spectra.f90	200
utilities/ drs-version.f90	203
utilities/ drs2dx.f90	204
utilities/ drs_io_DX.f90	207
utilities/ drs_renderers.f90	209
utilities/ getProfile.f90	211
utilities/ state-average.f90	214
utilities/ Yokoi-plots-rs.f90	216
utilities/ Yokoi-plots.f90	220

Chapter 5

Namespace Documentation

5.1 CrankNicholson Namespace Reference

Provides routines that compute the Crank-Nicholson inverse operators and variables to store them.

Functions

- subroutine **CrankNicholson_init** ()
Allocates memory for the Crank-Nicholson inverse operators.
- subroutine **updateCrankNicholson_matrices** (h, Pt, Pm, Pc)
Convenience subroutine that generates all of the Crank-Nicholson inverse operators.

Variables

- double precision, dimension(:, :, :), allocatable **field_lap_inv_tor**
The Crank-Nicholson inverse operator for the toroidal flow.
- double precision, dimension(:, :, :), allocatable **field_lap_inv_pol**
The Crank-Nicholson inverse operator for the poloidal flow.
- double precision, dimension(:, :, :), allocatable **flow_lap_inv_tor**
The Crank-Nicholson inverse operator for the toroidal field.
- double precision, dimension(:, :, :), allocatable **flow_lap_inv_pol**
The Crank-Nicholson inverse operator for the poloidal field.
- double precision, dimension(:, :, :), allocatable **temp_lap_inv**
The Crank-Nicholson inverse operator for the temperature.
- double precision, dimension(:, :, :), allocatable, target **pinv**
The inverse laplacian operator in real space.

5.1.1 Detailed Description

Provides routines that compute the Crank-Nicholson inverse operators and variables to store them.

5.1.2 Function Documentation

5.1.2.1 subroutine CrankNicholson::CrankNicholson_init ()

Allocates memory for the Crank-Nicholson inverse operators.

References field_lap_inv_pol, field_lap_inv_tor, flow_lap_inv_pol, flow_lap_inv_tor, drs_dims::Nr, drs_dims::Nt, drs_dims::Nt_s, pinv, and temp_lap_inv.

Referenced by drs_init().

5.1.2.2 subroutine CrankNicholson::updateCrankNicholson_matrices (double precision,intent(in) *h*, double precision,intent(in) *Pt*, double precision,intent(in) *Pm*, double precision,intent(in),optional *Pc*)

Convenience subroutine that generates all of the Crank-Nicholson inverse operators.

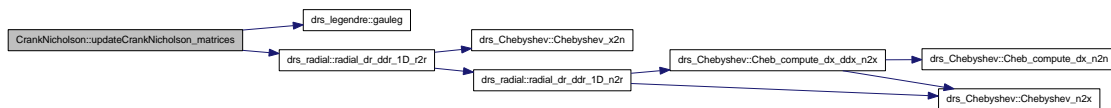
Parameters:

- h* the current time step size.
- Pt* the thermal Prandtl number.
- Pm* the magnetic Prandtl number.
- Pc* the compositional Prandtl number.

References field_lap_inv_pol, field_lap_inv_tor, flow_lap_inv_pol, flow_lap_inv_tor, drs_legendre::gauleg(), drs_legendre::llp1, drs_dims::Nr_s, drs_radial::poly, drs_radial::poly_ddr, drs_radial::poly_dr, drs_radial::radial_dr_ddr_1D_r2r(), drs_radial::rcoll, drs_radial::rcoll2, and temp_lap_inv.

Referenced by drs().

Here is the call graph for this function:



5.1.3 Variable Documentation

5.1.3.1 double precision,dimension(:, :, :),allocatable CrankNicholson::field_lap_inv_pol

The Crank-Nicholson inverse operator for the poloidal flow.

Referenced by CrankNicholson_init(), update_field(), and updateCrankNicholson_matrices().

5.1.3.2 double precision,dimension(:, :, :),allocatable CrankNicholson::field_lap_inv_tor

The Crank-Nicholson inverse operator for the toroidal flow.

Referenced by CrankNicholson_init(), update_field(), and updateCrankNicholson_matrices().

5.1.3.3 double precision,dimension(:, :, :),allocatable CrankNicholson::flow_lap_inv_pol

The Crank-Nicholson inverse operator for the poloidal field.

Referenced by CrankNicholson_init(), mk_green(), update_flow(), and updateCrankNicholson_matrices().

5.1.3.4 double precision,dimension(:, :, :),allocatable CrankNicholson::flow_lap_inv_tor

The Crank-Nicholson inverse operator for the toroidal field.

Referenced by CrankNicholson_init(), update_flow(), and updateCrankNicholson_matrices().

5.1.3.5 double precision,dimension(:, :, :),allocatable,target CrankNicholson::pinv

The inverse laplacian operator in real space.

Referenced by CrankNicholson_init(), mk_green(), and update_flow().

5.1.3.6 double precision,dimension(:, :, :),allocatable CrankNicholson::temp_lap_inv

The Crank-Nicholson inverse operator for the temperature.

Referenced by CrankNicholson_init(), update_temp(), and updateCrankNicholson_matrices().

5.2 drs_Chebyshev Namespace Reference

Module containing the implementation of the Chebyshev polynomials.

Functions

- subroutine **Chebyshev_init** (N, N_s)
Computes the Chebyshev polynomials of order up to N as a function of r.
- subroutine **Chebyshev_cleanup** ()
- subroutine **Cheb_compute_dx_ddx_n2x** (f, dfdx, d2fdx2)
Returns second radial derivative in d2fdx2, first derivative in dfdx Input f is supposed to be given in Chebyshev space, derivatives are returned in direct space.
- subroutine **Cheb_compute_dx_ddx_x2x** (f, dfdx, d2fdx2)
Returns second radial derivative in d2fdx2, first derivative in dfdx Input f is supposed to be given in real space, derivatives are returned in real space.
- subroutine **Cheb_compute_dx_n2n** (f, dfdx)
Computes the Chebyshev coefficients of the first derivative of f with respect to x.
- subroutine **Chebyshev_x2n** (input)
The forward real to spectral cosinus transform Since $T_n(\cos(t)) = \cos(nt)$, the forward cosinus transform gives us the coefficients of order n of the expansion of a scalar function $f(x)$ in terms of Chebyshev polynomials.
- subroutine **Chebyshev_n2x** (input)
The backward spectral to real cosinus transform Since $T_n(\cos(t)) = \cos(nt)$, the backward cosinus transform gives us the value of a scalar function $f(x)$ in terms of Chebyshev polynomials.

Variables

- double precision, parameter **pi** = 3.141592653589793d0
It makes use of fftw3.
- integer *8 **plan_x**
- integer **Nx**
- integer **Nx_s**
- double precision, dimension(:), allocatable **ct_buffer**
- double precision, allocatable **Cheb_x**
- double precision, allocatable, target **Chebyshev**
- double precision, allocatable, target **Chebyshev_dx**
- double precision, allocatable, target **Chebyshev_ddx**
First index is radial point, second index is mode index.

5.2.1 Detailed Description

Module containing the implementation of the Chebyshev polynomials.

5.2.2 Function Documentation

5.2.2.1 subroutine `drs_Chebyshev::Cheb_compute_dx_ddx_n2x` (double precision,dimension(nx),intent(in) *f*, double precision,dimension(nx),intent(out) *dfdx*, double precision,dimension(nx),intent(out) *d2fdx2*)

Returns second radial derivative in *d2fdx2*, first derivative in *dfdx* Input *f* is supposed to be given in Chebyshev space, derivatives are returned in direct space.

Since:

1.6.5

Parameters:

f Chebyshev coefficients of the input function

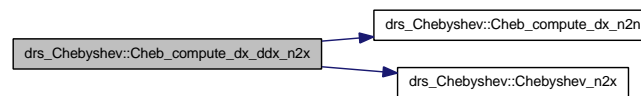
dfdx First derivative of *f* at points 1..Nx

d2fdx2 Second derivative of *f* at points 1..Nx

References `Cheb_compute_dx_n2n()`, `Chebyshev_n2x()`, and `Nx_s`.

Referenced by `Chebyshev_init()`, `drs_radial::radial_dr_ddr_1D_n2r()`, and `drs_radial::radial_dr_ddr_3D_n2r()`.

Here is the call graph for this function:



5.2.2.2 subroutine `drs_Chebyshev::Cheb_compute_dx_ddx_x2x` (double precision,dimension(nx),intent(in) *f*, double precision,dimension(nx),intent(out) *dfdx*, double precision,dimension(nx),intent(out) *d2fdx2*)

Returns second radial derivative in *d2fdx2*, first derivative in *dfdx* Input *f* is supposed to be given in real space, derivatives are returned in real space.

Since:

1.6.5

Parameters:

f Chebyshev coefficients of the input function

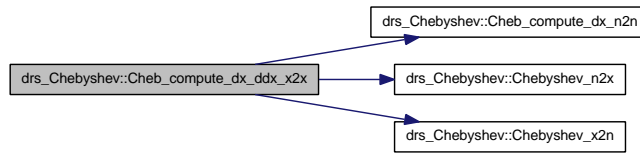
dfdx First derivative of *f* at points 1..Nx

d2fdx2 Second derivative of *f* at points 1..Nx

References `Cheb_compute_dx_n2n()`, `Chebyshev_n2x()`, `Chebyshev_x2n()`, and `Nx_s`.

Referenced by `drs_radial::radial_dr_ddr_3D_r2r()`.

Here is the call graph for this function:



5.2.2.3 subroutine `drs_Chebyshev::Cheb_compute_dx_n2n` (double precision,dimension(nx),intent(in) *f*, double precision,dimension(nx),intent(out) *dfdx*)

Computes the Chebyshev coefficients of the first derivative of *f* with respect to *x*.

Since:

1.6.5

Parameters:

f Chebyshev coefficients of the input function
dfdx Chebyshev coefficients of the derivative.

Referenced by `Cheb_compute_dx_ddx_n2x()`, `Cheb_compute_dx_ddx_x2x()`, and `drs_radial::drs_radial_init()`.

5.2.2.4 subroutine `drs_Chebyshev::Chebyshev_cleanup` ()

References `Cheb_x`, `Chebyshev`, `Chebyshev_ddx`, `Chebyshev_dx`, `ct_buffer`, and `plan_x`.

5.2.2.5 subroutine `drs_Chebyshev::Chebyshev_init` (integer,intent(in) *N*, integer,intent(in) *N_s*)

Computes the Chebyshev polynomials of order up to *N* as a function of *r*.

Since:

1.6.5

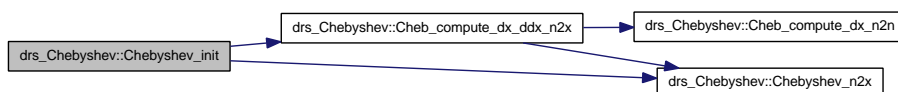
Parameters:

N Number of points the polynomials
N_s maximum order of the polynomials

References `Cheb_compute_dx_ddx_n2x()`, `Cheb_x`, `Chebyshev`, `Chebyshev_ddx`, `Chebyshev_dx`, `Chebyshev_n2x()`, `ct_buffer`, `Nx`, `Nx_s`, `pi`, and `plan_x`.

Referenced by `drs_radial::drs_radial_init()`, and `test_drs_radial()`.

Here is the call graph for this function:



5.2.2.6 subroutine drs_Chebyshev::Chebyshev_n2x (double precision,dimension(nx),intent(inout) *input*)

The backward spectral to real cosinus transform Since $T_n(\cos(t)) = \cos(nt)$, the backward cosinus transform gives us the value of a scalar function $f(x)$ in terms of Chebyshev polynomials.

Since:

1.6.5

References Nx_s, and plan_x.

Referenced by Cheb_compute_dx_ddx_n2x(), Cheb_compute_dx_ddx_x2x(), Chebyshev_init(), drs_radial::radial_derivative_r2r(), drs_radial::radial_dr_ddr_1D_n2r(), drs_radial::radial_dr_ddr_3D_n2r(), and test_drs_radial().

5.2.2.7 subroutine drs_Chebyshev::Chebyshev_x2n (double precision,dimension(nx),intent(inout) *input*)

The forward real to spectral cosinus transform Since $T_n(\cos(t)) = \cos(nt)$, the forward cosinus transform gives us the coefficients of order n of the expansion of a scalar function $f(x)$ in terms of Chebyshev polynomials.

Since:

1.6.5

Parameters:

input

References Nx_s, and plan_x.

Referenced by Cheb_compute_dx_ddx_x2x(), drs_radial::radial_derivative_r2r(), drs_radial::radial_dr_ddr_1D_r2r(), and test_drs_radial().

5.2.3 Variable Documentation

5.2.3.1 double precision,allocatable drs_Chebyshev::Cheb_x

Referenced by Chebyshev_cleanup(), Chebyshev_init(), and drs_radial::drs_radial_init().

5.2.3.2 double precision,allocatable,target drs_Chebyshev::Chebyshev

Referenced by Chebyshev_cleanup(), Chebyshev_init(), drs_radial::drs_radial_init(), and test_drs_radial().

5.2.3.3 double precision,allocatable,target drs_Chebyshev::Chebyshev_ddx

First index is radial point, second index is mode index.

Referenced by Chebyshev_cleanup(), Chebyshev_init(), and drs_radial::drs_radial_init().

5.2.3.4 double precision,allocatable,target drs_Chebyshev::Chebyshev_dx

Referenced by Chebyshev_cleanup(), Chebyshev_init(), drs_radial::drs_radial_init(), and test_drs_radial().

5.2.3.5 double precision,dimension(:),allocatable drs_Chebyshev::ct_buffer

Referenced by Chebyshev_cleanup(), and Chebyshev_init().

5.2.3.6 integer drs_Chebyshev::Nx

Referenced by Chebyshev_init().

5.2.3.7 integer drs_Chebyshev::Nx_s

Referenced by Cheb_compute_dx_ddx_n2x(), Cheb_compute_dx_ddx_x2x(), Chebyshev_init(), Chebyshev_n2x(), and Chebyshev_x2n().

5.2.3.8 double precision,parameter drs_Chebyshev::pi = 3.141592653589793d0

It makes use of fftw3.

Referenced by Chebyshev_init().

5.2.3.9 integer*8 drs_Chebyshev::plan_x

Referenced by Chebyshev_cleanup(), Chebyshev_init(), Chebyshev_n2x(), and Chebyshev_x2n().

5.3 drs_comp Namespace Reference

Module dealing with the composition.

Functions

- subroutine **drs_comp_allocation** ()
Allocates the variables required for computations involving composition.
- subroutine **drs_comp_init** ()
Initialises the composition boundary conditions, derivatives and profiles.
- character(len=16) **compProfName** ()
Outputs a human readable name for the composition profiles.
- subroutine **drs_comp_reset** ()
Resets the composition and its derivatives to 0.
- subroutine **drs_comp_randomize** (noise)
Computes the laplacian of the composition.
- subroutine **apply_comp_BC** (comp)
These lines take care of boundary conditions If the value at a boundary is bc different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.
- subroutine **calc_comp** (comp_spec, comp_real)
Computes the composition in real space from the composition in spectral space.

Variables

- double precision, dimension(:, :, :), allocatable **comp**
- double precision, dimension(:, :, :), allocatable **comp_dr**
- double precision, dimension(:, :, :), allocatable **comp_ddr**
- double precision, dimension(:, :, :), allocatable **comp_avg**
- double precision, dimension(:), allocatable **comp_dr_avg**
- double precision, dimension(:), allocatable **comp_profile**
- double precision, dimension(:), allocatable **comp_profile_dr**

5.3.1 Detailed Description

Module dealing with the composition.

5.3.2 Function Documentation

5.3.2.1 subroutine drs_comp::apply_comp_BC (double precision,dimension(nr),intent(inout) *comp*)

These lines take care of boundary conditions. If the value at a boundary is *bc* different from 0, insert *bc/2* because of the factor 2 between Chebychev and radial differentiation. If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value *bc* is different from 0.

Parameters:

comp A pencil with the forces in *lmr* space.

Referenced by `drs()`, and `update_temp()`.

5.3.2.2 subroutine drs_comp::calc_comp (double precision,dimension(0:nt_s,1:blk_ps_size(mpi_rank),intent(in) *comp_spec*, double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *comp_real*)

Computes the composition in real space from the composition in spectral space.

Parameters:

comp_spec Composition in spectral space.

comp_real Composition in real space.

Referenced by `drs_nonlinear::evaluate_real_space()`, and `drs_renderers::render_temperature()`.

5.3.2.3 character(len=16) drs_comp::compProfName ()

Outputs a human readable name for the composition profiles.

Since:

1.6.1

Referenced by `drs_comp_init()`.

5.3.2.4 subroutine drs_comp::drs_comp_allocation ()

Allocates the variables required for computations involving composition.

References `drs_mpi::blk_ps_size`, `comp`, `comp_avg`, `comp_ddr`, `comp_dr`, `comp_dr_avg`, `comp_profile`, `comp_profile_dr`, `drs_mpi::mpi_rank`, `drs_dims::Nr`, and `drs_dims::Nt_s`.

Referenced by `drs_init()`, and `init()`.

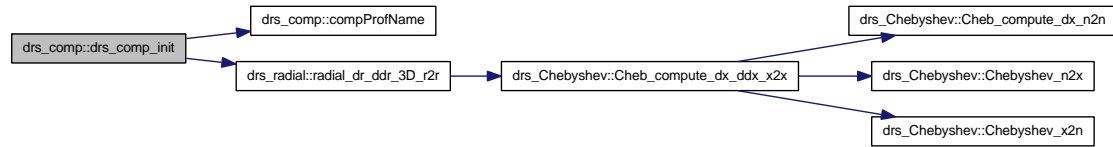
5.3.2.5 subroutine drs_comp::drs_comp_init ()

Initialises the composition boundary conditions, derivatives and profiles.

References `comp`, `comp_ddr`, `comp_dr`, `comp_profile`, `comp_profile_dr`, `compProfName()`, `drs_dims::Nr`, `drs_radial::radial_dr_ddr_3D_r2r()`, `drs_radial::rcoll`, and `drs_radial::rcoll2`.

Referenced by `drs_init()`, `getProfile()`, and `init()`.

Here is the call graph for this function:



5.3.2.6 subroutine `drs_comp::drs_comp_randomize` (double precision,intent(in) *noise*)

Computes the laplacian of the composition.

References `comp`.

5.3.2.7 subroutine `drs_comp::drs_comp_reset` ()

Resets the composition and its derivatives to 0.

References `comp`, `comp_ddr`, `comp_dr`, `drs_legendre::llp1`, `drs_radial::rcoll`, and `drs_radial::rcoll2`.

5.3.3 Variable Documentation

5.3.3.1 double precision,dimension(:, :, :),allocatable `drs_comp::comp`

Referenced by `drs()`, `drs_comp_allocation()`, `drs_comp_init()`, `drs_comp_randomize()`, `drs_comp_reset()`, `drs_io::drs_load_state()`, `drs_nonlinear::evaluate_real_space()`, `getProfile()`, `drs_renderers::render_temperature()`, `drs_io::save_l_spec()`, `drs_io::save_m_spec()`, `drs_io::save_n_spec()`, `drs_io::save_state()`, `drs_nonlinear::save_stuff()`, and `update_temp()`.

5.3.3.2 double precision,dimension(:, :, :),allocatable `drs_comp::comp_avg`

Referenced by `drs_comp_allocation()`.

5.3.3.3 double precision,dimension(:, :, :),allocatable `drs_comp::comp_ddr`

Referenced by `drs()`, `drs_comp_allocation()`, `drs_comp_init()`, `drs_comp_reset()`, and `update_temp()`.

5.3.3.4 double precision,dimension(:, :, :),allocatable `drs_comp::comp_dr`

Referenced by `drs()`, `drs_comp_allocation()`, `drs_comp_init()`, `drs_comp_reset()`, and `update_temp()`.

5.3.3.5 double precision,dimension(:),allocatable `drs_comp::comp_dr_avg`

Referenced by `drs_comp_allocation()`.

5.3.3.6 double precision,dimension(:),allocatable drs_comp::comp_profile

Referenced by `drs_comp_allocation()`, `drs_comp_init()`, `getProfile()`, and `drs_renderers::render_temperature()`.

5.3.3.7 double precision,dimension(:),allocatable drs_comp::comp_profile_dr

Referenced by `drs_comp_allocation()`, `drs_comp_init()`, and `drs_nonlinear::save_stuff()`.

5.4 drs_debug Namespace Reference

Module with helper subroutines for debug.

Functions

- subroutine **save_lmr_quantity** (*t*, *tname*)
- subroutine **save_tpr_quantity** (*t*, *tname*)

5.4.1 Detailed Description

Module with helper subroutines for debug.

5.4.2 Function Documentation

5.4.2.1 subroutine drs_debug::save_lmr_quantity (double precision,dimension(0:nt_s,blk_ps-size(mpi_rank),intent(in) *t*, character(*),intent(in) *tname*)

Parameters:

- t* Fragment of the array to be saved held by CPU with mpi_rank.
tname The name of the file that will be saved to disk.

References drs_mpi::blk_ps_start, drs_mpi::mpi_size, and drs_mpi::wait_for_everyone().

Here is the call graph for this function:



5.4.2.2 subroutine drs_debug::save_tpr_quantity (double precision,dimension(0:blk_t_size(mpi_rank),intent(in) *t*, character(*),intent(in) *tname*)

References drs_mpi::blk_t_start, drs_mpi::mpi_size, and drs_mpi::wait_for_everyone().

Here is the call graph for this function:



5.5 drs_dims Namespace Reference

Provides variables to store the real space and spectral space dimensions of the problem.

Functions

- subroutine **drs_dims_init** (error)
- subroutine **check_dims** (error)
Checks consistency of input parameters.

Variables

- integer, dimension(8), target **usr_dims**
- integer **Nr**
Number of radial points.
- integer **Nt**
Number of meridional points.
- integer **Np**
Number of azimuthal points.
- integer **Nr_s**
Highest index for the polynomials in the radial direction.
- integer **Nt_s**
Number of spherical harmonic degrees to use, including 0.
- integer **Np_s**
Number of spherical harmonic orders (positive, negative and zero) to use.
- integer **lsymm**
Equatorial symmetry.
- integer **m0**
Axial symmetry to use.

5.5.1 Detailed Description

Provides variables to store the real space and spectral space dimensions of the problem.

5.5.2 Function Documentation

5.5.2.1 subroutine drs_dims::check_dims (integer,intent(out) error)

Checks consistency of input parameters.

References m0, Np, Np_s, Nr, Nr_s, Nt, and Nt_s.

Referenced by drs_init(), and init().

5.5.2.2 subroutine drs_dims::drs_dims_init (integer,intent(out) error)

References lsymm, m0, Np, Np_s, Nr, Nr_s, Nt, Nt_s, and usr_dims.

Referenced by drs_init(), init(), test_drs_radial(), and test_saveDXMer().

5.5.3 Variable Documentation

5.5.3.1 integer drs_dims::lsymm

Equatorial symmetry.

Referenced by drs_dims_init(), drs_init(), drs_io_par::drs_read_conf(), drs_io_par::drs_read_conf_v2(), init(), and drs_io_par::write_parp().

5.5.3.2 integer drs_dims::m0

Axial symmetry to use.

Referenced by drs_probes::average_unnormalised_field_l_spectrum(), drs_probes::average_unnormalised_flow_l_spectrum(), drs_probes::average_unnormalised_scalar_l_spectrum(), drs_field::calc_field_lspec(), drs_field::calc_field_nspec(), drs_flow::calc_flow_lspec(), drs_flow::calc_flow_nspec(), check_dims(), drs_probes::check_resolution_Hartman(), drs_dims_init(), drs_field::drs_field_random_init(), drs_init(), drs_io::drs_load_state(), drs_io::drs_open_output(), drs_io_par::drs_read_conf(), drs_io_par::drs_read_conf_v2(), drs_temp::drs_temp_randomize(), init(), kd_grothrate(), drs_probes::l_spec_of_scalar_field(), drs_probes::measure_lm(), drs_probes::n_spec_of_scalar_field(), drs_renderers::render_poloidal_streamlines(), drs_renderers::render_streamlines_t(), drs_probes::save_angular_momentum(), drs_probes::save_field_coeffs(), drs_probes::save_flow_coeffs(), drs_probes::save_flow_dissipation(), drs_probes::save_magnetic_dissipation(), drs_io_DX::saveDXmeridional(), saveDXmeridional(), drs_io_DX::saveDXmeridional3DVec(), drs_io_DX::saveDXvolume(), drs_io_DX::saveDXvolume3DVec(), drs_io_DX::saveDXvolume_v2(), saveIDLmeridional(), test_saveDXMer(), and drs_io_par::write_parp().

5.5.3.3 integer drs_dims::Np

Number of azimuthal points.

Referenced by Benchmarkv1(), Benchmarkv2(), check_dims(), drs_probes::check_resolution_Hartman(), drs_dims_init(), drs_init(), drs_nonlinear::drs_nonlinear_init(), drs_probes::drs_probes_init(), drs_io_par::drs_read_conf(), drs_io_par::drs_read_conf_v2(), drs_renderers::drs_renderers_allocation(), init(), parse_drs2dx(), drs_renderers::render_temperature(), drs_renderers::render_temprature_grad_r(), drs_nonlinear::save_stuff(), StateAverage(), test_drs_radial(), test_saveDXMer(), drs_io_par::write_parp(), and YokoiPlots().

5.5.3.4 integer drs_dims::Np_s

Number of spherical harmonic orders (positive, negative and zero) to use.

Referenced by `drs_probes::average_unnormalised_field_l_spectrum()`, `drs_probes::average_unnormalised_flow_l_spectrum()`, `drs_probes::average_unnormalised_scalar_l_spectrum()`, `Benchmarkv1()`, `Benchmarkv2()`, `drs_field::calc_field_lspec()`, `drs_field::calc_field_nspec()`, `drs_flow::calc_flow_lspec()`, `drs_flow::calc_flow_nspec()`, `check_dims()`, `computeAndSaveAverage()`, `drs_dims_init()`, `drs_field::drs_field_random_init()`, `drs_init()`, `drs_io::drs_open_output()`, `drs_io_par::drs_read_conf()`, `drs_io_par::drs_read_conf_v2()`, `drs_temp::drs_temp_randomize()`, `init()`, `kd_grothrate()`, `drs_probes::l_spec_of_scalar_field()`, `drs_transforms::m2phi_2D()`, `drs_probes::measure_lm()`, `drs_probes::n_spec_of_scalar_field()`, `drs_renderers::render_poloidal_streamlines()`, `test_saveDXMer()`, `drs_io_par::write_parp()`, `drs_transforms::ylmt()`, and `drs_transforms::ylmt_3D()`.

5.5.3.5 integer `drs_dims::Nr`

Number of radial points.

Referenced by `applyGreen()`, `drs_probes::average_unnormalised_field_l_spectrum()`, `drs_probes::average_unnormalised_flow_l_spectrum()`, `Benchmarkv1()`, `Benchmarkv2()`, `drs_field::calc_field_lspec()`, `drs_field::calc_field_mspec()`, `drs_flow::calc_flow_lspec()`, `drs_flow::calc_flow_mspec()`, `check_dims()`, `drs_probes::check_resolution_Hartman()`, `CrankNicholson::CrankNicholson_init()`, `drs()`, `drs_comp::drs_comp_allocation()`, `drs_comp::drs_comp_init()`, `drs_dims_init()`, `drs_field::drs_field_allocation()`, `drs_field::drs_field_random_init()`, `drs_flow::drs_flow_allocation()`, `drs_init()`, `drs_io::drs_load_state()`, `drs_nonlinear::drs_nonlinear_init()`, `drs_probes::drs_probes_allocation()`, `drs_probes::drs_probes_init()`, `drs_radial::drs_radial_init()`, `drs_io_par::drs_read_conf()`, `drs_io_par::drs_read_conf_v2()`, `drs_renderers::drs_renderers_allocation()`, `drs_temp::drs_temp_allocation()`, `drs_temp::drs_temp_init()`, `drs_temp::drs_temp_randomize()`, `getProfile()`, `init()`, `kd_grothrate()`, `parse_drs2dx()`, `redefine_radial_coordinate()`, `drs_renderers::render_streamlines_t()`, `drs_renderers::render_temperature()`, `drs_renderers::render_temperature_grad_r()`, `drs_probes::save_field_coeffs()`, `drs_probes::save_flow_coeffs()`, `drs_nonlinear::save_stuff()`, `StateAverage()`, `test_drs_radial()`, `test_radial_colocation_points()`, `test_saveDXMer()`, `test_vectorField2Divergence()`, `update_field()`, `drs_field::update_field_pol_lap()`, `drs_field::update_field_tor_lap()`, `drs_flow::update_flow_pol_lap()`, `drs_flow::update_flow_tor_lap()`, `update_temp()`, `drs_temp::update_temp_lap()`, `drs_io_par::write_parp()`, and `YokoiPlots()`.

5.5.3.6 integer `drs_dims::Nr_s`

Highest index for the polynomials in the radial direction.

Referenced by `check_dims()`, `drs_dims_init()`, `drs_init()`, `drs_radial::drs_radial_init()`, `drs_io_par::drs_read_conf()`, `drs_io_par::drs_read_conf_v2()`, `init()`, `test_drs_radial()`, `test_saveDXMer()`, `CrankNicholson::updateCrankNicholson_matrices()`, and `drs_io_par::write_parp()`.

5.5.3.7 integer `drs_dims::Nt`

Number of meridional points.

Referenced by `Benchmarkv1()`, `check_dims()`, `drs_probes::check_resolution_Hartman()`, `drs_probes::compute_helicities()`, `computeAlpha()`, `computeBeta()`, `computeEMF()`, `computeGamma()`, `CrankNicholson::CrankNicholson_init()`, `drs_dims_init()`, `drs_init()`, `drs_legendre::drs_legendre_allocation()`, `drs_legendre::drs_legendre_init()`, `drs_io_par::drs_read_conf()`, `drs_io_par::drs_read_conf_v2()`, `drs_renderers::drs_renderers_allocation()`, `init()`, `drs_legendre::legendre_init_new()`, `parse_drs2dx()`, `drs_renderers::render_temperature()`, `drs_renderers::render_temperature_grad_r()`, `drs_nonlinear::save_stuff()`, `drs_io_DX::saveDXmeridional()`, `saveDXmeridional()`, `drs_io_DX::saveDXmeridional3DVec()`, `drs_io_DX::saveDXvolume()`, `drs_io_DX::saveDXvolume3DVec()`, `drs_io_DX::saveDXvolume_v2()`, `saveIDLmeridional()`, `selectEquatorMidShell()`, `StateAverage()`, `test_drs_radial()`, `test_saveDXMer()`, `drs_io_par::write_parp()`, and `YokoiPlots()`.

5.5.3.8 integer drs_dims::Nt_s

Number of spherical harmonic degrees to use, including 0.

Referenced by Benchmarkv1(), Benchmarkv2(), drs_field::calc_field_mspect(), drs_flow::calc_flow_mspect(), check_dims(), CrankNicholson::CrankNicholson_init(), drs_comp::drs_comp_allocation(), drs_dims_init(), drs_field::drs_field_allocation(), drs_field::drs_field_random_init(), drs_flow::drs_flow_allocation(), drs_init(), drs_legendre::drs_legendre_allocation(), drs_legendre::drs_legendre_init(), drs_nonlinear::drs_nonlinear_init(), drs_probes::drs_probes_allocation(), drs_io_par::drs_read_conf(), drs_io_par::drs_read_conf_v2(), drs_temp::drs_temp_allocation(), drs_temp::drs_temp_randomize(), init(), kd_grothrate(), drs_probes::measure_lm(), drs_renderers::render_B_outside(), drs_renderers::render_poloidal_streamlines(), drs_renderers::render_streamlines_t(), test_saveDXMer(), and drs_io_par::write_parp().

5.5.3.9 integer,dimension(8),target drs_dims::usr_dims

Referenced by drs_dims_init(), and drs_init().

5.6 drs_fftw3 Namespace Reference

This module abstracts the computation of Fourier and cosinus transforms.

Functions

- subroutine **drs_fftw3_init** (Nr, Nt, Np)
Initialises all the fftw3 plans for forward and backward Fourier and cosinus transforms.
- subroutine **drs_fftw3_cleanup** ()
Destroys the plans.
- subroutine **dft_forward** (input, output)
The forward real to spectral DFT.
- subroutine **dft_backward** (input, output)
The backward, spectral to real DFT.
- subroutine **cos_r2r_1_r2n** (input)
The forward real to spectral cosinus transform.
- subroutine **cos_r2r_1_n2r** (input)
The backward spectral to real cosinus transform.
- subroutine **remesh** (Nr1, f1, Nr2, f2)
Given a field f1 described at Nr1 points in an interval, remesh outputs the same field, in the same interval at Nr2 points as f2.

Variables

- integer *8 **plan_r**
It makes use of fftw3.
- integer *8 **plan_pf**
- integer *8 **plan_pb**
- double precision, dimension(:, :), allocatable **in_p**
- double precision, dimension(:, :), allocatable **inout_r**
- integer **drs_fftw3_Nr**
- integer **drs_fftw3_Np**
- integer **drs_fftw3_Nt**

5.6.1 Detailed Description

This module abstracts the computation of Fourier and cosinus transforms.

5.6.2 Function Documentation

5.6.2.1 subroutine drs_fftw3::cos_r2r_1_n2r (double precision,dimension(drs_fftw3_nr),intent(inout) *input*)

The backward spectral to real cosinus transform.

References plan_r.

Referenced by test_drs_fftw().

5.6.2.2 subroutine drs_fftw3::cos_r2r_1_r2n (double precision,dimension(drs_fftw3_nr),intent(inout) *input*)

The forward real to spectral cosinus transform.

References plan_r.

Referenced by test_drs_fftw().

5.6.2.3 subroutine drs_fftw3::dft_backward (double precision,dimension (0:drs_fftw3_nt-1,drs_fftw3_np),intent(in) *input*, double precision,dimension(0:drs_fftw3_nt-1,drs_fftw3_np),intent(out) *output*)

The backward, spectral to real DFT.

References in_p, and plan_pb.

Referenced by drs_transforms::m2phi_2D(), and test_drs_fftw().

5.6.2.4 subroutine drs_fftw3::dft_forward (double precision,dimension (0:drs_fftw3_nt-1,drs_fftw3_np),intent(in) *input*, double precision,dimension(0:drs_fftw3_nt-1,drs_fftw3_np),intent(out) *output*)

The forward real to spectral DFT.

References in_p, and plan_pf.

Referenced by test_drs_fftw(), drs_transforms::ylmt(), and drs_transforms::ylmt_3D().

5.6.2.5 subroutine drs_fftw3::drs_fftw3_cleanup ()

Destroies the plans.

References plan_pb, plan_pf, and plan_r.

Referenced by test_drs_fftw().

5.6.2.6 subroutine drs_fftw3::drs_fftw3_init (integer,intent(in) *Nr*, integer,intent(in) *Nt*, integer,intent(in) *Np*)

Initialises all the fftw3 plans for forward and backward Fourier and cosinus transforms.

Parameters:

Nr Number of points in real space for the radial cosinus transforms.

Nt Perform this many azimuthal Fourier transforms at a time.

Np Number of points in real space for the azimuthal Fourier transforms.

References `drs_fftw3_Np`, `drs_fftw3_Nr`, `drs_fftw3_Nt`, `in_p`, `inout_r`, `plan_pb`, `plan_pf`, and `plan_r`.

Referenced by `drs_init()`, `init()`, `test_drs_fftw()`, `test_drs_radial()`, and `test_saveDXMer()`.

5.6.2.7 subroutine `drs_fftw3::remesh` (`integer,intent(in) Nr1`, `double precision,dimension(nr1),intent(in) f1`, `integer,intent(in) Nr2`, `double precision,dimension(nr2),intent(out) f2`)

Given a field `f1` described at `Nr1` points in an interval, *remesh* outputs the same field, in the same interval at `Nr2` points as `f2`.

Parameters:

f1 The input field.

Nr2 The input and output number of points.

f2 The output field.

Referenced by `drs_io::drs_open_output()`, and `test_drs_fftw()`.

5.6.3 Variable Documentation

5.6.3.1 `integer drs_fftw3::drs_fftw3_Np`

Referenced by `drs_fftw3_init()`.

5.6.3.2 `integer drs_fftw3::drs_fftw3_Nr`

Referenced by `drs_fftw3_init()`.

5.6.3.3 `integer drs_fftw3::drs_fftw3_Nt`

Referenced by `drs_fftw3_init()`.

5.6.3.4 `double precision,dimension(:,:),allocatable drs_fftw3::in_p`

Referenced by `dft_backward()`, `dft_forward()`, and `drs_fftw3_init()`.

5.6.3.5 `double precision,dimension(:,:),allocatable drs_fftw3::inout_r`

Referenced by `drs_fftw3_init()`.

5.6.3.6 `integer*8 drs_fftw3::plan_pb`

Referenced by `dft_backward()`, `drs_fftw3_cleanup()`, and `drs_fftw3_init()`.

5.6.3.7 integer*8 drs_fftw3::plan_pf

Referenced by `dft_forward()`, `drs_fftw3_cleanup()`, and `drs_fftw3_init()`.

5.6.3.8 integer*8 drs_fftw3::plan_r

It makes use of `fftw3`.

Referenced by `cos_r2r_1_n2r()`, `cos_r2r_1_r2n()`, `drs_fftw3_cleanup()`, and `drs_fftw3_init()`.

5.7 drs_field Namespace Reference

Functions

- subroutine **drs_field_allocation** ()
- subroutine **drs_field_init** (field_tor_dr, field_tor_ddr, field_pol_dr, field_pol_ddr)
- subroutine **update_field_tor_lap** ()
- subroutine **update_field_pol_lap** ()
- subroutine **drs_field_random_init** (noise)
- subroutine **apply_field_pol_BC** (pol, l, m)

These lines take care of boundary conditions If the value at a boundary is different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

- subroutine **apply_field_tor_BC** (tor, l, m)

These lines take care of boundary conditions If the value at a boundary is different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

- subroutine **calc_B** (Br_t, Bt_t, Bp_t, rot_Br_t, rot_Bt_t, rot_Bp_t)
- subroutine **calc_field** (Br, Bt, Bp)
- subroutine **calc_rot_field** (rotB_r, rotB_t, rotB_p)
- subroutine **calc_field_lspec** (Bspec)
- subroutine **calc_field_mspec** (Bspec)
- subroutine **calc_field_nspec** (Bspec)

Variables

- double precision, dimension(:, :, :), allocatable **field_pol**
- double precision, dimension(:, :, :), allocatable **field_tor**
- double precision, dimension(:, :, :), allocatable **field_pol_dr**
- double precision, dimension(:, :, :), allocatable **field_tor_dr**
- double precision, dimension(:, :, :), allocatable **field_pol_ddr**
- double precision, dimension(:, :, :), allocatable **field_tor_ddr**
- double precision, dimension(:, :, :), allocatable **field_pol_lap**
- double precision, dimension(:, :, :), allocatable **field_tor_lap**
- double precision, dimension(:, :, :), allocatable **field_pol_avg**
- double precision, dimension(:, :, :), allocatable **field_tor_avg**

5.7.1 Function Documentation

5.7.1.1 subroutine drs_field::apply_field_pol_BC (double precision,dimension(nr),intent(inout) pol, integer,intent(in) l, integer,intent(in) m)

These lines take care of boundary conditions If the value at a boundary is different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

Referenced by drs(), and update_field().

5.7.1.2 subroutine drs_field::apply_field_tor_BC (double precision,dimension(nr),intent(inout) *tor*, integer,intent(in) *l*, integer,intent(in) *m*)

These lines take care of boundary conditions. If the value at a boundary is different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation. If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

References drs_radial::rcoll.

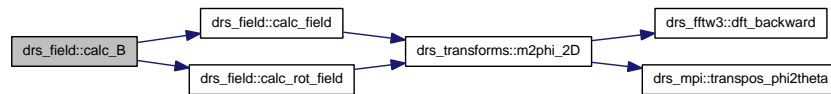
Referenced by drs(), and update_field().

5.7.1.3 subroutine drs_field::calc_B (double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) *Br_t*, double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) *Bt_t*, double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) *Bp_t*, double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) *rot_Br_t*, double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) *rot_Bt_t*, double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) *rot_Bp_t*)

References calc_field(), and calc_rot_field().

Referenced by drs_nonlinear::evaluate_real_space().

Here is the call graph for this function:

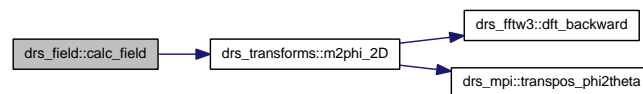


5.7.1.4 subroutine drs_field::calc_field (double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) *Br*, double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) *Bt*, double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) *Bp*)

References drs_mpi::blk_ps_start, drs_legendre::dleg, field_pol, field_pol_dr, field_tor, drs_legendre::leg_sin, drs_legendre::legendre, drs_legendre::llp1, drs_transforms::m2phi_2D(), drs_mpi::mm, drs_radial::rcoll, and drs_radial::rcoll2.

Referenced by Benchmarkv1(), Benchmarkv2(), calc_B(), computeAndSaveAverage(), drs_renderers::render_B(), drs_renderers::render_B_outside(), drs_renderers::render_Bp(), drs_renderers::render_Br(), drs_renderers::render_Bt(), drs_renderers::render_Bz(), StateAverage(), and YokoiPlots().

Here is the call graph for this function:



5.7.1.5 subroutine drs_field::calc_field_lspec (double precision,dimension(0:nt_s),intent(out) *bspec*)

References drs_mpi::blk_ps_size, drs_mpi::blk_ps_start, drs_radial::drcoll, field_pol, drs_legendre::llp1, drs_dims::m0, drs_mpi::mpi_rank, drs_dims::Np_s, drs_dims::Nr, drs_legendre::plmfac, and drs_radial::rcoll2.

Referenced by drs_io::save_l_spec().

5.7.1.6 subroutine drs_field::calc_field_mspect (double precision,dimension(m0*np_s+1),intent(out) *Bspec*)

References drs_mpi::blk_ps_size, drs_mpi::blk_ps_start, drs_radial::drcoll, field_pol, drs_legendre::llp1, drs_mpi::mpi_rank, drs_dims::Nr, drs_dims::Nt_s, drs_legendre::plmfac, and drs_radial::rcoll2.

Referenced by drs_io::save_m_spec().

5.7.1.7 subroutine drs_field::calc_field_nspec (double precision,dimension(nr_s),intent(out) *Bspec*)

References drs_mpi::blk_ps_start, field_pol, drs_legendre::llp1, drs_dims::m0, drs_dims::Np_s, drs_legendre::plmfac, and drs_radial::rcoll2.

Referenced by drs_io::save_n_spec().

5.7.1.8 subroutine drs_field::calc_rot_field (double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *rotB_r*, double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *rotB_t*, double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *rotB_p*)

References drs_mpi::blk_ps_start, drs_legendre::dleg, field_pol_lap, field_tor, field_tor_dr, drs_legendre::leg_sin, drs_legendre::legendre, drs_legendre::llp1, drs_transforms::m2phi_2D(), drs_mpi::mm, drs_radial::rcoll, and drs_radial::rcoll2.

Referenced by calc_B(), computeAndSaveAverage(), StateAverage(), and YokoiPlots().

Here is the call graph for this function:



5.7.1.9 subroutine drs_field::drs_field_allocation ()

References drs_mpi::blk_ps_size, field_pol, field_pol_avg, field_pol_dds, field_pol_dr, field_pol_lap, field_tor, field_tor_avg, field_tor_dds, field_tor_dr, field_tor_lap, drs_mpi::mpi_rank, drs_dims::Nr, and drs_dims::Nt_s.

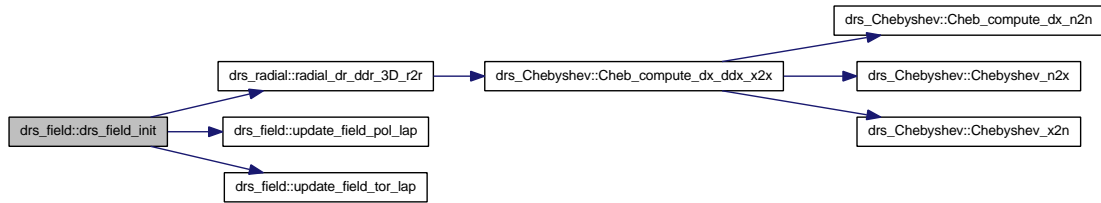
Referenced by drs_init(), and init().

5.7.1.10 subroutine `drs_field::drs_field_init` (double precision,dimension(0:nt_s, blk_ps_size(mpi_rank) field_tor_dr, double precision,dimension(0:nt_s, blk_ps_size(mpi_rank) field_tor_ddr, double precision,dimension(0:nt_s, blk_ps_size(mpi_rank) field_pol_dr, double precision,dimension(0:nt_s, blk_ps_size(mpi_rank) field_pol_ddr)

References `field_pol`, `field_tor`, `drs_radial::radial_dr_ddr_3D_r2r()`, `update_field_pol_lap()`, and `update_field_tor_lap()`.

Referenced by `computeAndSaveAverage()`, `drs_init()`, `init()`, `StateAverage()`, and `YokoiPlots()`.

Here is the call graph for this function:



5.7.1.11 subroutine `drs_field::drs_field_random_init` (double precision,intent(in) noise)

References `drs_mpi::blk_ps_start`, `field_tor`, `drs_dims::m0`, `drs_mpi::mpi_rank`, `drs_dims::Np_s`, `drs_dims::Nr`, `drs_dims::Nt_s`, `drs_legendre::pi`, `drs_legendre::plmfac`, and `drs_radial::rcoll`.

5.7.1.12 subroutine `drs_field::update_field_pol_lap` ()

References `field_pol`, `field_pol_ddr`, `field_pol_lap`, `drs_legendre::llp1`, `drs_dims::Nr`, and `drs_radial::rcoll2`.

Referenced by `drs_field_init()`, and `update_field()`.

5.7.1.13 subroutine `drs_field::update_field_tor_lap` ()

References `field_tor`, `field_tor_ddr`, `field_tor_lap`, `drs_legendre::llp1`, `drs_dims::Nr`, and `drs_radial::rcoll2`.

Referenced by `drs_field_init()`, and `update_field()`.

5.7.2 Variable Documentation

5.7.2.1 double precision,dimension(:, :,),allocatable `drs_field::field_pol`

Referenced by `drs_probes::average_unnormalised_field_l_spectrum()`, `Benchmarkv1()`, `calc_field()`, `calc_field_lspec()`, `calc_field_mspec()`, `calc_field_nspec()`, `computeAndSaveAverage()`, `drs()`, `drs_field_allocation()`, `drs_field_init()`, `drs_io::drs_load_state()`, `kd_grothrate()`, `drs_probes::measure_lm()`, `drs_renderers::render_B()`, `drs_renderers::render_B_outside()`, `drs_renderers::render_Bp()`, `drs_renderers::render_Br()`, `drs_renderers::render_Bt()`, `drs_renderers::render_Bz()`, `drs_probes::save_field_coeffs()`, `drs_probes::save_magnetic_dissipation()`, `drs_io::save_state()`, `StateAverage()`, `update_field()`, `update_field_pol_lap()`, and `YokoiPlots()`.

5.7.2.2 double precision,dimension(:, :, :),allocatable drs_field::field_pol_avg

Referenced by computeAndSaveAverage(), drs_field_allocation(), StateAverage(), and YokoiPlots().

5.7.2.3 double precision,dimension(:, :, :),allocatable drs_field::field_pol_ddd

Referenced by computeAndSaveAverage(), drs(), drs_field_allocation(), drs_init(), init(), kd_grothrate(), drs_renderers::render_B(), drs_renderers::render_B_outside(), drs_renderers::render_Bp(), drs_renderers::render_Br(), drs_renderers::render_Bt(), drs_renderers::render_Bz(), drs_probes::save_magnetic_dissipation(), StateAverage(), update_field(), update_field_pol_lap(), and YokoiPlots().

5.7.2.4 double precision,dimension(:, :, :),allocatable drs_field::field_pol_dr

Referenced by calc_field(), computeAndSaveAverage(), drs(), drs_field_allocation(), drs_init(), init(), drs_probes::measure_lm(), drs_renderers::render_B(), drs_renderers::render_B_outside(), drs_renderers::render_Bp(), drs_renderers::render_Br(), drs_renderers::render_Bt(), drs_renderers::render_Bz(), drs_probes::save_magnetic_dissipation(), StateAverage(), update_field(), and YokoiPlots().

5.7.2.5 double precision,dimension(:, :, :),allocatable drs_field::field_pol_lap

Referenced by calc_rot_field(), drs_field_allocation(), update_field(), and update_field_pol_lap().

5.7.2.6 double precision,dimension(:, :, :),allocatable drs_field::field_tor

Referenced by Benchmarkv1(), calc_field(), calc_rot_field(), computeAndSaveAverage(), drs(), drs_field_allocation(), drs_field_init(), drs_field_random_init(), drs_io::drs_load_state(), kd_grothrate(), drs_probes::measure_lm(), drs_renderers::render_B(), drs_renderers::render_B_outside(), drs_renderers::render_Bp(), drs_renderers::render_Br(), drs_renderers::render_Bt(), drs_renderers::render_Bz(), drs_probes::save_field_coeffs(), drs_probes::save_magnetic_dissipation(), drs_io::save_state(), StateAverage(), update_field(), update_field_tor_lap(), and YokoiPlots().

5.7.2.7 double precision,dimension(:, :, :),allocatable drs_field::field_tor_avg

Referenced by computeAndSaveAverage(), drs_field_allocation(), StateAverage(), and YokoiPlots().

5.7.2.8 double precision,dimension(:, :, :),allocatable drs_field::field_tor_ddd

Referenced by computeAndSaveAverage(), drs(), drs_field_allocation(), drs_init(), init(), kd_grothrate(), drs_renderers::render_B(), drs_renderers::render_B_outside(), drs_renderers::render_Bp(), drs_renderers::render_Br(), drs_renderers::render_Bt(), drs_renderers::render_Bz(), drs_probes::save_magnetic_dissipation(), StateAverage(), update_field(), update_field_tor_lap(), and YokoiPlots().

5.7.2.9 double precision,dimension(:, :, :),allocatable drs_field::field_tor_dr

Referenced by calc_rot_field(), computeAndSaveAverage(), drs(), drs_field_allocation(), drs_init(), init(), drs_renderers::render_B(), drs_renderers::render_B_outside(), drs_renderers::render_Bp(), drs_renderers::render_Br(), drs_renderers::render_Bt(), drs_renderers::render_Bz(), drs_probes::save_magnetic_dissipation(), StateAverage(), update_field(), and YokoiPlots().

5.7.2.10 double precision,dimension(:, :, :),allocatable drs_field::field_tor_lap

Referenced by `drs_field_allocation()`, `drs_probes::save_magnetic_dissipation()`, `update_field()`, and `update_field_tor_lap()`.

5.8 drs_flow Namespace Reference

Functions

- subroutine **drs_flow_allocation** ()
- subroutine **drs_flow_init** (flow_tor_dr, flow_tor_dds, flow_pol_dr, flow_pol_dds)
- subroutine **update_flow_tor_lap** ()
- subroutine **update_flow_pol_lap** ()
- subroutine **apply_flow_pol_BC** (pol)

These lines take care of boundary conditions If the value at a boundary is bc different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

- subroutine **apply_flow_tor_BC** (tor)

These lines take care of boundary conditions If the value at a boundary is bc different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

- subroutine **calc_u** (ur, ut, up, rotu_r, rotu_t, rotu_p)

Abstracts computing the flow and its curl in real space.

- subroutine **calc_flow** (ur_t, ut_t, up_t)

This routine computes:.

- subroutine **calc_rot_flow** (rotu_r, rotu_t, rotu_p)

This routine computes:.

- subroutine **calc_flow_lspec** (uspec)

Computes the l-spectrum of the radial flow.

- subroutine **calc_flow_mspec** (uspec)

Computes the m-spectrum of the radial flow.

- subroutine **calc_flow_nspec** (uspec)

Computes the n-spectrum of the radial flow.

Variables

- double precision, dimension(:, :, :), allocatable **flow_pol**
- double precision, dimension(:, :, :), allocatable **flow_tor**
- double precision, dimension(:, :, :), allocatable **flow_pol_dr**
- double precision, dimension(:, :, :), allocatable **flow_tor_dr**
- double precision, dimension(:, :, :), allocatable **flow_pol_dds**
- double precision, dimension(:, :, :), allocatable **flow_tor_dds**
- double precision, dimension(:, :, :), allocatable **flow_pol_lap**
- double precision, dimension(:, :, :), allocatable **flow_tor_lap**
- double precision, dimension(:, :, :), allocatable **flow_pol_avg**
- double precision, dimension(:, :, :), allocatable **flow_tor_avg**

5.8.1 Function Documentation

5.8.1.1 subroutine drs_flow::apply_flow_pol_BC (double precision,dimension(nr),intent(inout) pol)

These lines take care of boundary conditions If the value at a boundary is bc different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

Referenced by drs(), and update_flow().

5.8.1.2 subroutine drs_flow::apply_flow_tor_BC (double precision,dimension(nr),intent(inout) tor)

These lines take care of boundary conditions If the value at a boundary is bc different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

Referenced by drs(), and update_flow().

5.8.1.3 subroutine drs_flow::calc_flow (double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) ur_t, double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) ut_t, double precision,dimension(0:blk_t_size(mpi_rank)),intent(out) up_t)

This routine computes:

$$\vec{u} = \vec{\nabla} \times (\vec{\nabla} \times (\vec{r} flow_pol)) + \vec{\nabla} \times (r \vec{r} flow_tor)$$

~~~~~ the fields are defined as: field in program field in equation ----- flow\_pol = P flow\_tor = T/r P id the poloidal scalar, and T the Toroidal scalar

input: (not modified on output) common /fields/ flow\_tor,flow\_pol,.. (lmr) common /derivatives/ flow\_tor\_dr,..,flow\_pol\_dr,flow\_pol\_ddr,.. (lmr) output: (theta,phi,r, transposed): ur\_t,ut\_t,up\_t ~~~~~

References drs\_mpi::blk\_ps\_start, drs\_legendre::dleg, flow\_pol, flow\_pol\_dr, flow\_tor, drs\_legendre::leg\_sin, drs\_legendre::legendre, drs\_legendre::llp1, drs\_transforms::m2phi\_2D(), drs\_mpi::mm, and drs\_radial::rcoll.

Referenced by Benchmarkv1(), Benchmarkv2(), calc\_u(), computeAndSaveAverage(), drs\_io::dump\_state(), drs\_renderers::render\_helicity(), drs\_renderers::render\_u(), drs\_renderers::render\_up(), drs\_renderers::render\_ur(), drs\_renderers::render\_ut(), drs\_renderers::render\_uz(), StateAverage(), and YokoiPlots().

Here is the call graph for this function:



### 5.8.1.4 subroutine drs\_flow::calc\_flow\_lspec (double precision,dimension(0:nt\_s),intent(out) uspec)

Computes the l-spectrum of the radial flow.

References `drs_mpi::blk_ps_size`, `drs_mpi::blk_ps_start`, `drs_radial::drcoll`, `flow_pol`, `drs_legendre::llp1`, `drs_dims::m0`, `drs_mpi::mpi_rank`, `drs_dims::Np_s`, `drs_dims::Nr`, `drs_legendre::plmfac`, and `drs_radial::rcoll`.

Referenced by `drs_io::save_l_spec()`.

#### 5.8.1.5 subroutine `drs_flow::calc_flow_mspec` (double precision,dimension(m0\*np\_s+1),intent(out) *uspec*)

Computes the m-spectrum of the radial flow.

References `drs_mpi::blk_ps_size`, `drs_mpi::blk_ps_start`, `drs_radial::drcoll`, `flow_pol`, `drs_legendre::llp1`, `drs_mpi::mpi_rank`, `drs_dims::Nr`, `drs_dims::Nt_s`, `drs_legendre::plmfac`, and `drs_radial::rcoll`.

Referenced by `drs_io::save_m_spec()`.

#### 5.8.1.6 subroutine `drs_flow::calc_flow_nspec` (double precision,dimension(nr\_s),intent(out) *uspec*)

Computes the n-spectrum of the radial flow.

References `drs_mpi::blk_ps_start`, `flow_pol`, `drs_legendre::llp1`, `drs_dims::m0`, `drs_dims::Np_s`, `drs_legendre::plmfac`, and `drs_radial::rcoll`.

Referenced by `drs_io::save_n_spec()`.

#### 5.8.1.7 subroutine `drs_flow::calc_rot_flow` (double precision,dimension(0:blk\_t\_size(mpi\_rank)),intent(out) *rotu\_r*, double precision,dimension(0:blk\_t\_size(mpi\_rank)),intent(out) *rotu\_t*, double precision,dimension(0:blk\_t\_size(mpi\_rank)),intent(out) *rotu\_p*)

This routine computes:  $\text{rotu} = \text{rot}(\text{rotrot}(rP) + \text{rot}(rT))$  ~~~~~ the fields are defined as: field in program field in equation ----- flow\_pol = P flow\_tor = T/r P is the poloidal scalar, and T the Toroidal scalar

input: (not modified on output) common /fields/ flow\_tor,flow\_pol,.. (lmr) common /derivatives/ flow\_tor\_dr,..,flow\_pol\_dr,flow\_pol\_ddr,.. (lmr) output: (theta,phi,r, transposed): *ur\_t*,*ut\_t*,*up\_t*,*rotu\_r\_t*,*rotu\_t\_t*,*rotu\_p\_t* ~~~~~

References `drs_mpi::blk_ps_start`, `drs_legendre::dleg`, `flow_pol_dr`, `flow_pol_lap`, `flow_tor`, `flow_tor_dr`, `drs_legendre::leg_sin`, `drs_legendre::legendre`, `drs_legendre::llp1`, `drs_transforms::m2phi_2D()`, `drs_mpi::mm`, and `drs_radial::rcoll`.

Referenced by `calc_u()`, `computeAndSaveAverage()`, `drs_renderers::render_rotu()`, `drs_renderers::render_rotu_p()`, `drs_renderers::render_rotu_r()`, `drs_renderers::render_rotu_t()`, `drs_renderers::render_rotu_z()`, `StateAverage()`, and `YokoiPlots()`.

Here is the call graph for this function:



**5.8.1.8 subroutine drs\_flow::calc\_u** (double precision,dimension(0:blk\_t\_size(mpi\_rank),intent(out) *ur*, double precision,dimension(0:blk\_t\_size(mpi\_rank),intent(out) *ut*, double precision,dimension(0:blk\_t\_size(mpi\_rank),intent(out) *up*, double precision,dimension(0:blk\_t\_size(mpi\_rank),intent(out) *rotu\_r*, double precision,dimension(0:blk\_t\_size(mpi\_rank),intent(out) *rotu\_t*, double precision,dimension(0:blk\_t\_size(mpi\_rank),intent(out) *rotu\_p*)

Abstracts computing the flow and its curl in real space.

References calc\_flow(), and calc\_rot\_flow().

Referenced by drs\_nonlinear::evaluate\_real\_space().

Here is the call graph for this function:



**5.8.1.9 subroutine drs\_flow::drs\_flow\_allocation ()**

References drs\_mpi::blk\_ps\_size, flow\_pol, flow\_pol\_avg, flow\_pol\_ddr, flow\_pol\_dr, flow\_pol\_lap, flow\_tor, flow\_tor\_avg, flow\_tor\_ddr, flow\_tor\_dr, flow\_tor\_lap, drs\_mpi::mpi\_rank, drs\_dims::Nr, and drs\_dims::Nt\_s.

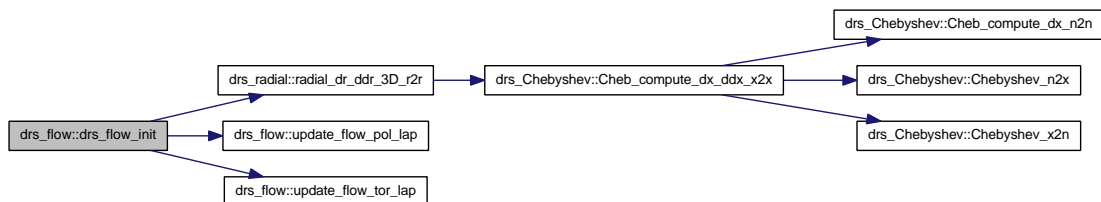
Referenced by drs\_init(), and init().

**5.8.1.10 subroutine drs\_flow::drs\_flow\_init** (double precision,dimension(0:nt\_s, blk\_ps\_size(mpi\_rank) *flow\_tor\_dr*, double precision,dimension(0:nt\_s, blk\_ps\_size(mpi\_rank) *flow\_tor\_ddr*, double precision,dimension(0:nt\_s, blk\_ps\_size(mpi\_rank) *flow\_pol\_dr*, double precision,dimension(0:nt\_s, blk\_ps\_size(mpi\_rank) *flow\_pol\_ddr*)

References flow\_pol, flow\_tor, drs\_radial::radial\_dr\_ddr\_3D\_r2r(), update\_flow\_pol\_lap(), and update\_flow\_tor\_lap().

Referenced by computeAndSaveAverage(), drs\_init(), init(), StateAverage(), and YokoiPlots().

Here is the call graph for this function:



**5.8.1.11 subroutine drs\_flow::update\_flow\_pol\_lap ()**

References flow\_pol, flow\_pol\_ddr, flow\_pol\_lap, drs\_legendre::llp1, drs\_dims::Nr, and drs\_radial::rcoll2.

Referenced by `drs_flow_init()`, and `update_flow()`.

#### 5.8.1.12 subroutine `drs_flow::update_flow_tor_lap()`

References `flow_tor`, `flow_tor_ddr`, `flow_tor_lap`, `drs_legendre::llp1`, `drs_dims::Nr`, and `drs_radial::rcoll2`.

Referenced by `drs_flow_init()`, and `update_flow()`.

### 5.8.2 Variable Documentation

#### 5.8.2.1 double precision,dimension(:, :, :),allocatable `drs_flow::flow_pol`

Referenced by `applyGreen()`, `drs_probes::average_unnormalised_flow_l_spectrum()`, `Benchmarkv1()`, `calc_flow()`, `calc_flow_lspec()`, `calc_flow_mspec()`, `calc_flow_nspec()`, `computeAndSaveAverage()`, `drs()`, `drs_flow_allocation()`, `drs_flow_init()`, `drs_io::drs_load_state()`, `drs_io::dump_state()`, `drs_probes::measure_lm()`, `drs_renderers::render_helicity()`, `drs_renderers::render_poloidal_streamlines()`, `drs_renderers::render_rotu()`, `drs_renderers::render_rotu_p()`, `drs_renderers::render_rotu_r()`, `drs_renderers::render_rotu_t()`, `drs_renderers::render_rotu_z()`, `drs_renderers::render_streamlines_t()`, `drs_renderers::render_u()`, `drs_renderers::render_up()`, `drs_renderers::render_ur()`, `drs_renderers::render_ut()`, `drs_renderers::render_uz()`, `drs_probes::save_flow_coeffs()`, `drs_probes::save_flow_dissipation()`, `drs_io::save_state()`, `drs_nonlinear::save_stuff()`, `StateAverage()`, `update_flow()`, `update_flow_pol_lap()`, and `YokoiPlots()`.

#### 5.8.2.2 double precision,dimension(:, :, :),allocatable `drs_flow::flow_pol_avg`

Referenced by `computeAndSaveAverage()`, `drs_flow_allocation()`, `drs_io::dump_state()`, `drs_probes::measure_lm()`, `StateAverage()`, and `YokoiPlots()`.

#### 5.8.2.3 double precision,dimension(:, :, :),allocatable `drs_flow::flow_pol_ddr`

Referenced by `applyGreen()`, `computeAndSaveAverage()`, `drs()`, `drs_flow_allocation()`, `drs_init()`, `drs_io::dump_state()`, `init()`, `drs_renderers::render_helicity()`, `drs_renderers::render_rotu()`, `drs_renderers::render_rotu_p()`, `drs_renderers::render_rotu_r()`, `drs_renderers::render_rotu_t()`, `drs_renderers::render_rotu_z()`, `drs_renderers::render_u()`, `drs_renderers::render_up()`, `drs_renderers::render_ur()`, `drs_renderers::render_ut()`, `drs_renderers::render_uz()`, `drs_probes::save_flow_dissipation()`, `StateAverage()`, `update_flow()`, `update_flow_pol_lap()`, and `YokoiPlots()`.

#### 5.8.2.4 double precision,dimension(:, :, :),allocatable `drs_flow::flow_pol_dr`

Referenced by `applyGreen()`, `calc_flow()`, `calc_rot_flow()`, `computeAndSaveAverage()`, `drs()`, `drs_flow_allocation()`, `drs_init()`, `drs_io::dump_state()`, `init()`, `drs_probes::measure_lm()`, `drs_renderers::render_helicity()`, `drs_renderers::render_rotu()`, `drs_renderers::render_rotu_p()`, `drs_renderers::render_rotu_r()`, `drs_renderers::render_rotu_t()`, `drs_renderers::render_rotu_z()`, `drs_renderers::render_u()`, `drs_renderers::render_up()`, `drs_renderers::render_ur()`, `drs_renderers::render_ut()`, `drs_renderers::render_uz()`, `drs_probes::save_flow_dissipation()`, `StateAverage()`, `update_flow()`, and `YokoiPlots()`.

#### 5.8.2.5 double precision,dimension(:, :, :),allocatable `drs_flow::flow_pol_lap`

Referenced by `calc_rot_flow()`, `drs_flow_allocation()`, `update_flow()`, and `update_flow_pol_lap()`.

**5.8.2.6 double precision,dimension(:, :, :),allocatable drs\_flow::flow\_tor**

Referenced by Benchmarkv1(), calc\_flow(), calc\_rot\_flow(), computeAndSaveAverage(), drs(), drs\_flow\_allocation(), drs\_flow\_init(), drs\_io::drs\_load\_state(), drs\_io::dump\_state(), drs\_probes::measure\_lm(), drs\_renderers::render\_helicity(), drs\_renderers::render\_radial\_streamfunction(), drs\_renderers::render\_rotu(), drs\_renderers::render\_rotu\_p(), drs\_renderers::render\_rotu\_r(), drs\_renderers::render\_rotu\_t(), drs\_renderers::render\_rotu\_z(), drs\_renderers::render\_u(), drs\_renderers::render\_up(), drs\_renderers::render\_ur(), drs\_renderers::render\_ut(), drs\_renderers::render\_uz(), drs\_probes::save\_flow\_coeffs(), drs\_probes::save\_flow\_dissipation(), drs\_io::save\_state(), StateAverage(), update\_flow(), update\_flow\_tor\_lap(), and YokoiPlots().

**5.8.2.7 double precision,dimension(:, :, :),allocatable drs\_flow::flow\_tor\_avg**

Referenced by computeAndSaveAverage(), drs\_flow\_allocation(), drs\_io::dump\_state(), drs\_probes::measure\_lm(), StateAverage(), and YokoiPlots().

**5.8.2.8 double precision,dimension(:, :, :),allocatable drs\_flow::flow\_tor\_ddr**

Referenced by computeAndSaveAverage(), drs(), drs\_flow\_allocation(), drs\_init(), drs\_io::dump\_state(), init(), drs\_renderers::render\_helicity(), drs\_renderers::render\_rotu(), drs\_renderers::render\_rotu\_p(), drs\_renderers::render\_rotu\_r(), drs\_renderers::render\_rotu\_t(), drs\_renderers::render\_rotu\_z(), drs\_renderers::render\_u(), drs\_renderers::render\_up(), drs\_renderers::render\_ur(), drs\_renderers::render\_ut(), drs\_renderers::render\_uz(), drs\_probes::save\_flow\_dissipation(), StateAverage(), update\_flow(), update\_flow\_tor\_lap(), and YokoiPlots().

**5.8.2.9 double precision,dimension(:, :, :),allocatable drs\_flow::flow\_tor\_dr**

Referenced by calc\_rot\_flow(), computeAndSaveAverage(), drs(), drs\_flow\_allocation(), drs\_init(), drs\_io::dump\_state(), init(), drs\_renderers::render\_helicity(), drs\_renderers::render\_rotu(), drs\_renderers::render\_rotu\_p(), drs\_renderers::render\_rotu\_r(), drs\_renderers::render\_rotu\_t(), drs\_renderers::render\_rotu\_z(), drs\_renderers::render\_u(), drs\_renderers::render\_up(), drs\_renderers::render\_ur(), drs\_renderers::render\_ut(), drs\_renderers::render\_uz(), drs\_probes::save\_flow\_dissipation(), StateAverage(), update\_flow(), and YokoiPlots().

**5.8.2.10 double precision,dimension(:, :, :),allocatable drs\_flow::flow\_tor\_lap**

Referenced by drs\_flow\_allocation(), update\_flow(), and update\_flow\_tor\_lap().

## 5.9 drs\_hypDiff Namespace Reference

### Classes

- interface **drs\_apply\_hypDiff**

### Functions

- subroutine **drs\_hypDiff\_init** (*Nt*)

### Variables

- double precision, allocatable **hypDiff**
- logical **drs\_want\_hypDiff** = .FALSE.

#### 5.9.1 Function Documentation

##### 5.9.1.1 subroutine **drs\_hypDiff::drs\_hypDiff\_init** (*integer,intent(in) Nt*)

References **drs\_want\_hypDiff**, and **hypDiff**.

Referenced by **drs\_init()**, and **init()**.

#### 5.9.2 Variable Documentation

##### 5.9.2.1 logical **drs\_hypDiff::drs\_want\_hypDiff** = .FALSE.

Referenced by **drs\_hypDiff\_init()**, **drs\_init()**, and **init()**.

##### 5.9.2.2 double precision,allocatable **drs\_hypDiff::hypDiff**

Referenced by **drs\_hypDiff\_init()**.



## 5.10 drs\_io Namespace Reference

Deals with input and output of state files and derived quantities.

### Functions

- subroutine **drs\_load\_state** (error)  
*Reads a state performing interpolation as needed. The state is stored in the files with name given by io\_calc\_file\_in and are described by the file with extension .par.*
- subroutine **drs\_open\_output** ()  
*Opens units for regularly probed quantities to be saved.*
- subroutine **dump\_state** ()
- subroutine **save\_state** ()  
*Saves the present state to file. At this point all files are saved with the file name given by io\_calc\_file\_out.*
- subroutine **save\_l\_spec** ()  
*Saves the normalized power spectra with respect to l.*
- subroutine **save\_m\_spec** ()  
*Saves the normalized power spectra with respect to m.*
- subroutine **save\_n\_spec** ()  
*Saves the normalized power spectra of all quantities with respect to n.*

### Variables

- character(len=60) **io\_calc\_file\_in**
- character(len=60) **io\_calc\_file\_out**
- character(len=13), parameter **deflate**
- character(len=15), parameter **inflate**

#### 5.10.1 Detailed Description

Deals with input and output of state files and derived quantities.

#### 5.10.2 Function Documentation

##### 5.10.2.1 subroutine drs\_io::drs\_load\_state (integer,intent(out) error)

Reads a state performing interpolation as needed. The state is stored in the files with name given by io\_calc\_file\_in and are described by the file with extension .par.

References drs\_comp::comp, drs\_io\_par::etai, drs\_field::field\_pol, drs\_field::field\_tor, drs\_flow::flow\_pol, drs\_flow::flow\_tor, io\_calc\_file\_in, drs\_io\_par::lformi, drs\_io\_par::lsymmi, drs\_dims::m0, drs\_io\_par::m0i, drs\_mpi::mpi\_rank, drs\_io\_par::Npi, drs\_io\_par::Npi\_s, drs\_dims::Nr, drs\_io\_par::Nri, drs\_io\_par::Nri\_s, drs\_io\_par::Nti, drs\_io\_par::Nti\_s, drs\_io\_par::Pmi, drs\_io\_par::Pti, drs\_io\_par::Ra\_ti,

drs\_io\_par::read\_input\_par(), drs\_time::steps, drs\_time::stepstart, drs\_io\_par::Tai, drs\_temp::temp, drs\_time::time, drs\_time::time\_start, and drs\_io\_par::usr\_dimsi.

Referenced by computeAndSaveAverage(), drs\_init(), getProfile(), init(), StateAverage(), and YokoiPlots().

Here is the call graph for this function:



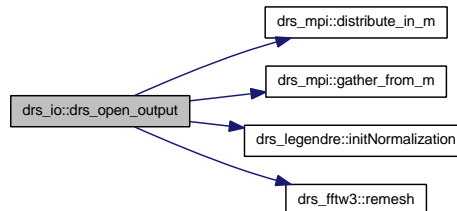
### 5.10.2.2 subroutine drs\_io::drs\_open\_output ()

Opens units for regularly probed quantities to be saved.

References drs\_mpi::blk\_ps\_start, drs\_mpi::distribute\_in\_m(), drs\_mpi::gather\_from\_m(), drs\_legendre::initNormalization(), io\_calc\_file\_out, drs\_io\_par::lformi, drs\_dims::m0, drs\_io\_par::m0i, drs\_io\_par::magici, drs\_mpi::mpi\_rank, drs\_dims::Np\_s, drs\_io\_par::Npi\_s, drs\_io\_par::Nti\_s, drs\_fftw3::remesh(), drs\_io\_units::unit\_am, drs\_io\_units::unit\_cfl, drs\_io\_units::unit\_dissB, drs\_io\_units::unit\_dissu, drs\_io\_units::unit\_eb, drs\_io\_units::unit\_ek, drs\_io\_units::unit\_evp, drs\_io\_units::unit\_evt, drs\_io\_units::unit\_koeb, drs\_io\_units::unit\_koeu, drs\_io\_units::unit\_nu, and drs\_io\_units::unit\_u\_mid.

Referenced by drs\_init().

Here is the call graph for this function:

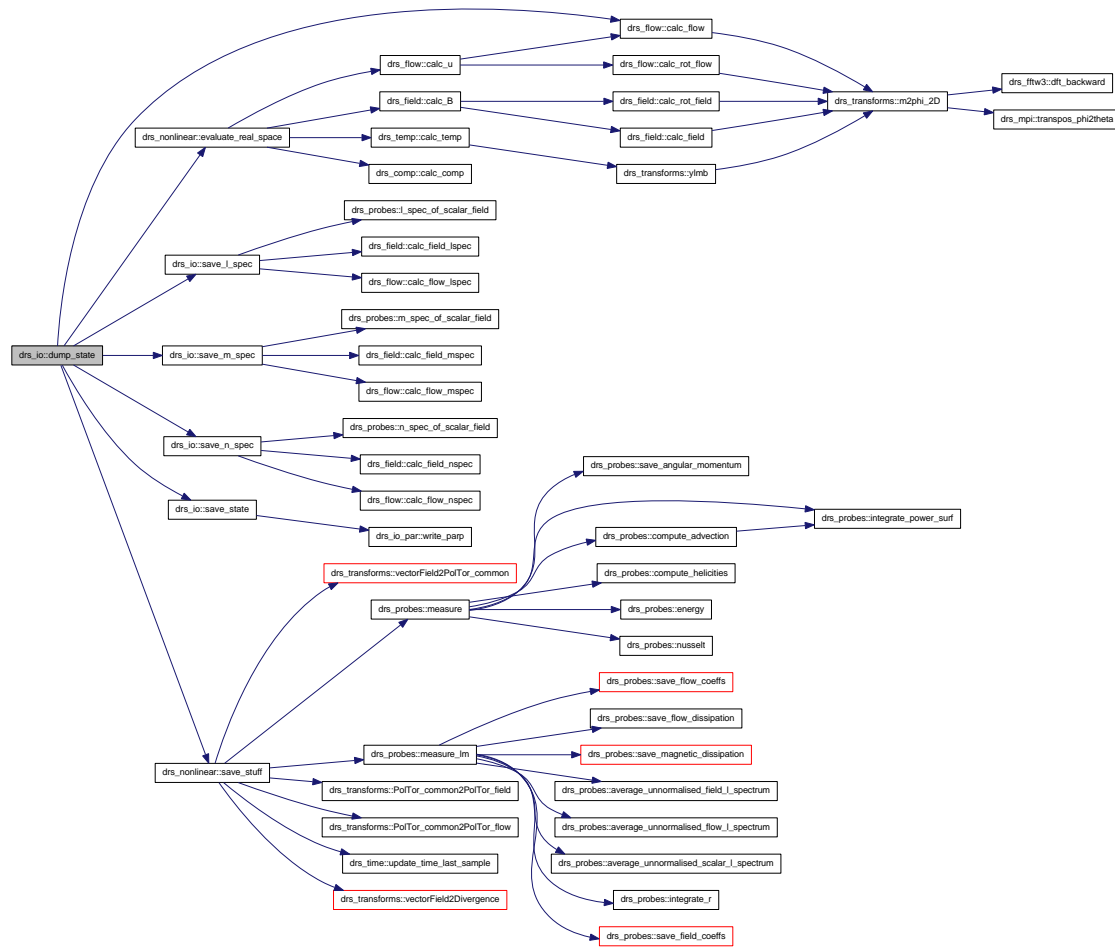


### 5.10.2.3 subroutine drs\_io::dump\_state ()

References drs\_probes::adv\_avg, drs\_mpi::blk\_ps\_size, drs\_mpi::blk\_ps\_start, drs\_mpi::blk\_t\_start, drs\_flow::calc\_flow(), drs\_nonlinear::evaluate\_real\_space(), drs\_flow::flow\_pol, drs\_flow::flow\_pol\_avg, drs\_flow::flow\_pol\_ddr, drs\_flow::flow\_pol\_dr, drs\_flow::flow\_tor, drs\_flow::flow\_tor\_avg, drs\_flow::flow\_tor\_ddr, drs\_flow::flow\_tor\_dr, io\_calc\_file\_out, drs\_time::nsample, save\_l\_spec(), save\_m\_spec(), save\_n\_spec(), save\_state(), drs\_nonlinear::save\_stuff(), drs\_time::steps, drs\_probes::t2\_avg, drs\_temp::temp\_avg, drs\_temp::temp\_dr\_avg, drs\_temp::temp\_profile, drs\_time::time, drs\_time::time\_start, drs\_probes::up2, drs\_probes::up\_avg, drs\_probes::ur\_avg, drs\_probes::ut2, and drs\_probes::ut\_avg.

Referenced by drs().

Here is the call graph for this function:



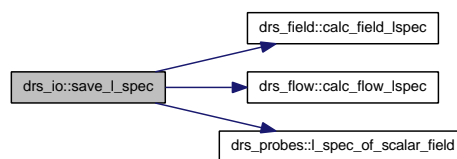
#### 5.10.2.4 subroutine drs\_io::save\_l\_spec ()

Saves the normalized power spectra with respect to l.

References drs\_field::calc\_field\_lspec(), drs\_flow::calc\_flow\_lspec(), drs\_comp::comp, io\_calc\_file\_out, drs\_probes::l\_spec\_of\_scalar\_field(), drs\_mpi::mpi\_rank, drs\_temp::temp, and drs\_io\_units::unit\_lspec.

Referenced by dump\_state(), and StateAverage().

Here is the call graph for this function:



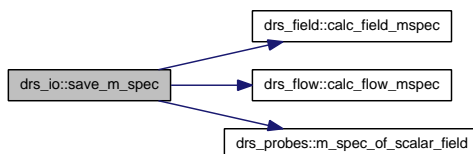
### 5.10.2.5 subroutine `drs_io::save_m_spec()`

Saves the normalized power spectra with respect to m.

References `drs_field::calc_field_mspec()`, `drs_flow::calc_flow_mspec()`, `drs_comp::comp`, `io_calc_file_out`, `drs_probes::m_spec_of_scalar_field()`, `drs_mpi::mpi_rank`, `drs_temp::temp`, and `drs_io_units::unit_mspec`.

Referenced by `dump_state()`, and `StateAverage()`.

Here is the call graph for this function:



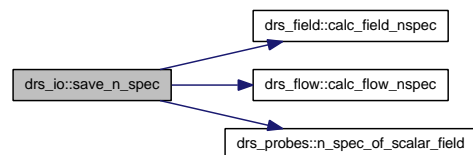
### 5.10.2.6 subroutine `drs_io::save_n_spec()`

Saves the normalized power spectra of all quantities with respect to n.

References `drs_field::calc_field_nspec()`, `drs_flow::calc_flow_nspec()`, `drs_comp::comp`, `io_calc_file_out`, `drs_mpi::mpi_rank`, `drs_probes::n_spec_of_scalar_field()`, `drs_temp::temp`, and `drs_io_units::unit_nspec`.

Referenced by `dump_state()`, and `StateAverage()`.

Here is the call graph for this function:



### 5.10.2.7 subroutine `drs_io::save_state()`

Saves the present state to file. At this point all files are saved with the file name given by `io_calc_file_out`.

#### Todo

Describe the format of the following files.

References `drs_comp::comp`, `drs_field::field_pol`, `drs_field::field_tor`, `drs_flow::flow_pol`, `drs_flow::flow_tor`, `io_calc_file_out`, `drs_mpi::mpi_rank`, `drs_temp::temp`, and `drs_io_par::write_parp()`.

Referenced by `Benchmarkv1()`, `computeAndSaveAverage()`, `dump_state()`, and `StateAverage()`.

Here is the call graph for this function:



### 5.10.3 Variable Documentation

#### 5.10.3.1 `character(len=13),parameter drs_io::deflate`

Referenced by `computeAndSaveAverage()`, `getProfile()`, `init()`, `StateAverage()`, and `YokoiPlots()`.

#### 5.10.3.2 `character(len=15),parameter drs_io::inflate`

Referenced by `computeAndSaveAverage()`, `getProfile()`, `init()`, `StateAverage()`, and `YokoiPlots()`.

#### 5.10.3.3 `character(len=60) drs_io::io_calc_file_in`

Referenced by `Benchmarkv1()`, `Benchmarkv2()`, `computeAndSaveAverage()`, `drs2dx()`, `drs_init()`, `drs_load_state()`, `getProfile()`, `init()`, `parse_drs2dx()`, `parseConfig()`, `StateAverage()`, and `YokoiPlots()`.

#### 5.10.3.4 `character(len=60) drs_io::io_calc_file_out`

Referenced by `computeAndSaveAverage()`, `drs_init()`, `drs_open_output()`, `dump_state()`, `init()`, `save_l_spec()`, `save_m_spec()`, `save_n_spec()`, `save_state()`, and `StateAverage()`.

## 5.11 drs\_io\_DX Namespace Reference

### Classes

- interface **save2DX**

### Functions

- subroutine **save2DXscalar** (field, filename)  
*Saves the contents of a scalar field to file.*
- subroutine **save2DXvector** (XX, YY, ZZ, filename)  
*Saves the contents of a vector field to file given its three components.*
- subroutine **saveDXmeridional** (field, filename)  
*Writes a meridional slice of the field.*
- subroutine **saveDXmeridional3DVec** (field\_x, field\_y, field\_z, filename)  
*Writes a meridional slice of the field.*
- subroutine **saveDXvolume** (field, filename)  
*Writes a volume render of the field.*
- subroutine **saveDXvolume\_v2** (field, filename)  
*Writes a volume render of the field.*
- subroutine **saveDXvolume3DVec** (XX, YY, ZZ, filename)  
*Writes a volume render of the vector field components.*

### Variables

- double precision **cut\_phi**  
*the azimuth to use on meridional cuts*
- double precision **cut\_z**  
*the azimuth to use on equator parallell cuts*
- double precision **where\_to\_cut** = 0.0d0
- integer **cut\_type**  
*the type of cut or render to save*

#### 5.11.1 Function Documentation

- 5.11.1.1 subroutine drs\_io\_DX::save2DXscalar** (double precision,dimension(0:(blk\_t\_size(mpi\_rank)),intent(in) *field*, character(len=\*),intent(in) *filename*)

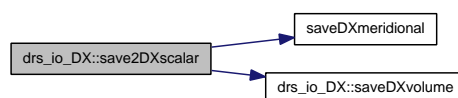
Saves the contents of a scalar field to file.

is a field in real (tpr) space.

**filename** is the base name for the output files.

References cut\_phi, cut\_type, saveDXmeridional(), saveDXvolume(), and where\_to\_cut.

Here is the call graph for this function:



**5.11.1.2 subroutine drs\_io\_DX::save2DXvector** (double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *XX*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *YY*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *ZZ*, character(len=\*),intent(in) *filename*)

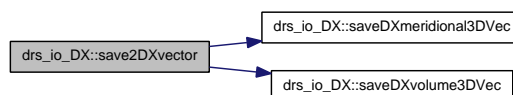
Saves the contents of a vector field to file given its three components.

***XX*, *YY*, *ZZ*** are the real space components of the vector field

**filename** is the base name for the output files.

References cut\_phi, cut\_type, saveDXmeridional3DVec(), saveDXvolume3DVec(), and where\_to\_cut.

Here is the call graph for this function:



**5.11.1.3 subroutine drs\_io\_DX::saveDXmeridional** (double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *field*, character(len=\*),intent(in) *filename*)

Writes a meridional slice of the field.

**field**

into the files with basename

**filename.**

References drs\_legendre::costheta, cut\_phi, drs\_probes::dOmega, drs\_dims::m0, drs\_dims::Nt, drs\_legendre::pi, and drs\_radial::rcoll.

**5.11.1.4** subroutine `drs_io_DX::saveDXmeridional3DVec` (double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *field\_x*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *field\_y*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *field\_z*, character(len=\*),intent(in) *filename*)

Writes a meridional slice of the field.

**field**

into the files with basename

**filename.**

References `drs_legendre::costheta`, `cut_phi`, `drs_dims::m0`, `drs_dims::Nt`, `drs_legendre::pi`, and `drs_radial::rcoll`.

Referenced by `save2DXvector()`.

**5.11.1.5** subroutine `drs_io_DX::saveDXvolume` (double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *field*, character(len=\*),intent(in) *filename*)

Writes a volume render of the field.

**field**

into the files with basename

**filename.**

References `drs_legendre::costheta`, `drs_dims::m0`, `drs_dims::Nt`, `drs_legendre::pi`, and `drs_radial::rcoll`.

Referenced by `save2DXscalar()`.

**5.11.1.6** subroutine `drs_io_DX::saveDXvolume3DVec` (double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *XX*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *YY*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *ZZ*, character(len=\*),intent(in) *filename*)

Writes a volume render of the vector field components.

**XX, YY and ZZ**

into the files with basename

**filename.**

References `drs_legendre::costheta`, `drs_dims::m0`, `drs_dims::Nt`, `drs_legendre::pi`, and `drs_radial::rcoll`.

Referenced by `save2DXvector()`.



**5.11.1.7 subroutine drs\_io\_DX::saveDXvolume\_v2 (double precision,dimension(0:(blk\_t-size(mpi\_rank)),intent(in) *field*, character(len=\*),intent(in) *filename*)**

Writes a volume render of the field.

**field**

into the files with basename

**filename.**

References drs\_legendre::costheta, drs\_dims::m0, drs\_dims::Nt, drs\_legendre::pi, and drs\_radial::rcoll.

**5.11.2 Variable Documentation****5.11.2.1 double precision drs\_io\_DX::cut\_phi**

the azimuth to use on meridional cuts

Referenced by save2DXscalar(), save2DXvector(), saveDXmeridional(), and saveDXmeridional3DVec().

**5.11.2.2 integer drs\_io\_DX::cut\_type**

the type of cut or render to save

Referenced by parse\_drs2dx(), save2DXscalar(), and save2DXvector().

**5.11.2.3 double precision drs\_io\_DX::cut\_z**

the azimuth to use on equator parallel cuts

**5.11.2.4 double precision drs\_io\_DX::where\_to\_cut = 0.0d0**

Referenced by parse\_drs2dx(), save2DXscalar(), and save2DXvector().

## 5.12 drs\_io\_par Namespace Reference

Module to read and write parameter and configuration files.

### Functions

- subroutine **drs\_read\_conf\_v2** (io\_calc\_file\_in, io\_calc\_file\_out, comment, error)
- subroutine **drs\_read\_conf** (io\_calc\_file\_in, io\_calc\_file\_out, comment, error)  
*reads parameters for the calculation from the standard input*
- subroutine **read\_input\_par** (unit\_in)  
*reads the parameterfile 'file'.par*
- subroutine **write\_parp** (unit\_out)  
*writes the parameter file 'file'.par*

### Variables

- integer, dimension(8), target **usr\_dimsi**
- integer **lformi**
- integer **drs\_calc\_typei**
- integer **tempBCi**
- integer **flowBCi**
- integer **magBCi**
- integer **Nri**
- integer **Nti**
- integer **Npi**
- integer **Nri\_s**
- integer **Nti\_s**
- integer **Npi\_s**
- integer **lsymmi**
- integer **m0i**
- double precision **etai**
- double precision **Pti**
- double precision **Tai**
- double precision **Ra\_ti**
- double precision **Pmi**
- double precision **hi**
- double precision **drifti**
- double precision **noise**
- integer **stepmaxi**
- integer **sampling\_ratei**
- integer **transienti**
- character(len=60) **commenti**
- integer **magici**
- integer, parameter **magic** = 10205
- integer, parameter **MAGICC1** = 10101
- integer, parameter **MAGICC2** = 10102

- integer, parameter **MAGICC3** = 10103
- integer, parameter **MAGICC4** = 10104
- integer, parameter **MAGICC5** = 10105
- integer, parameter **MAGICC6** = 10106
- integer, parameter **MAGICC7** = 10107
- integer, parameter **MAGICC9** = 10109

### 5.12.1 Detailed Description

Module to read and write parameter and configuration files.

### 5.12.2 Function Documentation

**5.12.2.1** subroutine `drs_io_par::drs_read_conf` (`character(len=60),intent(out) io_calc_file_in`, `character(len=60),intent(out) io_calc_file_out`, `character(len=60),intent(out) comment`, `integer,intent(out) error`)

reads parameters for the calculation from the standard input

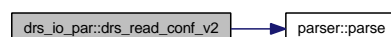
References `drs_time::cpu_max_time`, `drs_time::h`, `drs_dims::lsymm`, `drs_dims::m0`, `magic`, `MAGICC5`, `MAGICC9`, `noise`, `drs_dims::Np`, `drs_dims::Np_s`, `drs_dims::Nr`, `drs_dims::Nr_s`, `drs_dims::Nt`, `drs_dims::Nt_s`, `drs_time::sampling_rate`, `drs_time::stepmax`, and `drs_time::transient`.

Referenced by `drs_init()`.

**5.12.2.2** subroutine `drs_io_par::drs_read_conf_v2` (`character(len=60),intent(out) io_calc_file_in`, `character(len=60),intent(out) io_calc_file_out`, `character(len=60),intent(out) comment`, `integer,intent(out) error`)

References `drs_time::cpu_max_time`, `drs_time::h`, `drs_dims::lsymm`, `drs_dims::m0`, `drs_dims::Np`, `drs_dims::Np_s`, `drs_dims::Nr`, `drs_dims::Nr_s`, `drs_dims::Nt`, `drs_dims::Nt_s`, `parser::parse()`, `drs_time::sampling_rate`, `drs_time::stepmax`, and `drs_time::transient`.

Here is the call graph for this function:



**5.12.2.3** subroutine `drs_io_par::read_input_par` (`integer,intent(in) unit_in`)

reads the parameterfile 'file'.par

#### Todo

Restore read states with other magic numbers.

References `commenti`, `drifti`, `drs_calc_typei`, `etai`, `flowBCi`, `hi`, `lformi`, `lsymmi`, `m0i`, `magBCi`, `MAGICC4`, `MAGICC5`, `MAGICC7`, `MAGICC9`, `magici`, `Npi`, `Npi_s`, `Nri`, `Nri_s`, `Nti`, `Nti_s`, `Pmi`, `Pti`, `Ra_ti`, `sampling_ratei`, `stepmaxi`, `drs_time::stepstart`, `Tai`, `tempBCi`, `drs_time::time`, and `transienti`.

Referenced by `drs_io::drs_load_state()`, and `init()`.

#### 5.12.2.4 subroutine `drs_io_par::write_parp (integer,intent(in) unit_out)`

writes the parameter file 'file'.par

References `drs_time::drift`, `drs_time::h`, `drs_dims::lsymm`, `drs_dims::m0`, `drs_dims::Np`, `drs_dims::Np_s`, `drs_dims::Nr`, `drs_dims::Nr_s`, `drs_dims::Nt`, `drs_dims::Nt_s`, `drs_time::sampling_rate`, `drs_time::stepmax`, `drs_time::steps`, `drs_time::time`, and `drs_time::transient`.

Referenced by `drs_io::save_state()`.

### 5.12.3 Variable Documentation

#### 5.12.3.1 character(len=60) `drs_io_par::commenti`

Referenced by `init()`, and `read_input_par()`.

#### 5.12.3.2 double precision `drs_io_par::drifti`

Referenced by `init()`, and `read_input_par()`.

#### 5.12.3.3 integer `drs_io_par::drs_calc_typei`

Referenced by `init()`, and `read_input_par()`.

#### 5.12.3.4 double precision `drs_io_par::etai`

Referenced by `drs_io::drs_load_state()`, `init()`, and `read_input_par()`.

#### 5.12.3.5 integer `drs_io_par::flowBCi`

Referenced by `read_input_par()`.

#### 5.12.3.6 double precision `drs_io_par::hi`

Referenced by `drs_init()`, and `read_input_par()`.

#### 5.12.3.7 integer `drs_io_par::lformi`

Referenced by `drs_io::drs_load_state()`, `drs_io::drs_open_output()`, `init()`, and `read_input_par()`.

#### 5.12.3.8 integer `drs_io_par::lsymmi`

Referenced by `drs_io::drs_load_state()`, and `read_input_par()`.

#### 5.12.3.9 integer `drs_io_par::m0i`

Referenced by `drs_io::drs_load_state()`, `drs_io::drs_open_output()`, `init()`, and `read_input_par()`.

**5.12.3.10 integer drs\_io\_par::magBCi**

Referenced by read\_input\_par().

**5.12.3.11 integer,parameter drs\_io\_par::magic = 10205**

Referenced by drs\_read\_conf().

**5.12.3.12 integer,parameter drs\_io\_par::MAGICC1 = 10101****5.12.3.13 integer,parameter drs\_io\_par::MAGICC2 = 10102****5.12.3.14 integer,parameter drs\_io\_par::MAGICC3 = 10103****5.12.3.15 integer,parameter drs\_io\_par::MAGICC4 = 10104**

Referenced by read\_input\_par().

**5.12.3.16 integer,parameter drs\_io\_par::MAGICC5 = 10105**

Referenced by drs\_read\_conf(), and read\_input\_par().

**5.12.3.17 integer,parameter drs\_io\_par::MAGICC6 = 10106****5.12.3.18 integer,parameter drs\_io\_par::MAGICC7 = 10107**

Referenced by read\_input\_par().

**5.12.3.19 integer,parameter drs\_io\_par::MAGICC9 = 10109**

Referenced by drs\_read\_conf(), and read\_input\_par().

**5.12.3.20 integer drs\_io\_par::magici**

Referenced by drs\_io::drs\_open\_output(), and read\_input\_par().

**5.12.3.21 double precision drs\_io\_par::noise**

Referenced by drs\_read\_conf().

**5.12.3.22 integer drs\_io\_par::Npi**

Referenced by drs\_io::drs\_load\_state(), init(), and read\_input\_par().

**5.12.3.23 integer drs\_io\_par::Npi\_s**

Referenced by drs\_io::drs\_load\_state(), drs\_io::drs\_open\_output(), init(), and read\_input\_par().

**5.12.3.24 integer drs\_io\_par::Nri**

Referenced by drs\_io::drs\_load\_state(), init(), and read\_input\_par().

**5.12.3.25 integer drs\_io\_par::Nri\_s**

Referenced by drs\_io::drs\_load\_state(), init(), and read\_input\_par().

**5.12.3.26 integer drs\_io\_par::Nti**

Referenced by drs\_io::drs\_load\_state(), init(), and read\_input\_par().

**5.12.3.27 integer drs\_io\_par::Nti\_s**

Referenced by drs\_io::drs\_load\_state(), drs\_io::drs\_open\_output(), init(), and read\_input\_par().

**5.12.3.28 double precision drs\_io\_par::Pmi**

Referenced by drs\_io::drs\_load\_state(), init(), and read\_input\_par().

**5.12.3.29 double precision drs\_io\_par::Pti**

Referenced by drs\_io::drs\_load\_state(), init(), and read\_input\_par().

**5.12.3.30 double precision drs\_io\_par::Ra\_ti**

Referenced by drs\_io::drs\_load\_state(), init(), and read\_input\_par().

**5.12.3.31 integer drs\_io\_par::sampling\_ratei**

Referenced by read\_input\_par().

**5.12.3.32 integer drs\_io\_par::stepmaxi**

Referenced by read\_input\_par().

**5.12.3.33 double precision drs\_io\_par::Tai**

Referenced by drs\_io::drs\_load\_state(), init(), and read\_input\_par().

**5.12.3.34 integer drs\_io\_par::tempBCi**

Referenced by read\_input\_par().

**5.12.3.35 integer drs\_io\_par::transienti**

Referenced by read\_input\_par().

**5.12.3.36 integer,dimension(8),target drs\_io\_par::usr\_dimsi**

Referenced by drs\_io::drs\_load\_state().

## 5.13 drs\_io\_units Namespace Reference

Manages the I/O units of DRS.

### Variables

- integer, parameter **unit\_ek** = 11
- integer, parameter **unit\_ur** = 12
- integer, parameter **unit\_uzon** = 13
- integer, parameter **unit\_koeu** = 14
- integer, parameter **unit\_uaz** = 15
- integer, parameter **unit\_u\_mid** = 16
- integer, parameter **unit\_am** = 17
- integer, parameter **unit\_nu** = 21
- integer, parameter **unit\_adv** = 22
- integer, parameter **unit\_t** = 23
- integer, parameter **unit\_eb** = 31
- integer, parameter **unit\_koeb** = 32
- integer, parameter **unit\_dissu** = 33
- integer, parameter **unit\_dissB** = 34
- integer, parameter **unit\_mspect** = 41
- integer, parameter **unit\_lspect** = 42
- integer, parameter **unit\_nspect** = 43
- integer, parameter **unit\_evp** = 51
- integer, parameter **unit\_evt** = 52
- integer, parameter **unit\_cfl** = 99

### 5.13.1 Detailed Description

Manages the I/O units of DRS.

### 5.13.2 Variable Documentation

**5.13.2.1 integer,parameter drs\_io\_units::unit\_adv = 22**

**5.13.2.2 integer,parameter drs\_io\_units::unit\_am = 17**

Referenced by drs\_io::drs\_open\_output(), and drs\_probes::save\_angular\_momentum().

**5.13.2.3 integer,parameter drs\_io\_units::unit\_cfl = 99**

Referenced by drs\_io::drs\_open\_output(), and drs\_probes::measure().

**5.13.2.4 integer,parameter drs\_io\_units::unit\_dissB = 34**

Referenced by drs\_io::drs\_open\_output(), and drs\_probes::save\_magnetic\_dissipation().



**5.13.2.5 integer,parameter drs\_io\_units::unit\_dissu = 33**

Referenced by drs\_io::drs\_open\_output(), and drs\_probes::save\_flow\_dissipation().

**5.13.2.6 integer,parameter drs\_io\_units::unit\_eb = 31**

Referenced by drs\_io::drs\_open\_output(), and drs\_probes::measure().

**5.13.2.7 integer,parameter drs\_io\_units::unit\_ek = 11**

Referenced by drs\_io::drs\_open\_output(), and drs\_probes::measure().

**5.13.2.8 integer,parameter drs\_io\_units::unit\_evp = 51**

Referenced by drs\_io::drs\_open\_output().

**5.13.2.9 integer,parameter drs\_io\_units::unit\_evt = 52**

Referenced by drs\_io::drs\_open\_output().

**5.13.2.10 integer,parameter drs\_io\_units::unit\_koeb = 32**

Referenced by drs\_io::drs\_open\_output(), and drs\_probes::save\_field\_coeffs().

**5.13.2.11 integer,parameter drs\_io\_units::unit\_koeu = 14**

Referenced by drs\_io::drs\_open\_output(), and drs\_probes::save\_flow\_coeffs().

**5.13.2.12 integer,parameter drs\_io\_units::unit\_lspec = 42**

Referenced by drs\_io::save\_l\_spec().

**5.13.2.13 integer,parameter drs\_io\_units::unit\_mspec = 41**

Referenced by drs\_io::save\_m\_spec().

**5.13.2.14 integer,parameter drs\_io\_units::unit\_nspec = 43**

Referenced by drs\_io::save\_n\_spec().

**5.13.2.15 integer,parameter drs\_io\_units::unit\_nu = 21**

Referenced by drs\_io::drs\_open\_output(), and drs\_probes::measure().

**5.13.2.16** integer,parameter drs\_io\_units::unit\_t = 23

**5.13.2.17** integer,parameter drs\_io\_units::unit\_u\_mid = 16

Referenced by drs\_io::drs\_open\_output(), and drs\_probes::measure().

**5.13.2.18** integer,parameter drs\_io\_units::unit\_uaz = 15

**5.13.2.19** integer,parameter drs\_io\_units::unit\_ur = 12

**5.13.2.20** integer,parameter drs\_io\_units::unit\_uzon = 13

## 5.14 drs\_legendre Namespace Reference

### Classes

- interface **interface**

### Functions

- subroutine **drs\_legendre\_allocation** ()
- subroutine **drs\_legendre\_init** ()  
*Initialize the Legendre associated Polynomials and the Gauss-Legendre co-location points.*
- subroutine **initNormalization** (normType, lmax, norms)  
*Computes the normalization factors for the the Legendre associated Polynomials.*
- subroutine **legendre\_init\_new** ()  
*Initializes the tables of Associated Legendre Polynomials.*
- subroutine **gauleg** (x1, x2, x, w, n)  
*Computes the Gauss-Legendre quadrature points and weights.*

### Variables

- double precision, dimension(:, :, :), allocatable **legendre**  
*The unnormalised Legendre polynomials.*
- double precision, dimension(:, :, :), allocatable **leg\_neg**  
*The unnormalised Legendre polynomials for negative m multiplied by the integration factors.*
- double precision, dimension(:, :, :), allocatable **dleg**  
 *$d P_l m(\cos(\theta)) / d \theta$*
- double precision, dimension(:, :, :), allocatable **leg\_sin**  
 *$P_l m(\sin(\theta))$ .*
- double precision, dimension(:, :), allocatable **plmfac**  
 *$\sqrt{(l+m)!/(l-m)!/(2l+1)} = \sqrt{(l-m+1)*(l-m+2)*...*((l+m)/(2l+1))}$*
- double precision, dimension(:), allocatable, target **costheta**  
*Gauss-Legendre integration points.*
- double precision, dimension(:), allocatable, target **sintheta**  
*Gauss-Legendre integration points.*
- double precision, dimension(:), allocatable **w**  
*Gauss-Legendre integration weights.*
- integer, dimension(:), allocatable **llp1**

Table of  $l(l+1)$ .

- double precision, parameter **pi** = 3.141592653589793d0

### 5.14.1 Function Documentation

#### 5.14.1.1 subroutine drs\_legendre::drs\_legendre\_allocation ()

References drs\_mpi::blk\_ps\_size, costheta, dleg, leg\_neg, leg\_sin, legendre, llp1, drs\_mpi::mpi\_rank, drs\_dims::Nt, drs\_dims::Nt\_s, plmfac, sintheta, and w.

Referenced by drs\_init(), init(), and test\_saveDXMer().

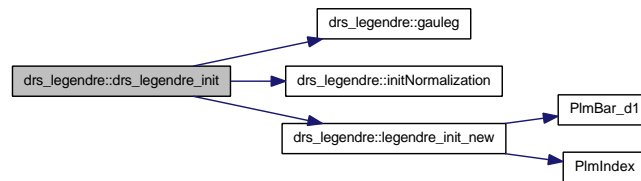
#### 5.14.1.2 subroutine drs\_legendre::drs\_legendre\_init ()

Initialize the Legendre associated Polynomials and the Gauss-Legendre co-location points.

References costheta, gauleg(), initNormalization(), legendre\_init\_new(), llp1, drs\_dims::Nt, drs\_dims::Nt\_s, plmfac, sintheta, and w.

Referenced by drs\_init(), init(), and test\_saveDXMer().

Here is the call graph for this function:



#### 5.14.1.3 subroutine drs\_legendre::gauleg (double precision,intent(in) x1, double precision,intent(in) x2, double precision,dimension(n),intent(out) x, double precision,dimension(n),intent(out) w, integer,intent(in) n)

Computes the Gauss-Legendre quadrature points and weights.

References pi.

Referenced by drs\_legendre\_init(), and CrankNicholson::updateCrankNicholson\_matrices().

#### 5.14.1.4 subroutine drs\_legendre::initNormalization (integer,intent(in) normType, integer,intent(in) lmax, double precision,dimension(0:lmax+2, 0:lmax+2),intent(out) norms)

Computes the normalization factors for the the Legendre associated Polynomials.

##### Parameters:

**normType** normalization types:

- Normalised for normalization to 2.
- UnNormalized for no normalization, that is,  $\int_{-1}^1 (P_l^m)^2 dx = 2 \cdot \frac{(l+m)!}{(l-m)!} \cdot \frac{1}{(2l+1)}$ .

**lmax** normalization types:

- Normalised for normalization to 2.
- UnNormalized for no normalization, that is,  $\sqrt{(P_l^m)^2} = 2 \cdot \{(l+m)!\} \{(l-m)!(2l+1)\} \sqrt{}$ .

Referenced by `drs_legendre_init()`, and `drs_io::drs_open_output()`.

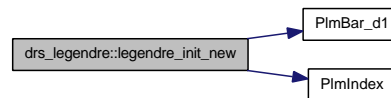
#### 5.14.1.5 subroutine `drs_legendre::legendre_init_new ()`

Initializes the tables of Associated Legendre Polynomials.

References `drs_mpi::blk_ps_size`, `drs_mpi::blk_ts_start`, `costheta`, `dleg`, `leg_neg`, `leg_sin`, `legendre`, `drs_mpi::mpi_rank`, `drs_dims::Nt`, `PlmBar_d1()`, `plmfac`, `PlmIndex()`, `sintheta`, and `w`.

Referenced by `drs_legendre_init()`.

Here is the call graph for this function:



## 5.14.2 Variable Documentation

### 5.14.2.1 double precision,dimension(:),allocatable,target `drs_legendre::costheta`

Gauss-Legendre integration points.

Referenced by `Benchmarkv1()`, `computeEMF()`, `drs_legendre_allocation()`, `drs_legendre_init()`, `legendre_init_new()`, `drs_renderers::render_Bz()`, `drs_renderers::render_rotu_z()`, `drs_renderers::render_streamlines_t()`, `drs_renderers::render_uz()`, `drs_io_DX::saveDXmeridional()`, `saveDXmeridional()`, `drs_io_DX::saveDXmeridional3DVec()`, `drs_io_DX::saveDXvolume()`, `drs_io_DX::saveDXvolume3DVec()`, `drs_io_DX::saveDXvolume_v2()`, `saveIDLmeridional()`, `selectEquatorMidShell()`, `StateAverage()`, and `YokoiPlots()`.

### 5.14.2.2 double precision,dimension (:,:,),allocatable `drs_legendre::dleg`

$d \text{Plm}(\cos(\theta))/d \theta$

Referenced by `drs_field::calc_field()`, `drs_flow::calc_flow()`, `drs_field::calc_rot_field()`, `drs_flow::calc_rot_flow()`, `drs_legendre_allocation()`, `legendre_init_new()`, and `drs_renderers::render_streamlines_t()`.

### 5.14.2.3 double precision,dimension (:,:,),allocatable `drs_legendre::leg_neg`

The unnormalised Legendre polynomials for negative m multiplied by the integration factors.

Referenced by `drs_legendre_allocation()`, `legendre_init_new()`, `drs_transforms::ylmt()`, and `drs_transforms::ylmt_3D()`.

### 5.14.2.4 double precision,dimension (:,:,),allocatable `drs_legendre::leg_sin`

$\text{Plm}/\sin(\theta)$ .

Referenced by `drs_field::calc_field()`, `drs_flow::calc_flow()`, `drs_field::calc_rot_field()`, `drs_flow::calc_rot_flow()`, `drs_legendre_allocation()`, and `legendre_init_new()`.

#### 5.14.2.5 double precision,dimension(:,,:),allocatable drs\_legendre::legendre

The unnormalised Legendre polynomials.

Referenced by `drs_field::calc_field()`, `drs_flow::calc_flow()`, `drs_field::calc_rot_field()`, `drs_flow::calc_rot_flow()`, `drs_legendre_allocation()`, `legendre_init_new()`, and `drs_transforms::ylmb()`.

#### 5.14.2.6 integer,dimension(:),allocatable drs\_legendre::llp1

Table of  $l(l+1)$ .

Referenced by `drs_field::calc_field()`, `drs_field::calc_field_lspec()`, `drs_field::calc_field_mspec()`, `drs_field::calc_field_nspec()`, `drs_flow::calc_flow()`, `drs_flow::calc_flow_lspec()`, `drs_flow::calc_flow_mspec()`, `drs_flow::calc_flow_nspec()`, `drs_field::calc_rot_field()`, `drs_flow::calc_rot_flow()`, `drs_comp::drs_comp_reset()`, `drs_legendre_allocation()`, `drs_legendre_init()`, `kd_grothrate()`, `drs_transforms::my_rotrot()`, `drs_transforms::PolTor_common2PolTor_field()`, `drs_transforms::PolTor_common2PolTor_flow()`, `drs_field::update_field_pol_lap()`, `drs_field::update_field_tor_lap()`, `drs_flow::update_flow_pol_lap()`, `drs_flow::update_flow_tor_lap()`, `drs_temp::update_temp_lap()`, and `CrankNicholson::updateCrankNicholson_matrices()`.

#### 5.14.2.7 double precision,parameter drs\_legendre::pi = 3.141592653589793d0

Referenced by `Benchmarkv1()`, `Benchmarkv2()`, `drs_field::drs_field_random_init()`, `drs_init()`, `drs_temp::drs_temp_randomize()`, `gauleg()`, `drs_renderers::render_Bz()`, `drs_renderers::render_rotu_z()`, `drs_renderers::render_uz()`, `drs_io_DX::saveDXmeridional()`, `saveDXmeridional()`, `drs_io_DX::saveDXmeridional3DVec()`, `drs_io_DX::saveDXvolume()`, `drs_io_DX::saveDXvolume3DVec()`, `drs_io_DX::saveDXvolume_v2()`, `saveIDLmeridional()`, `test_saveDXMer()`, and `volume()`.

#### 5.14.2.8 double precision,dimension (:,:),allocatable drs\_legendre::plmfac

$$\text{sqrt}((l+m)!/(l-m)!/(2l+1)) = \text{sqrt}((l-m+1)*(l-m+2)*\dots*(l+m)/(2l+1))$$

Referenced by `drs_field::calc_field_lspec()`, `drs_field::calc_field_mspec()`, `drs_field::calc_field_nspec()`, `drs_flow::calc_flow_lspec()`, `drs_flow::calc_flow_mspec()`, `drs_flow::calc_flow_nspec()`, `drs_field::drs_field_random_init()`, `drs_legendre_allocation()`, `drs_legendre_init()`, and `legendre_init_new()`.

#### 5.14.2.9 double precision,dimension(:),allocatable,target drs\_legendre::sintheta

Gauss-Legendre integration points.

Referenced by `Benchmarkv1()`, `drs_legendre_allocation()`, `drs_legendre_init()`, `legendre_init_new()`, `drs_transforms::vectorField2Divergence()`, and `drs_transforms::vectorField2PolTor_common()`.

#### 5.14.2.10 double precision,dimension(:),allocatable drs\_legendre::w

Gauss-Legendre integration weights.

Referenced by `drs_legendre_allocation()`, `drs_legendre_init()`, and `legendre_init_new()`.

## 5.15 drs\_lock Namespace Reference

This module provides a locking mechanism for the dynamo code.

### Functions

- subroutine **drs\_lock\_init** (*u*, *f*)  
*Sets the lock file name to f and manages it on unit u.*
- subroutine **add\_lock** (*error*)  
*Creates the lock file.*
- subroutine **rm\_lock** (*error*)  
*Removes the lock file.*
- logical **lockExists** ()  
*Checks whether the lock file exists.*

### Variables

- character(len=128) **lockFileName**
- integer **lockFileUnit** = -1

#### 5.15.1 Detailed Description

This module provides a locking mechanism for the dynamo code.

Since:

1.6.1

#### 5.15.2 Function Documentation

##### 5.15.2.1 subroutine drs\_lock::add\_lock (integer,intent(inout) *error*)

Creates the lock file.

References lockFileName, and lockFileUnit.

Referenced by drs\_init().

##### 5.15.2.2 subroutine drs\_lock::drs\_lock\_init (integer *u*, character(len=\*)*f*)

Sets the lock file name to *f* and manages it on unit *u*.

**Parameters:**

- u* The unit it is going to be opened on.
- f* The name of the lock file.

References lockFileName, and lockFileUnit.

Referenced by drs\_init().

#### **5.15.2.3 logical drs\_lock::lockExists ()**

Checks whether the lock file exists.

References lockFileName.

Referenced by need\_to\_step().

#### **5.15.2.4 subroutine drs\_lock::rm\_lock (integer,intent(inout) error)**

Removes the lock file.

References lockFileName, and lockFileUnit.

Referenced by drs().

### **5.15.3 Variable Documentation**

#### **5.15.3.1 character(len=128) drs\_lock::lockFileName**

Referenced by add\_lock(), drs\_lock\_init(), lockExists(), and rm\_lock().

#### **5.15.3.2 integer drs\_lock::lockFileUnit = -1**

Referenced by add\_lock(), drs\_lock\_init(), and rm\_lock().



## 5.16 drs\_mpi Namespace Reference

Provides initialisation and variables to be used with the mpi implementation.

### Classes

- interface **sum\_over\_all\_cpus**  
*Encapsulates sums of several types and ranks.*
- interface **drs\_minimize**  
*Encapsulates minimization of several types and ranks.*
- interface **drs\_maximize**  
*Encapsulates maximization of several types and ranks.*
- interface **drs\_bcast**  
*Encapsulates broadcast of several types and ranks.*

### Functions

- subroutine **drs\_mpi\_init** ()  
*Gets initial values for mpi\_size and mpi\_rank. Allocates block indices accordingly.*
- subroutine **mpi\_dims\_init** (Nt, Np\_s, m0, error)  
*Initializes mpi variables and sizes.*
- subroutine **transpos\_phi2theta** (input, Nt, output, Np)  
*transposition: t distrib(phi) --> tt\_t distrib(theta):*
- subroutine **transpos\_theta2phi** (input, Np\_s, output, Nt)  
*transposition: tt\_t distrib(theta) --> t distrib(phi):*
- subroutine **distribute\_in\_m** (buffer, Nt, Nr)  
*Performs a one-to-all communication of the contents of buffer. It is essentially a targeted version of mpi\_scatter.*
- subroutine **gather\_from\_m** (buffer, Nt, Nr)  
*Performs an all-to-one communication of the contents of buffer. It is essentially a targeted version of mpi\_gather.*
- subroutine **blk\_ts\_start\_init** (m0)  
*Initialises blk\_ts\_start.*
- subroutine **sum\_over\_all\_cpus\_scal** (val)  
*Subroutine to encapsulate sums across all the cpu's.*  
.
- subroutine **sum\_over\_all\_cpus\_vect** (val)

*Subroutine to encapsulate mpi calls that sum arrays over all cpu's.*

- subroutine **wait\_for\_everyone** ()  
*Encapsulate mpi barrier.*
- subroutine **drs\_minimize\_dble** (array)  
*Encapsulate mpi\_reduce min.*
- subroutine **drs\_minimize\_dble\_scal** (val)  
*Encapsulate mpi\_reduce min (scalars).*
- subroutine **drs\_maximize\_dble** (array)  
*Encapsulate mpi\_reduce max.*
- subroutine **drs\_maximize\_dble\_scal** (val)  
*Encapsulate mpi\_reduce max (scalars).*
- subroutine **drs\_gather\_vars** (rank, val)
- subroutine **drs\_bcast\_dble** (array, num)
- subroutine **drs\_bcast\_int** (array, num)
- subroutine **drs\_bcast\_dble\_scal** (val)
- subroutine **drs\_bcast\_int\_scal** (val)
- subroutine **drs\_bcast\_logical\_scal** (val)
- subroutine **drs\_abort** (error)
- subroutine **mpi\_cleanup** ()

## Variables

- integer **mpi\_size**  
*How many CPU's are in use.*
- integer **mpi\_rank**  
*The rank of the present CPU.*
- integer, dimension(:), allocatable, target **blk\_ps\_start**  
*Start index of the blocks in m for each CPU.*
- integer, dimension(:), allocatable, target **blk\_ps\_size**  
*Size of the blocks in m for each CPU.*
- integer, dimension(:), allocatable, target **blk\_t\_start**  
*Start index of the blocks in theta dor each CPU.*
- integer, dimension(:), allocatable, target **blk\_t\_size**  
*Size of the blocks in theta for each CPU.*
- integer, dimension(:), allocatable, target **blk\_ts\_start**  
*Stores the index of the first nonzero l value in the block.*
- integer, dimension(:), pointer **mm**

*A convinience shorthand for blk\_ts\_start.*

- integer **blk\_t\_max\_size**

*Maximum size of theta block per cpu.*

- integer **blk\_ps\_max\_size**

*Maximum size of phi block per cpu.*

## 5.16.1 Detailed Description

Provides initialisation and variables to be used with the mpi implementation.

## 5.16.2 Function Documentation

### 5.16.2.1 subroutine drs\_mpi::blk\_ts\_start\_init (integer,intent(in) *m0*)

Initialises blk\_ts\_start.

References blk\_ps\_max\_size, blk\_ps\_start, blk\_ts\_start, mm, and mpi\_rank.

Referenced by mpi\_dims\_init().

### 5.16.2.2 subroutine drs\_mpi::distribute\_in\_m (double precision,dimension(:,,:),intent(inout) *buffer*, integer,intent(in) *Nt*, integer,intent(in) *Nr*)

Performs a one-to-all communication of the contents of buffer. It is essentially a targeted version of mpi\_scatter.

References blk\_ps\_size, blk\_ps\_start, mpi\_rank, and mpi\_size.

Referenced by drs\_io::drs\_open\_output().

### 5.16.2.3 subroutine drs\_mpi::drs\_abort (integer,intent(in) *error*)

References mpi\_rank.

Referenced by Benchmarkv1(), computeAndSaveAverage(), drs(), drs\_init(), drs\_mpi\_init(), getProfile(), init(), drs\_nonlinear::rhs(), StateAverage(), test\_saveDXMer(), and YokoiPlots().

### 5.16.2.4 subroutine drs\_mpi::drs\_bcast\_dble (double precision,dimension(:),intent(inout) *array*, integer,intent(in) *num*)

References mpi\_size.

**5.16.2.5** subroutine `drs_mpi::drs_bcast_dble_scal` (double precision,intent(inout) *val*)

**5.16.2.6** subroutine `drs_mpi::drs_bcast_int` (integer,dimension(:),intent(inout) *array*, integer,intent(in) *num*)

**5.16.2.7** subroutine `drs_mpi::drs_bcast_int_scal` (integer,intent(inout) *val*)

**5.16.2.8** subroutine `drs_mpi::drs_bcast_logical_scal` (logical,intent(inout) *val*)

**5.16.2.9** subroutine `drs_mpi::drs_gather_vars` (integer,dimension(:),intent(in) *rank*, double precision,dimension(:),intent(inout) *val*)

References `mpi_rank`, and `mpi_size`.

Referenced by `drs_probes::save_field_coeffs()`, and `drs_probes::save_flow_coeffs()`.

**5.16.2.10** subroutine `drs_mpi::drs_maximize_dble` (double precision,dimension(:),intent(inout) *array*)

Encapsulate `mpi_reduce max`.

References `mpi_size`.

**5.16.2.11** subroutine `drs_mpi::drs_maximize_dble_scal` (double precision,intent(inout) *val*)

Encapsulate `mpi_reduce max (scalars)`.

References `mpi_size`.

**5.16.2.12** subroutine `drs_mpi::drs_minimize_dble` (double precision,dimension(:),intent(inout) *array*)

Encapsulate `mpi_reduce min`.

References `mpi_size`.

**5.16.2.13** subroutine `drs_mpi::drs_minimize_dble_scal` (double precision,intent(inout) *val*)

Encapsulate `mpi_reduce min (scalars)`.

References `mpi_size`.

**5.16.2.14** subroutine `drs_mpi::drs_mpi_init` ()

Gets initial values for `mpi_size` and `mpi_rank`. Allocates block indices accordingly.

References `blk_ps_size`, `blk_ps_start`, `blk_t_size`, `blk_t_start`, `drs_abort()`, `mpi_rank`, and `mpi_size`.

Referenced by `drs_init()`, `init()`, and `test_saveDXMer()`.

Here is the call graph for this function:



### 5.16.2.15 subroutine drs\_mpi::gather\_from\_m (double precision,dimension(:,,:),intent(inout) *buffer*, integer,intent(in) *Nt*, integer,intent(in) *Nr*)

Performs an all-to-one communication of the contents of *buffer*. It is essentially a targeted version of `mpi_gather`.

References `blk_ps_size`, `blk_ps_start`, `mpi_rank`, and `mpi_size`.

Referenced by `drs_io::drs_open_output()`.

### 5.16.2.16 subroutine drs\_mpi::mpi\_cleanup ()

Referenced by `drs()`.

### 5.16.2.17 subroutine drs\_mpi::mpi\_dims\_init (integer,intent(in) *Nt*, integer,intent(in) *Np\_s*, integer,intent(in) *m0*, integer,intent(out) *error*)

Initializes mpi variables and sizes.

References `blk_ps_max_size`, `blk_ps_size`, `blk_ps_start`, `blk_t_max_size`, `blk_t_size`, `blk_t_start`, `blk_ts_start_init()`, and `mpi_size`.

Referenced by `drs_init()`, `init()`, and `test_saveDXMer()`.

Here is the call graph for this function:



### 5.16.2.18 subroutine drs\_mpi::sum\_over\_all\_cpus\_scal (double precision,intent(inout) *val*)

Subroutine to encapsulate sums across all the cpu's.

.

References `mpi_size`.

### 5.16.2.19 subroutine drs\_mpi::sum\_over\_all\_cpus\_vect (double precision,dimension(:),intent(inout) *val*)

Subroutine to encapsulate mpi calls that sum arrays over all cpu's.

References `mpi_size`.

### 5.16.2.20 subroutine drs\_mpi::transpos\_phi2theta (double precision,dimension(0:nt, 1:blk\_ps\_size(mpi\_rank),intent(in) *input*, integer,intent(in) *Nt*, double precision,dimension(0:blk\_t\_size(mpi\_rank),intent(out) *output*, integer,intent(out) *Np*)

transposition: `t distrib(phi) --> tt_t distrib(theta): ~~~~~~ input: t(0:Nt,Np_s) in (theta,phi) (distr. in phi) blk_ps_size() (p. 72),blk_ps_start() (p. 73) contain the local input dims in phi for all processors. blk_t_size() (p. 73),blk_t_start() (p. 73) contain the local output dims in theta. blk_ps_max_size,blk_t_max_size: maximum blocksizes in phi, theta. mpi_size,mpi_rank ir,tag: radial index,message-tag`

output: `tt_t(0:(Ntl-1),mpi_size*blk_ps_max_size)` (distr. in theta) `tt_t(0:(blk_t_max_size-1),Np)` (dynamic physical dims!) ~~~~~~ 04.10.96 M.A. original version (blocking).

References `blk_ps_start`, `blk_t_start`, and `mpi_size`.

Referenced by `drs_transforms::m2phi_2D()`.

#### 5.16.2.21 subroutine `drs_mpi::transpos_theta2phi` (double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *input*, integer,intent(in) *Np\_s*, double precision,dimension(0:nt,blk\_ps\_size(mpi\_rank),intent(out) *output*, integer,intent(in) *Nt*)

transposition: `tt_t distrib(theta) --> t distrib(phi)`: ~~~~~~ input: `tt_t(0:Ntl,mpi_size*blk_ps_max_size)` (transposed) `tt_t(0:(blk_t_max_size-1),Np)` (dynamic physical dims!)

`blk_ps_size()` (p. 72), `blk_ps_start()` (p. 73) contain the local output dims in phi for all pes. `blk_t_size()` (p. 73), `blk_t_start()` (p. 73) contain the local input dims in theta. `blk_ps_max_size`, `blk_t_max_size`: maximum blocksizes in phi, theta. `mpi_size`, `mpi_rank` ir,tag: radial index,message-tag

output: `t(0:Nt,Np_s)` in (theta,phi) (distr. in phi) ~~~~~~

04.10.96 M.A. original version.

References `blk_ps_start`, `blk_t_start`, and `mpi_size`.

Referenced by `drs_transforms::ylmt()`, and `drs_transforms::ylmt_3D()`.

#### 5.16.2.22 subroutine `drs_mpi::wait_for_everyone` ()

Encapsulate mpi barrier.

References `mpi_size`.

Referenced by `need_to_step()`, `drs_debug::save_lmr_quantity()`, and `drs_debug::save_tpr_quantity()`.

### 5.16.3 Variable Documentation

#### 5.16.3.1 integer `drs_mpi::blk_ps_max_size`

Maximum size of phi block per cpu.

Referenced by `blk_ts_start_init()`, and `mpi_dims_init()`.

#### 5.16.3.2 integer,dimension(:),allocatable,target `drs_mpi::blk_ps_size`

Size of the blocks in m for each CPU.

Referenced by `drs_probes::average_unnormalised_field_l_spectrum()`, `drs_probes::average_unnormalised_flow_l_spectrum()`, `drs_field::calc_field_lspec()`, `drs_field::calc_field_mspec()`, `drs_flow::calc_flow_lspec()`, `drs_flow::calc_flow_mspec()`, `distribute_in_m()`, `drs_comp::drs_comp_allocation()`, `drs_field::drs_field_allocation()`, `drs_flow::drs_flow_allocation()`, `drs_init()`, `drs_legendre::drs_legendre_allocation()`, `drs_mpi_init()`, `drs_nonlinear::drs_nonlinear_init()`, `drs_temp::drs_temp_allocation()`, `drs_io::dump_state()`, `gather_from_m()`, `kd_grothrate()`, `drs_legendre::legendre_init_new()`, `drs_probes::measure_lm()`, `mpi_dims_init()`, `drs_renderers::render_B_outside()`, `drs_probes::save_field_coeffs()`, and `drs_probes::save_flow_coeffs()`.

**5.16.3.3 integer,dimension(:),allocatable,target drs\_mpi::blk\_ps\_start**

Start index of the blocks in m for each CPU.

Referenced by drs\_probes::average\_unnormalised\_field\_l\_spectrum(), drs\_probes::average\_unnormalised\_flow\_l\_spectrum(), drs\_probes::average\_unnormalised\_scalar\_l\_spectrum(), blk\_ts\_start\_init(), drs\_field::calc\_field(), drs\_field::calc\_field\_lspec(), drs\_field::calc\_field\_mspec(), drs\_field::calc\_field\_nspec(), drs\_flow::calc\_flow(), drs\_flow::calc\_flow\_lspec(), drs\_flow::calc\_flow\_mspec(), drs\_flow::calc\_flow\_nspec(), drs\_field::calc\_rot\_field(), drs\_flow::calc\_rot\_flow(), distribute\_in\_m(), drs\_field::drs\_field\_random\_init(), drs\_init(), drs\_mpi\_init(), drs\_io::drs\_open\_output(), drs\_temp::drs\_temp\_randomize(), drs\_io::dump\_state(), gather\_from\_m(), kd\_grothrate(), drs\_probes::l\_spec\_of\_scalar\_field(), drs\_probes::m\_spec\_of\_scalar\_field(), drs\_probes::measure\_lm(), mpi\_dims\_init(), drs\_transforms::my\_div(), drs\_transforms::my\_rot(), drs\_transforms::my\_rotrot(), drs\_probes::n\_spec\_of\_scalar\_field(), drs\_probes::save\_field\_coeffs(), drs\_probes::save\_flow\_coeffs(), drs\_probes::save\_flow\_dissipation(), drs\_debug::save\_lmr\_quantity(), drs\_probes::save\_magnetic\_dissipation(), drs\_nonlinear::save\_stuff(), transpos\_phi2theta(), and transpos\_theta2phi().

**5.16.3.4 integer drs\_mpi::blk\_t\_max\_size**

Maximum size of theta block per cpu.

Referenced by mpi\_dims\_init().

**5.16.3.5 integer,dimension(:),allocatable,target drs\_mpi::blk\_t\_size**

Size of the blocks in theta for each CPU.

Referenced by drs\_init(), drs\_mpi\_init(), drs\_nonlinear::drs\_nonlinear\_init(), drs\_probes::drs\_probes\_allocation(), drs\_probes::drs\_probes\_init(), init(), mpi\_dims\_init(), drs\_nonlinear::save\_stuff(), and test\_saveDXMer().

**5.16.3.6 integer,dimension(:),allocatable,target drs\_mpi::blk\_t\_start**

Start index of the blocks in theta for each CPU.

Referenced by drs\_probes::compute\_helicities(), drs\_mpi\_init(), drs\_probes::drs\_probes\_init(), drs\_io::dump\_state(), drs\_probes::integrate\_power\_surf(), mpi\_dims\_init(), drs\_probes::save\_angular\_momentum(), drs\_nonlinear::save\_stuff(), drs\_debug::save\_tpr\_quantity(), selectEquatorMidShell(), transpos\_phi2theta(), transpos\_theta2phi(), drs\_transforms::vectorField2Divergence(), and drs\_transforms::vectorField2PolTor\_common().

**5.16.3.7 integer,dimension(:),allocatable,target drs\_mpi::blk\_ts\_start**

Stores the index of the first nonzero l value in the block.

Referenced by blk\_ts\_start\_init(), drs\_legendre::legendre\_init\_new(), drs\_transforms::my\_div(), drs\_transforms::my\_rot(), drs\_transforms::my\_rotrot(), drs\_radial::radial\_dr\_ddr\_3D\_n2r(), drs\_radial::radial\_dr\_ddr\_3D\_r2r(), and drs\_nonlinear::save\_stuff().

**5.16.3.8 integer,dimension(:),pointer drs\_mpi::mm**

A convenience shorthand for blk\_ts\_start.

Referenced by `blk_ts_start_init()`, `drs_field::calc_field()`, `drs_flow::calc_flow()`, `drs_field::calc_rot_field()`, `drs_flow::calc_rot_flow()`, `drs()`, `update_field()`, `update_flow()`, `drs_transforms::ylmb()`, `drs_transforms::ylmt()`, and `drs_transforms::ylmt_3D()`.

#### 5.16.3.9 integer `drs_mpi::mpi_rank`

The rank of the present CPU.

Referenced by `drs_probes::average_unnormalised_field_l_spectrum()`, `drs_probes::average_unnormalised_flow_l_spectrum()`, `blk_ts_start_init()`, `drs_field::calc_field_lspec()`, `drs_field::calc_field_mspect()`, `drs_flow::calc_flow_lspec()`, `drs_flow::calc_flow_mspect()`, `distribute_in_m()`, `drs()`, `drs_abort()`, `drs_comp::drs_comp_allocation()`, `drs_field::drs_field_allocation()`, `drs_field::drs_field_random_init()`, `drs_flow::drs_flow_allocation()`, `drs_gather_vars()`, `drs_init()`, `drs_legendre::drs_legendre_allocation()`, `drs_io::drs_load_state()`, `drs_mpi_init()`, `drs_nonlinear::drs_nonlinear_init()`, `drs_io::drs_open_output()`, `drs_probes::drs_probes_allocation()`, `drs_probes::drs_probes_init()`, `drs_temp::drs_temp_allocation()`, `drs_temp::drs_temp_randomize()`, `gather_from_m()`, `init()`, `kd_grothrate()`, `drs_legendre::legendre_init_new()`, `drs_probes::measure_lm()`, `need_to_step()`, `drs_renderers::render_B_outside()`, `drs_nonlinear::rhs()`, `drs_probes::save_field_coeffs()`, `drs_probes::save_flow_coeffs()`, `drs_io::save_l_spec()`, `drs_io::save_m_spec()`, `drs_io::save_n_spec()`, `drs_io::save_state()`, `drs_nonlinear::save_stuff()`, and `test_saveDXMer()`.

#### 5.16.3.10 integer `drs_mpi::mpi_size`

How many CPU's are in use.

Referenced by `distribute_in_m()`, `drs_bcast_dble()`, `drs_gather_vars()`, `drs_maximize_dble()`, `drs_maximize_dble_scal()`, `drs_minimize_dble()`, `drs_minimize_dble_scal()`, `drs_mpi_init()`, `gather_from_m()`, `init()`, `mpi_dims_init()`, `drs_debug::save_lmr_quantity()`, `drs_debug::save_tpr_quantity()`, `sum_over_all_cpus_scal()`, `sum_over_all_cpus_vect()`, `test_saveDXMer()`, `transpos_phi2theta()`, `transpos_theta2phi()`, and `wait_for_everyone()`.



## 5.17 drs\_nonlinear Namespace Reference

Takes care of constructing the nonlinear terms of all equations and other quantities in real space.

### Functions

- subroutine **drs\_nonlinear\_init** ()
- subroutine **evaluate\_real\_space** ()
- subroutine **rhs** (h\_old, h)
- subroutine **save\_stuff** (nsample)

*Encapsulate saving quantities in real and spectral space.*

### Variables

- double precision, allocatable **rhs\_NS\_tor**

*NS for Navier-Stokes.*

- double precision, allocatable **rhs\_NS\_pol**
- double precision, allocatable **rhs\_IE\_tor**

*IE for Induction Equation.*

- double precision, allocatable **rhs\_IE\_pol**
- double precision, allocatable **rhs\_TE**

*TE for Temperature Equation.*

- double precision, dimension(:,:,:), allocatable **temp\_t**
- double precision, dimension(:,:,:), allocatable **flow\_r\_t**

*Quantities in real space.*

- double precision, dimension(:,:,:), allocatable **flow\_t\_t**
- double precision, dimension(:,:,:), allocatable **flow\_p\_t**
- double precision, dimension(:,:,:), allocatable **field\_r\_t**
- double precision, dimension(:,:,:), allocatable **field\_t\_t**
- double precision, dimension(:,:,:), allocatable **field\_p\_t**
- double precision, dimension(:,:,:), allocatable **rot\_flow\_r\_t**
- double precision, dimension(:,:,:), allocatable **rot\_flow\_t\_t**
- double precision, dimension(:,:,:), allocatable **rot\_flow\_p\_t**
- double precision, dimension(:,:,:), allocatable **rot\_field\_r\_t**
- double precision, dimension(:,:,:), allocatable **rot\_field\_t\_t**
- double precision, dimension(:,:,:), allocatable **rot\_field\_p\_t**
- integer, parameter **ncfl** = 5
- double precision, dimension(ncfl) **cfl**

### 5.17.1 Detailed Description

Takes care of constructing the nonlinear terms of all equations and other quantities in real space.

## 5.17.2 Function Documentation

### 5.17.2.1 subroutine drs\_nonlinear::drs\_nonlinear\_init ()

References drs\_mpi::blk\_ps\_size, drs\_mpi::blk\_t\_size, field\_p\_t, field\_r\_t, field\_t\_t, flow\_p\_t, flow\_r\_t, flow\_t\_t, drs\_mpi::mpi\_rank, drs\_dims::Np, drs\_dims::Nr, drs\_dims::Nt\_s, rhs\_IE\_pol, rhs\_IE\_tor, rhs\_NS\_pol, rhs\_NS\_tor, rhs\_TE, rot\_field\_p\_t, rot\_field\_r\_t, rot\_field\_t\_t, rot\_flow\_p\_t, rot\_flow\_r\_t, rot\_flow\_t\_t, and temp\_t.

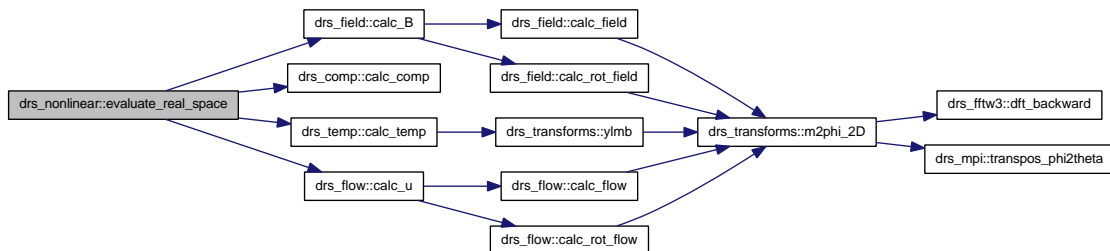
Referenced by drs\_init().

### 5.17.2.2 subroutine drs\_nonlinear::evaluate\_real\_space ()

References drs\_field::calc\_B(), drs\_comp::calc\_comp(), drs\_temp::calc\_temp(), drs\_flow::calc\_u(), drs\_comp::comp, field\_p\_t, field\_r\_t, field\_t\_t, flow\_p\_t, flow\_r\_t, flow\_t\_t, rot\_field\_p\_t, rot\_field\_r\_t, rot\_field\_t\_t, rot\_flow\_p\_t, rot\_flow\_r\_t, rot\_flow\_t\_t, and temp\_t.

Referenced by drs(), and drs\_io::dump\_state().

Here is the call graph for this function:

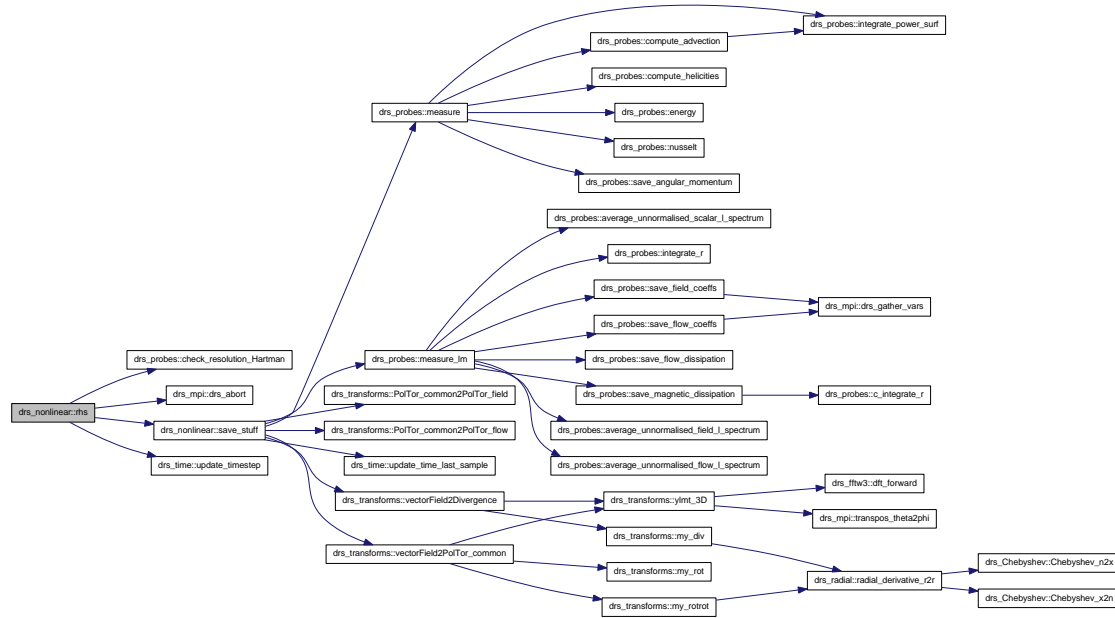


### 5.17.2.3 subroutine drs\_nonlinear::rhs (double precision,intent(out) h\_old, double precision,intent(inout) h)

References cfl, drs\_probes::check\_resolution\_Hartman(), drs\_mpi::drs\_abort(), drs\_mpi::mpi\_rank, drs\_time::nsample, rhs\_IE\_pol, rhs\_IE\_tor, rhs\_NS\_pol, rhs\_NS\_tor, rhs\_TE, drs\_probes::Rm, drs\_time::sampling\_rate, save\_stuff(), drs\_time::steps, drs\_time::transient, and drs\_time::update\_timestep().

Referenced by drs().

Here is the call graph for this function:



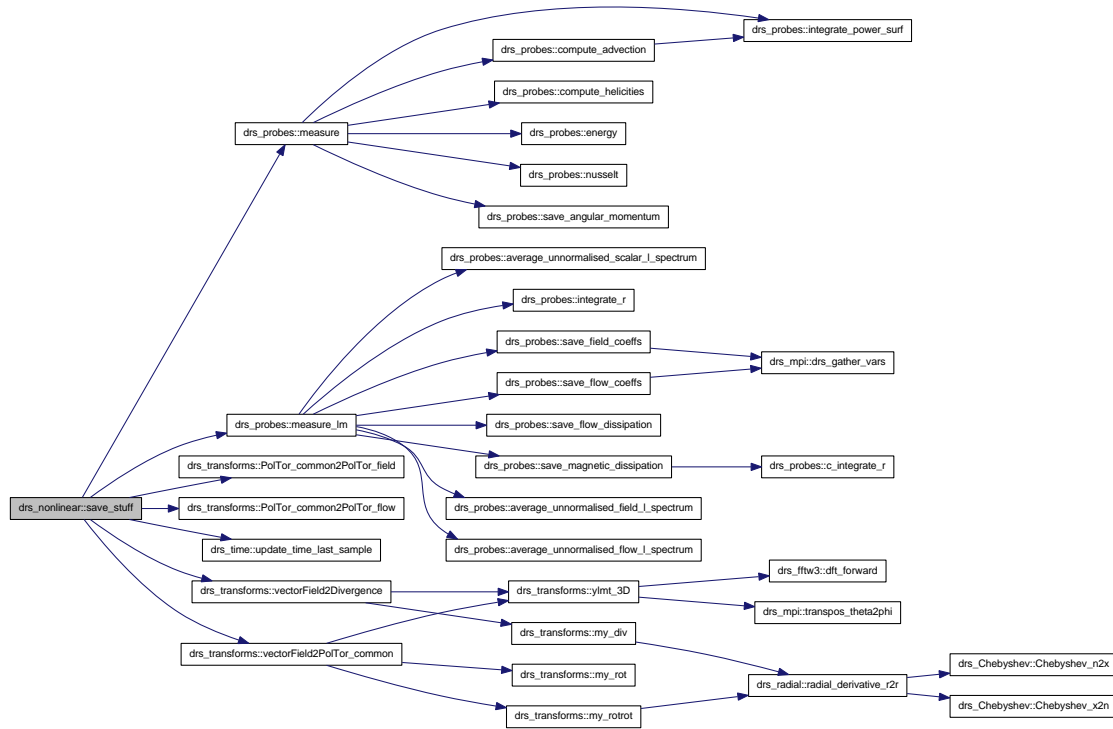
#### 5.17.2.4 subroutine drs\_nonlinear::save\_stuff (integer,intent(inout) nsample)

Encapsulate saving quantities in real and spectral space.

References drs\_mpi::blk\_ps\_start, drs\_mpi::blk\_t\_size, drs\_mpi::blk\_t\_start, drs\_mpi::blk\_ts\_start, cfl, drs\_comp::comp, drs\_comp::comp\_profile\_dr, drs\_radial::drcoll, field\_p\_t, field\_r\_t, field\_t\_t, flow\_p\_t, drs\_flow::flow\_pol, flow\_r\_t, flow\_t\_t, drs\_probes::measure(), drs\_probes::measure\_lm(), drs\_mpi::mpi\_rank, drs\_dims::Np, drs\_dims::Nr, drs\_dims::Nt, drs\_transforms::PolTor\_common2PolTor\_field(), drs\_transforms::PolTor\_common2PolTor\_flow(), drs\_radial::rcoll, rot\_field\_p\_t, rot\_field\_r\_t, rot\_field\_t\_t, rot\_flow\_p\_t, rot\_flow\_r\_t, rot\_flow\_t\_t, drs\_temp::temp, drs\_temp::temp\_profile\_dr, temp\_t, drs\_time::time, drs\_time::time\_last\_sample, drs\_time::time\_since\_last\_sample, drs\_time::update\_time\_last\_sample(), drs\_transforms::vectorField2Divergence(), and drs\_transforms::vectorField2PolTor\_common().

Referenced by drs\_io::dump\_state(), and rhs().

Here is the call graph for this function:



## 5.17.3 Variable Documentation

### 5.17.3.1 `double precision, dimension(ncfl) drs_nonlinear::cfl`

Referenced by `rhs()`, and `save_stuff()`.

### 5.17.3.2 `double precision, dimension(:, :, :), allocatable drs_nonlinear::field_p_t`

Referenced by `drs_nonlinear_init()`, `evaluate_real_space()`, and `save_stuff()`.

### 5.17.3.3 `double precision, dimension(:, :, :), allocatable drs_nonlinear::field_r_t`

Referenced by `drs_nonlinear_init()`, `evaluate_real_space()`, and `save_stuff()`.

### 5.17.3.4 `double precision, dimension(:, :, :), allocatable drs_nonlinear::field_t_t`

Referenced by `drs_nonlinear_init()`, `evaluate_real_space()`, and `save_stuff()`.

### 5.17.3.5 `double precision, dimension(:, :, :), allocatable drs_nonlinear::flow_p_t`

Referenced by `drs_nonlinear_init()`, `evaluate_real_space()`, and `save_stuff()`.

**5.17.3.6 double precision,dimension(:, :, :),allocatable drs\_nonlinear::flow\_r\_t**

Quantities in real space.

Referenced by drs\_nonlinear\_init(), evaluate\_real\_space(), and save\_stuff().

**5.17.3.7 double precision,dimension(:, :, :),allocatable drs\_nonlinear::flow\_t\_t**

Referenced by drs\_nonlinear\_init(), evaluate\_real\_space(), and save\_stuff().

**5.17.3.8 integer,parameter drs\_nonlinear::ncfl = 5****5.17.3.9 double precision,allocatable drs\_nonlinear::rhs\_IE\_pol**

Referenced by drs\_nonlinear\_init(), kd\_grothrate(), rhs(), and update\_field().

**5.17.3.10 double precision,allocatable drs\_nonlinear::rhs\_IE\_tor**

IE for Induction Equation.

Referenced by drs\_nonlinear\_init(), kd\_grothrate(), rhs(), and update\_field().

**5.17.3.11 double precision,allocatable drs\_nonlinear::rhs\_NS\_pol**

Referenced by drs\_nonlinear\_init(), rhs(), and update\_flow().

**5.17.3.12 double precision,allocatable drs\_nonlinear::rhs\_NS\_tor**

NS for Navier-Stokes.

Referenced by drs\_nonlinear\_init(), rhs(), and update\_flow().

**5.17.3.13 double precision,allocatable drs\_nonlinear::rhs\_TE**

TE for Temperature Equation.

Referenced by drs\_nonlinear\_init(), rhs(), and update\_temp().

**5.17.3.14 double precision,dimension(:, :, :),allocatable drs\_nonlinear::rot\_field\_p\_t**

Referenced by drs\_nonlinear\_init(), evaluate\_real\_space(), and save\_stuff().

**5.17.3.15 double precision,dimension(:, :, :),allocatable drs\_nonlinear::rot\_field\_r\_t**

Referenced by drs\_nonlinear\_init(), evaluate\_real\_space(), and save\_stuff().

**5.17.3.16 double precision,dimension(:, :, :),allocatable drs\_nonlinear::rot\_field\_t\_t**

Referenced by drs\_nonlinear\_init(), evaluate\_real\_space(), and save\_stuff().

**5.17.3.17 double precision,dimension(:, :, :),allocatable drs\_nonlinear::rot\_flow\_p\_t**

Referenced by `drs_nonlinear_init()`, `evaluate_real_space()`, and `save_stuff()`.

**5.17.3.18 double precision,dimension(:, :, :),allocatable drs\_nonlinear::rot\_flow\_r\_t**

Referenced by `drs_nonlinear_init()`, `evaluate_real_space()`, and `save_stuff()`.

**5.17.3.19 double precision,dimension(:, :, :),allocatable drs\_nonlinear::rot\_flow\_t\_t**

Referenced by `drs_nonlinear_init()`, `evaluate_real_space()`, and `save_stuff()`.

**5.17.3.20 double precision,dimension(:, :, :),allocatable drs\_nonlinear::temp\_t**

Referenced by `drs_nonlinear_init()`, `evaluate_real_space()`, and `save_stuff()`.

## 5.18 drs\_probes Namespace Reference

This module implements some prbing facilities for the running models.

### Functions

- subroutine **drs\_probes\_allocation** ()
- subroutine **drs\_probes\_init** (time)
- subroutine **measure\_lm** ()
- subroutine **average\_unnormalised\_flow\_l\_spectrum** (urspec\_avg)
- subroutine **average\_unnormalised\_field\_l\_spectrum** (Brspec\_avg)
- subroutine **average\_unnormalised\_scalar\_l\_spectrum** (scalar, scalar\_spec\_avg)
- subroutine **l\_spec\_of\_scalar\_field** (field, spec)
 

*Calculate the normalized power spectrum with respect to l of a scalar field.*
- subroutine **m\_spec\_of\_scalar\_field** (field, spec)
 

*Calculates the normalized power spectrum of a scalar field with respect to m.*
- subroutine **n\_spec\_of\_scalar\_field** (field, spec)
 

*Calculates the normalized power spectrum of a scalar quantity f with respect to the Chebyshev polynomials.*

$$R_n = \sum_{l,m} N_l^m (f_{nl}^m)^2$$
- double precision **integrate\_r** (input)
 

*Performs the integration of the 1d real array.*
- function **c\_integrate\_r** (input)
 

*Performs the integration of the 1d complex array.*
- subroutine **save\_magnetic\_dissipation** (mmax)
 

*Computes the magnetic dissipation truncated up to degree.*
- subroutine **save\_flow\_dissipation** (mmax)
 

*Computes the viscous dissipation.*
- subroutine **save\_flow\_coeffs** ()
 

*Saves some flow coefficients at the present instant.*
- subroutine **save\_field\_coeffs** ()
 

*Saves some field coefficients at the present instant.*
- subroutine **check\_resolution\_Hartman** (Rm, error)
- double precision **energy** (vr, vt, vp)
 

*Computes the energy of a vector field based on its components.  
Only root contains the solution.*
- subroutine **measure** (temp2\_t, ur\_t, utheta\_t, uphi\_t, rotu\_r\_t, rotu\_theta\_t, rotu\_phi\_t, Br\_t, Btheta\_t, Bphi\_t, cfl)

- subroutine **compute\_helicities** (ur, ut, up, rotu\_r, rotu\_t, rotu\_p, helicity\_south, helicity\_north)
- subroutine **compute\_advection** (ur, temp, advect)

*Computes the heat transported by advection. as*

$$Q(r) = \int \int u_r(r, \theta, \phi) * (\Theta(r, \theta, \phi) + T_S(r)) \sin \theta d\theta d\phi$$

- double precision **nusselt** (r)

*Computes the Nusselt number, that is, the the ratio between the convective and the diffusive heat fluxes.*

- subroutine **integrate\_power\_surf** (f, n, f\_int)

*Performs the integration in theta and phy of a function f raised to the power n. as*

$$F(r) = \int \int f(r, \theta, \phi)^n \sin \theta d\theta d\phi$$

- subroutine **save\_angular\_momentum** (u\_t, u\_p)

*computes and saves the three cartesian components of the total angular momentum*

## Variables

- double precision, dimension(:), allocatable **ur\_avg**
- double precision, dimension(:), allocatable **ut\_avg**
- double precision, dimension(:), allocatable **up\_avg**
- double precision, dimension(:), allocatable **up2**
- double precision, dimension(:), allocatable **ut2**
- double precision, dimension(:), allocatable **adv\_avg**
- double precision, dimension(:), allocatable **t2\_avg**
- double precision, dimension(:), allocatable **tspec\_avg**
- double precision, dimension(:), allocatable **urspec\_avg**
- double precision, allocatable **Brspec\_avg**
- double precision **groth**
- double precision **Ekin**
- double precision **EB**
- double precision **nkes**

*energies from measure\_lm:*

- double precision **nkea**
- double precision **etors**
- double precision **etora**
- double precision **drkes**
- double precision **drkea**
- double precision **mckes**
- double precision **mckea**
- double precision **Bnkes**
- double precision **Bnkea**
- double precision **Betors**
- double precision **Betora**



- double precision **Bdrkes**
- double precision **Bdrkea**
- double precision **Bmckes**
- double precision **Bmckea**
- double precision, allocatable **dOmega**

*Weights for volume integration.*

- double precision **Rm** = 1.0d0

### 5.18.1 Detailed Description

This module implements some prbing facilities for the running models.

### 5.18.2 Function Documentation

#### 5.18.2.1 subroutine drs\_probes::average\_unnormalised\_field\_l\_spectrum (double precision,dimension(0:nt\_s),intent(inout) *Brspec\_avg*)

References drs\_mpi::blk\_ps\_size, drs\_mpi::blk\_ps\_start, drs\_field::field\_pol, drs\_dims::m0, drs\_mpi::mpi\_rank, drs\_dims::Np\_s, drs\_dims::Nr, and drs\_time::time\_since\_last\_sample.

Referenced by measure\_lm().

#### 5.18.2.2 subroutine drs\_probes::average\_unnormalised\_flow\_l\_spectrum (double precision,dimension(0:nt\_s),intent(inout) *urspec\_avg*)

References drs\_mpi::blk\_ps\_size, drs\_mpi::blk\_ps\_start, drs\_flow::flow\_pol, drs\_dims::m0, drs\_mpi::mpi\_rank, drs\_dims::Np\_s, drs\_dims::Nr, and drs\_time::time\_since\_last\_sample.

Referenced by measure\_lm().

#### 5.18.2.3 subroutine drs\_probes::average\_unnormalised\_scalar\_l\_spectrum (double precision,dimension(0:nt\_s,1:blk\_ps\_size(mpi\_rank),intent(in) *scalar*, double precision,dimension(0:nt\_s),intent(inout) *scalar\_spec\_avg*)

References drs\_mpi::blk\_ps\_start, drs\_dims::m0, drs\_dims::Np\_s, and drs\_time::time\_since\_last\_sample.

Referenced by measure\_lm().

#### 5.18.2.4 function drs\_probes::c\_integrate\_r (input)

Performs the integration of the 1d complex array.

**input in the radial direction.**

Referenced by save\_magnetic\_dissipation().

### 5.18.2.5 subroutine `drs_probes::check_resolution_Hartman` (double precision,intent(in) *Rm*, integer,intent(out) *error*)

References `drs_dims::m0`, `drs_dims::Np`, `drs_dims::Nr`, and `drs_dims::Nt`.

Referenced by `drs_nonlinear::rhs()`.

### 5.18.2.6 subroutine `drs_probes::compute_advection` (double precision,dimension(0:(blk\_t\_size(mpi\_rank)),intent(in) *ur*, double precision,dimension(0:(blk\_t\_size(mpi\_rank)),intent(in) *temp*, double precision,dimension(nr),intent(out) *advect*)

Computes the heat transported by advection. as

$$Q(r) = \int \int u_r(r, \theta, \phi) * (\Theta(r, \theta, \phi) + T_S(r)) \sin \theta d\theta d\phi$$

References `integrate_power_surf()`, and `drs_temp::temp_profile`.

Referenced by `measure()`.

Here is the call graph for this function:



### 5.18.2.7 subroutine `drs_probes::compute_helicities` (double precision,dimension(0:(blk\_t\_size(mpi\_rank)),intent(in) *ur*, double precision,dimension(0:(blk\_t\_size(mpi\_rank)),intent(in) *ut*, double precision,dimension(0:(blk\_t\_size(mpi\_rank)),intent(in) *up*, double precision,dimension(0:(blk\_t\_size(mpi\_rank)),intent(in) *rotu\_r*, double precision,dimension(0:(blk\_t\_size(mpi\_rank)),intent(in) *rotu\_t*, double precision,dimension(0:(blk\_t\_size(mpi\_rank)),intent(in) *rotu\_p*, double precision,intent(out) *helicity\_south*, double precision,intent(out) *helicity\_north*)

References `drs_mpi::blk_t_start`, `dOmega`, and `drs_dims::Nt`.

Referenced by `measure()`.

### 5.18.2.8 subroutine `drs_probes::drs_probes_allocation` ()

References `adv_avg`, `drs_mpi::blk_t_size`, `Brspec_avg`, `dOmega`, `drs_mpi::mpi_rank`, `drs_dims::Nr`, `drs_dims::Nt_s`, `t2_avg`, `tspec_avg`, `up2`, `up_avg`, `ur_avg`, `urspec_avg`, `ut2`, and `ut_avg`.

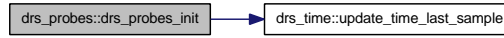
Referenced by `drs_init()`, and `init()`.

### 5.18.2.9 subroutine `drs_probes::drs_probes_init` (double precision *time*)

References `adv_avg`, `drs_mpi::blk_t_size`, `drs_mpi::blk_t_start`, `Brspec_avg`, `dOmega`, `drs_mpi::mpi_rank`, `drs_dims::Np`, `drs_dims::Nr`, `t2_avg`, `tspec_avg`, `up2`, `up_avg`, `drs_time::update_time_last_sample()`, `ur_avg`, `urspec_avg`, `ut2`, and `ut_avg`.

Referenced by `drs_init()`, and `init()`.

Here is the call graph for this function:



**5.18.2.10 double precision drs\_probes::energy** (double precision,dimension(0:blk\_t\_size(mpi\_rank),intent(in) *vr*, double precision,dimension(0:blk\_t\_size(mpi\_rank),intent(in) *vt*, double precision,dimension(0:blk\_t\_size(mpi\_rank),intent(in) *vp*)

Computes the energy of a vector field based on its components.

Only root contains the solution.

References dOmega.

Referenced by Benchmarkv1(), Benchmarkv2(), and measure().

**5.18.2.11 subroutine drs\_probes::integrate\_power\_surf** (double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *f*, integer,intent(in) *n*, double precision,dimension(nr),intent(out) *f\_int*)

Performs the integration in theta and phi of a function *f* raised to the power *n*. as

$$F(r) = \int \int f(r, \theta, \phi)^n \sin \theta d\theta d\phi$$

.

#### Parameters:

*f* The function to be integrated.

*n* The power it should be raised to.

*f\_int* Integral as a function of *r*.

References drs\_mpi::blk\_t\_start.

Referenced by compute\_advection(), and measure().

**5.18.2.12 double precision drs\_probes::integrate\_r** (double precision,dimension(nr),intent(in) *input*)

Performs the integration of the 1d real array.

**input in the radial direction.**

Referenced by measure\_lm().

**5.18.2.13 subroutine drs\_probes::l\_spec\_of\_scalar\_field** (double precision,dimension(0:nt\_s,blk\_ps\_size(mpi\_rank),intent(in) *field*, double precision,dimension(0:nt\_s),intent(out) *spec*)

Calculate the normalized power spectrum with respect to *l* of a scalar field.

**Parameters:**

*field* is in lmr space.

References drs\_mpi::blk\_ps\_start, drs\_dims::m0, and drs\_dims::Np\_s.

Referenced by drs\_io::save\_l\_spec().

#### 5.18.2.14 subroutine drs\_probes::m\_spec\_of\_scalar\_field (double precision,dimension(0:nt\_s,blk\_ps\_size(mpi\_rank),intent(in) *field*, double precision,dimension(m0\*np\_s+1),intent(out) *spec*)

Calculates the normalized power spectrum of a scalar field with respect to m.

**Parameters:**

*field* is in lmr space.

References drs\_mpi::blk\_ps\_start.

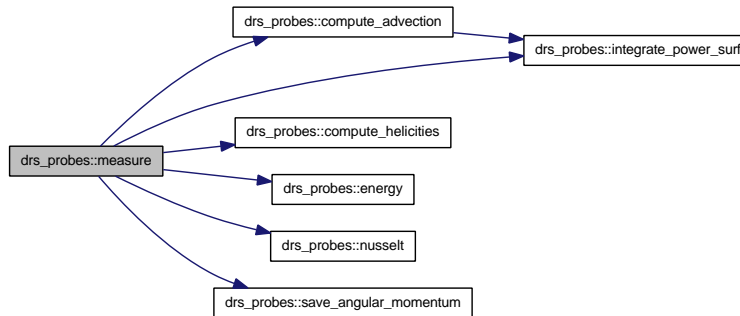
Referenced by drs\_io::save\_m\_spec().

#### 5.18.2.15 subroutine drs\_probes::measure (double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *temp2\_t*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *ur\_t*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *utheta\_t*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *uphi\_t*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *rotu\_r\_t*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *rotu\_theta\_t*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *rotu\_phi\_t*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *Br\_t*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *Btheta\_t*, double precision,dimension(0:(blk\_t\_size(mpi\_rank),intent(in) *Bphi\_t*, double precision,dimension(:),intent(in) *cfl*)

References adv\_avg, Bdrkea, Bdrkes, Betora, Betors, Bmckea, Bmckes, Bnkea, Bnkes, compute\_advection(), compute\_helicities(), drkea, drkes, EB, Ekin, energy(), etora, etors, groth, integrate\_power\_surf(), mckea, mckes, nkea, nkes, nusselt(), Rm, save\_angular\_momentum(), drs\_time::steps, t2\_avg, drs\_time::time, drs\_time::time\_since\_last\_sample, drs\_io\_units::unit\_cfl, drs\_io\_units::unit\_eb, drs\_io\_units::unit\_ek, drs\_io\_units::unit\_nu, drs\_io\_units::unit\_u\_mid, up2, up\_avg, ur\_avg, ut2, and ut\_avg.

Referenced by drs\_nonlinear::save\_stuff().

Here is the call graph for this function:

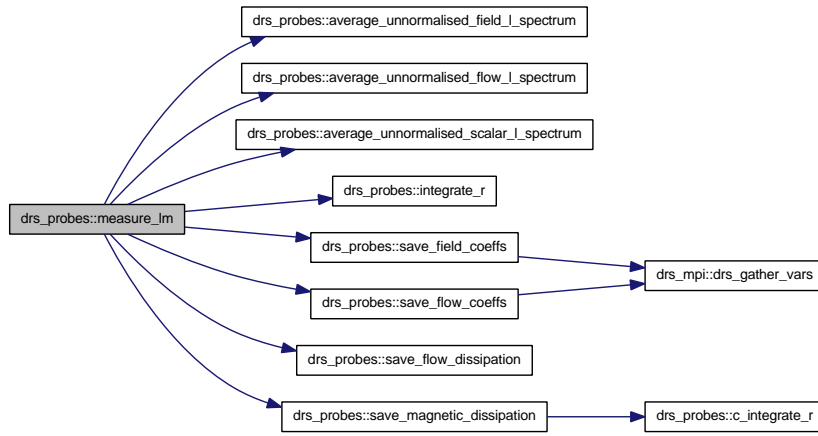


### 5.18.2.16 subroutine drs\_probes::measure\_lm ()

References `average_unnormalised_field_l_spectrum()`, `average_unnormalised_flow_l_spectrum()`, `average_unnormalised_scalar_l_spectrum()`, `Bdrkea`, `Bdrkes`, `Betora`, `Betors`, `drs_mpi::blk_ps_size`, `drs_mpi::blk_ps_start`, `Bmckea`, `Bmckes`, `Bnkea`, `Bnkes`, `Brspec_avg`, `drkea`, `drkes`, `etora`, `etors`, `drs_field::field_pol`, `drs_field::field_pol_dr`, `drs_field::field_tor`, `drs_flow::flow_pol`, `drs_flow::flow_pol_avg`, `drs_flow::flow_pol_dr`, `drs_flow::flow_tor`, `drs_flow::flow_tor_avg`, `integrate_r()`, `drs_dims::m0`, `mckea`, `mckes`, `drs_mpi::mpi_rank`, `nkea`, `nkes`, `drs_dims::Np_s`, `drs_dims::Nt_s`, `save_field_coeffs()`, `save_flow_coeffs()`, `save_flow_dissipation()`, `save_magnetic_dissipation()`, `drs_temp::temp`, `drs_temp::temp_avg`, `drs_temp::temp_dr`, `drs_temp::temp_dr_avg`, `drs_time::time_since_last_sample`, `tspec_avg`, and `urspec_avg`.

Referenced by `drs_nonlinear::save_stuff()`.

Here is the call graph for this function:



### 5.18.2.17 subroutine drs\_probes::n\_spec\_of\_scalar\_field (double precision,dimension(0:nt\_s,blk\_ps\_size(mpi\_rank),intent(in) *field*, double precision,dimension(nr\_s),intent(out) *spec*)

Calculates the normalized power spectrum of a scalar quantity  $f$  with respect to the Chebyshev polynomials.

$$R_n = \sum_{l,m} N_l^m (f_{nl}^m)^2$$

#### Parameters:

*field* is lmr space.

References `drs_mpi::blk_ps_start`, `drs_dims::m0`, and `drs_dims::Np_s`.

Referenced by `drs_io::save_n_spec()`.

### 5.18.2.18 double precision drs\_probes::nusselt (integer,intent(in) *r*)

Computes the Nusselt number, that is, the ratio between the convective and the diffusive heat fluxes.

$$Nu = \frac{\partial(T + \Theta)/\partial r}{\partial T/\partial r}$$

**Todo**

It only works for serial runs. Needs to be parallelized.

References `drs_temp::temp_dr`, and `drs_temp::temp_profile_dr`.

Referenced by `measure()`.

#### 5.18.2.19 subroutine `drs_probes::save_angular_momentum` (double precision,dimension(0:(blk\_t\_size(mpi\_rank)),intent(in) *u\_t*, double precision,dimension(0:(blk\_t\_size(mpi\_rank)),intent(in) *u\_p*)

computes and saves the three cartesian components of the total angular momentum

**Todo**

Split the saving part and move it to io.

**Parameters:**

*u\_t* Theta component of the flow in real (tpr) space.

*u\_p* Phi component of the flow in real (tpr) space.

References `drs_mpi::blk_t_start`, `dOmega`, `drs_dims::m0`, `drs_time::time`, and `drs_io_units::unit_am`.

Referenced by `measure()`.

#### 5.18.2.20 subroutine `drs_probes::save_field_coeffs` ()

Saves some field coefficients at the present instant.

**Todo**

Should take a list of l's and m's and reply with a list of values

Writing should be moved to io.

References `drs_mpi::blk_ps_size`, `drs_mpi::blk_ps_start`, `drs_mpi::drs_gather_vars()`, `drs_field::field_pol`, `drs_field::field_tor`, `drs_dims::m0`, `drs_mpi::mpi_rank`, `drs_dims::Nr`, `drs_time::time`, and `drs_io_units::unit_koeb`.

Referenced by `measure_lm()`.

Here is the call graph for this function:



#### 5.18.2.21 subroutine `drs_probes::save_flow_coeffs` ()

Saves some flow coefficients at the present instant.

**Todo**

Should take a list of l's and m's and reply with a list of values  
Writing should be moved to io.

References drs\_mpi::blk\_ps\_size, drs\_mpi::blk\_ps\_start, drs\_mpi::drs\_gather\_vars(), drs\_flow::flow\_pol, drs\_flow::flow\_tor, drs\_dims::m0, drs\_mpi::mpi\_rank, drs\_dims::Nr, drs\_time::time, and drs\_io\_units::unit\_koeu.

Referenced by measure\_lm().

Here is the call graph for this function:

**5.18.2.22 subroutine drs\_probes::save\_flow\_dissipation (integer,intent(in) mmax)**

Computes the viscous dissipation.

**Todo**

Separate computing from writing.

References drs\_mpi::blk\_ps\_start, drs\_flow::flow\_pol, drs\_flow::flow\_pol\_ddr, drs\_flow::flow\_pol\_dr, drs\_flow::flow\_tor, drs\_flow::flow\_tor\_ddr, drs\_flow::flow\_tor\_dr, drs\_dims::m0, drs\_time::time, and drs\_io\_units::unit\_dissu.

Referenced by measure\_lm().

**5.18.2.23 subroutine drs\_probes::save\_magnetic\_dissipation (integer,intent(in) mmax)**

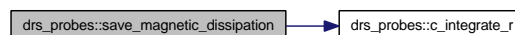
Computes the magnetic dissipation truncated up to degree.

**mmax.**

References drs\_mpi::blk\_ps\_start, c\_integrate\_r(), drs\_field::field\_pol, drs\_field::field\_pol\_ddr, drs\_field::field\_pol\_dr, drs\_field::field\_tor, drs\_field::field\_tor\_ddr, drs\_field::field\_tor\_dr, drs\_field::field\_tor\_lap, drs\_dims::m0, drs\_time::time, and drs\_io\_units::unit\_dissB.

Referenced by measure\_lm().

Here is the call graph for this function:

**5.18.3 Variable Documentation****5.18.3.1 double precision,dimension(:),allocatable drs\_probes::adv\_avg**

Referenced by drs\_probes\_allocation(), drs\_probes\_init(), drs\_io::dump\_state(), and measure().

**5.18.3.2 double precision drs\_probes::Bdrkea**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.3 double precision drs\_probes::Bdrkes**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.4 double precision drs\_probes::Betora**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.5 double precision drs\_probes::Betors**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.6 double precision drs\_probes::Bmckea**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.7 double precision drs\_probes::Bmckes**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.8 double precision drs\_probes::Bnkea**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.9 double precision drs\_probes::Bnkes**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.10 double precision,allocatable drs\_probes::Brspec\_avg**

Referenced by `drs_probes_allocation()`, `drs_probes_init()`, and `measure_lm()`.

**5.18.3.11 double precision,allocatable drs\_probes::dOmega**

Weights for volume integration.

Referenced by `compute_helicities()`, `drs_probes_allocation()`, `drs_probes_init()`, `energy()`, `save_angular_momentum()`, `saveDXmeridional()`, `drs_io_DX::saveDXmeridional()`, and `saveIDLmeridional()`.

**5.18.3.12 double precision drs\_probes::drkea**

Referenced by `measure()`, and `measure_lm()`.



**5.18.3.13 double precision drs\_probes::drkes**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.14 double precision drs\_probes::EB**

Referenced by `measure()`.

**5.18.3.15 double precision drs\_probes::Ekin**

Referenced by `Benchmarkv1()`, `Benchmarkv2()`, and `measure()`.

**5.18.3.16 double precision drs\_probes::etora**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.17 double precision drs\_probes::etors**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.18 double precision drs\_probes::groth**

Referenced by `measure()`.

**5.18.3.19 double precision drs\_probes::mckea**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.20 double precision drs\_probes::mckes**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.21 double precision drs\_probes::nkea**

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.22 double precision drs\_probes::nkes**

energies from `measure_lm`:

Referenced by `measure()`, and `measure_lm()`.

**5.18.3.23 double precision drs\_probes::Rm = 1.0d0**

Referenced by `measure()`, and `drs_nonlinear::rhs()`.

**5.18.3.24 double precision,dimension(:),allocatable drs\_probes::t2\_avg**

Referenced by `drs_probes_allocation()`, `drs_probes_init()`, `drs_io::dump_state()`, and `measure()`.

**5.18.3.25 double precision,dimension(:),allocatable drs\_probes::tspec\_avg**

Referenced by `drs_probes_allocation()`, `drs_probes_init()`, and `measure_lm()`.

**5.18.3.26 double precision,dimension(:),allocatable drs\_probes::up2**

Referenced by `drs_probes_allocation()`, `drs_probes_init()`, `drs_io::dump_state()`, and `measure()`.

**5.18.3.27 double precision,dimension(:),allocatable drs\_probes::up\_avg**

Referenced by `drs_probes_allocation()`, `drs_probes_init()`, `drs_io::dump_state()`, and `measure()`.

**5.18.3.28 double precision,dimension(:),allocatable drs\_probes::ur\_avg**

Referenced by `drs_probes_allocation()`, `drs_probes_init()`, `drs_io::dump_state()`, and `measure()`.

**5.18.3.29 double precision,dimension(:),allocatable drs\_probes::urspec\_avg**

Referenced by `drs_probes_allocation()`, `drs_probes_init()`, and `measure_lm()`.

**5.18.3.30 double precision,dimension(:),allocatable drs\_probes::ut2**

Referenced by `drs_probes_allocation()`, `drs_probes_init()`, `drs_io::dump_state()`, and `measure()`.

**5.18.3.31 double precision,dimension(:),allocatable drs\_probes::ut\_avg**

Referenced by `drs_probes_allocation()`, `drs_probes_init()`, `drs_io::dump_state()`, and `measure()`.

## 5.19 drs\_radial Namespace Reference

This module implements the radial domain and operations in it.

### Functions

- subroutine **drs\_radial\_init** (riro)  
*Initializes the radial domain and the Chebyshev polynomials and their derivatives.*
- double precision, dimension(nr) **radial\_derivative\_r2r** (radarr)  
*Returns the first derivative of radarr. radarr is supposed to be given in direct space, derivative is returned in direct space.*
- subroutine **radial\_dr\_ddr\_1D\_n2r** (t, t1, t2)  
*Returns first radial derivative in t1, second derivative in t2. Input t is supposed to be given in spectral space. On output both t and its derivatives are returned in real space.*
- subroutine **radial\_dr\_ddr\_1D\_r2r** (t, t1, t2)  
*A factor of 2 for each derivative is due to the mapping from the radial coordinate r to the Chebyshev coordinate x, where x runs from -1 to 1. The interrelation is  $r = \eta/(1-\eta) + 0.5(x+1)$  see the def. of rcoll in the initialization routine. Obviously,  $d/dr = 2*d/dx$ .*
- subroutine **radial\_dr\_ddr\_3D\_r2r** (t, t1, t2)  
*Calculates first and second radial derivatives of 3D-array in lmr space. Includes dealiasing in n.*
- subroutine **radial\_dr\_ddr\_3D\_n2r** (t0, t1, t2)  
*Calculates first and second radial derivatives of 3D-array in lmn space. includes dealiasing in n. Transforms the original field to lmr space.*

### Variables

- double precision, dimension(:), allocatable **rcoll**  
*Radial collocation points for Chebychev polynomials.*
- double precision, dimension(:), allocatable **rcoll2**  
*Squares of radial collocation points for Chebychev polynomials.*
- double precision, dimension(:), allocatable **drcoll**  
*Differences for radial collocation points for Chebychev polynomials.*
- double precision, dimension(:, :), allocatable **poly**
- double precision, dimension(:, :), allocatable **poly\_dr**
- double precision, dimension(:, :), allocatable **poly\_ddr**
- integer, dimension(2) **b**  
*Index of the boundaries: b(1)=inner boundary; b(2)=outer boundary.*

### 5.19.1 Detailed Description

This module implements the radial domain and operations in it.

### 5.19.2 Function Documentation

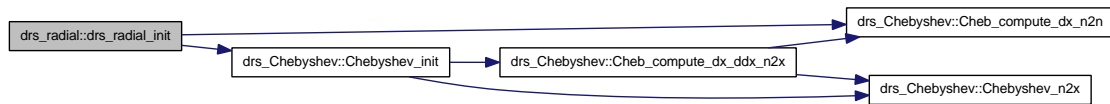
#### 5.19.2.1 subroutine `drs_radial::drs_radial_init` (double precision,intent(in) *riro*)

Initializes the radial domain and the Chebyshev polynomials and their derivatives.

References `b`, `drs_Chebyshev::Cheb_compute_dx_n2n()`, `drs_Chebyshev::Cheb_x`, `drs_Chebyshev::Chebyshev`, `drs_Chebyshev::Chebyshev_ddx`, `drs_Chebyshev::Chebyshev_dx`, `drs_Chebyshev::Chebyshev_init()`, `drcoll`, `drs_dims::Nr`, `drs_dims::Nr_s`, `poly`, `poly_ddr`, `poly_dr`, `rcoll`, and `rcoll2`.

Referenced by `drs_init()`, `init()`, `test_drs_radial()`, and `test_saveDXMer()`.

Here is the call graph for this function:



#### 5.19.2.2 double precision,dimension(nr) `drs_radial::radial_derivative_r2r` (double precision,dimension(nr),intent(in) *radarr*)

Returns the first derivative of *radarr*. *radarr* is supposed to be given in direct space, derivative is returned in direct space.

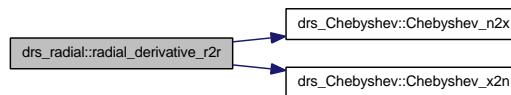
Since:

1.6.6

References `drs_Chebyshev::Chebyshev_n2x()`, and `drs_Chebyshev::Chebyshev_x2n()`.

Referenced by `drs_transforms::my_div()`, `drs_transforms::my_rotrot()`, and `test_radial_derivative_r2r()`.

Here is the call graph for this function:



#### 5.19.2.3 subroutine `drs_radial::radial_dr_ddr_1D_n2r` (double precision,dimension(nr),intent(inout) *t*, double precision,dimension(nr),intent(out) *tI*, double precision,dimension(nr),intent(out) *t2*)

Returns first radial derivative in *tI*, second derivative in *t2*. Input *t* is supposed to be given in spectral space. On output both *t* and its derivatives are returned in real space.

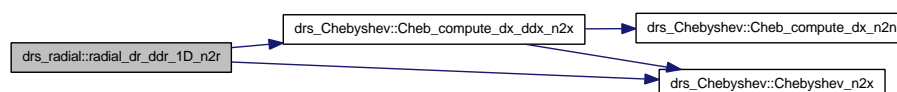
Since:

1.6.6

References `drs_Chebyshev::Cheb_compute_dx_ddx_n2x()`, and `drs_Chebyshev::Chebyshev_n2x()`.

Referenced by `mk_green()`, and `radial_dr_ddr_1D_r2r()`.

Here is the call graph for this function:



#### 5.19.2.4 subroutine `drs_radial::radial_dr_ddr_1D_r2r` (double precision,dimension(nr),intent(in) *t*, double precision,dimension(nr),intent(out) *t1*, double precision,dimension(nr),intent(out) *t2*)

A factor of 2 for each derivative is due to the mapping from the radial coordinate *r* to the Chebyshev coordinate *x*, where *x* runs from -1 to 1. The interrelation is  $r = \eta / (1 - \eta) + 0.5(x + 1)$  see the def. of `rcoll` in the initialization routine. Obviously,  $d/dr = 2 * d/dx$ . Returns first radial derivative in *t1*, second derivative in *t2*. Input *t* is supposed to be given in real space. On output both derivatives are returned in real space.

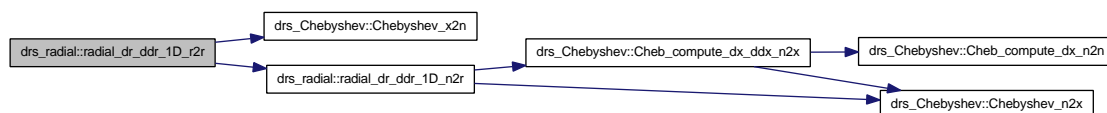
Since:

1.6.6

References `drs_Chebyshev::Chebyshev_x2n()`, and `radial_dr_ddr_1D_n2r()`.

Referenced by `test_radial_dr_ddr_1D_r2r()`, and `CrankNicholson::updateCrankNicholson_matrices()`.

Here is the call graph for this function:



#### 5.19.2.5 subroutine `drs_radial::radial_dr_ddr_3D_n2r` (double precision,dimension(0:nt\_s, blk\_ps\_size(mpi\_rank),intent(inout) *t0*, double precision,dimension(0:nt\_s, blk\_ps\_size(mpi\_rank),intent(out) *t1*, double precision,dimension(0:nt\_s, blk\_ps\_size(mpi\_rank),intent(out) *t2*)

Calculates first and second radial derivatives of 3D-array in lmn space. includes dealiasing in *n*. Transforms the original field to lmr space.

**Parameters:**

*t0* The original field. lmn space on entry, lmr space on exit.

*t1* The first radial drivative in lmr space.

*t2* The second radial drivative in lmr space.

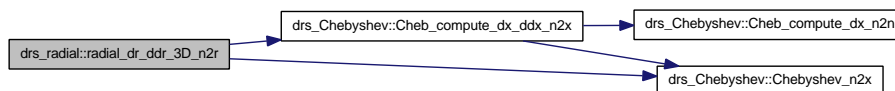
**Since:**

1.6.6

References `drs_mpi::blk_ts_start`, `drs_Chebyshev::Cheb_compute_dx_ddx_n2x()`, and `drs_Chebyshev::Chebyshev_n2x()`.

Referenced by `update_field()`, `update_flow()`, and `update_temp()`.

Here is the call graph for this function:



### 5.19.2.6 subroutine `drs_radial::radial_dr_ddr_3D_r2r` (double precision,dimension(0:nt\_s, blk\_ps\_size(mpi\_rank),intent(in) *t*, double precision,dimension(0:nt\_s, blk\_ps\_size(mpi\_rank),intent(out) *t1*, double precision,dimension(0:nt\_s, blk\_ps\_size(mpi\_rank),intent(out) *t2*)

Calculates first and second radial derivatives of 3D-array in lmr space. Includes dealiasing in n.

**Parameters:**

- t0* The original field in lmr space.
- t1* The first radial drivative in lmr space.
- t2* The second radial drivative in lmr space.

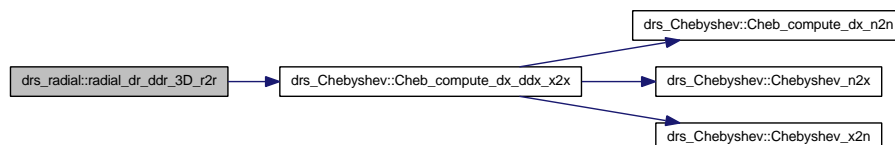
**Since:**

1.6.6

References `drs_mpi::blk_ts_start`, and `drs_Chebyshev::Cheb_compute_dx_ddx_x2x()`.

Referenced by `drs()`, `drs_comp::drs_comp_init()`, `drs_field::drs_field_init()`, `drs_flow::drs_flow_init()`, and `drs_temp::drs_temp_init()`.

Here is the call graph for this function:



## 5.19.3 Variable Documentation

### 5.19.3.1 integer,dimension(2) `drs_radial::b`

Index of the boundaries: `b(1)`=inner boundary; `b(2)`=outer boundary.

Referenced by `drs_radial_init()`.

**5.19.3.2 double precision,dimension(:),allocatable drs\_radial::drcoll**

Differences for radial collocation points for Chebychev polynomials.

Referenced by `cacheTemperatureProfile()`, `drs_field::calc_field_lspec()`, `drs_field::calc_field_mspec()`, `drs_flow::calc_flow_lspec()`, `drs_flow::calc_flow_mspec()`, `drs_radial_init()`, `redefine_radial_coordinate()`, `drs_nonlinear::save_stuff()`, and `selectEquatorMidShell()`.

**5.19.3.3 double precision,dimension(:,),allocatable drs\_radial::poly**

Referenced by `drs_radial_init()`, and `CrankNicholson::updateCrankNicholson_matrices()`.

**5.19.3.4 double precision,dimension(:,),allocatable drs\_radial::poly\_ddd**

Referenced by `drs_radial_init()`, and `CrankNicholson::updateCrankNicholson_matrices()`.

**5.19.3.5 double precision,dimension(:,),allocatable drs\_radial::poly\_dr**

Referenced by `drs_radial_init()`, and `CrankNicholson::updateCrankNicholson_matrices()`.

**5.19.3.6 double precision,dimension(:),allocatable drs\_radial::rcoll**

Radial collocation points for Chebychev polynomials.

Referenced by `drs_field::apply_field_tor_BC()`, `Benchmarkv1()`, `cacheTemperatureProfile()`, `drs_field::calc_field()`, `drs_flow::calc_flow()`, `drs_flow::calc_flow_lspec()`, `drs_flow::calc_flow_mspec()`, `drs_flow::calc_flow_nspec()`, `drs_field::calc_rot_field()`, `drs_flow::calc_rot_flow()`, `drs_comp::drs_comp_init()`, `drs_comp::drs_comp_reset()`, `drs_field::drs_field_random_init()`, `drs_radial_init()`, `drs_temp::drs_temp_init()`, `drs_temp::drs_temp_randomize()`, `getProfile()`, `kd_grothrate()`, `drs_transforms::PolTor_common2PolTor_flow()`, `redefine_radial_coordinate()`, `drs_nonlinear::save_stuff()`, `drs_io_DX::saveDXmeridional()`, `saveDXmeridional()`, `drs_io_DX::saveDXmeridional3DVec()`, `drs_io_DX::saveDXvolume()`, `drs_io_DX::saveDXvolume3DVec()`, `drs_io_DX::saveDXvolume_v2()`, `saveIDLmeridional()`, `selectEquatorMidShell()`, `test_radial_colocation_points()`, `test_radial_derivative_r2r()`, `test_radial_dr_ddd_1D_r2r()`, `test_vectorField2Divergence()`, `update_flow()`, `drs_temp::update_temp_lap()`, `CrankNicholson::updateCrankNicholson_matrices()`, `drs_transforms::vectorField2Divergence()`, and `drs_transforms::vectorField2PolTor_common()`.

**5.19.3.7 double precision,dimension(:),allocatable drs\_radial::rcoll2**

Squares of radial collocation points for Chebychev polynomials.

Referenced by `drs_field::calc_field()`, `drs_field::calc_field_lspec()`, `drs_field::calc_field_mspec()`, `drs_field::calc_field_nspec()`, `drs_field::calc_rot_field()`, `drs_comp::drs_comp_init()`, `drs_comp::drs_comp_reset()`, `drs_radial_init()`, `drs_temp::drs_temp_init()`, `drs_transforms::my_div()`, `drs_transforms::my_rotrot()`, `drs_transforms::PolTor_common2PolTor_field()`, `redefine_radial_coordinate()`, `test_radial_derivative_r2r()`, `test_radial_dr_ddd_1D_r2r()`, `drs_field::update_field_pol_lap()`, `drs_field::update_field_tor_lap()`, `drs_flow::update_flow_pol_lap()`, `drs_flow::update_flow_tor_lap()`, `drs_temp::update_temp_lap()`, `CrankNicholson::updateCrankNicholson_matrices()`, `drs_transforms::vectorField2Divergence()`, and `drs_transforms::vectorField2PolTor_common()`.

## 5.20 drs\_renderers Namespace Reference

### Functions

- subroutine **drs\_renderers\_allocation** (what)
- subroutine **render** (what)
 

*Makes a decision about what to render.*

*Numbers are coded as:*

*~~~~~ a b c d e | | | |> e - component 1, 2 or 3 for vectors, irrelevant for scalars | | |> d - coordinate system or stream lines | |> c - quantity to be plotted |> b - curl, gradient or divergence or 0 |> a - scalar product with selection or 0.*
- subroutine **render\_ur** ()
- subroutine **render\_u** ()
- subroutine **render\_Br** ()
- subroutine **render\_Bt** ()
- subroutine **render\_Bp** ()
 

*Renders the azimuthal component of the magnetic field.*
- subroutine **render\_Bz** ()
 

*Renders the z component of the magnetic field.*
- subroutine **render\_B** ()
 

*Render all three spherical components of the magnetic field.*
- subroutine **render\_B\_outside** ()
 

*Render all three spherical components of the magnetic field outside the outer core.*
- subroutine **render\_rotu\_r** ()
 

*Renders the radial component of the curl of the flow.*
- subroutine **render\_rotu** ()
 

*Renders all three spherical components of the curl of the flow (vorticity).*
- subroutine **render\_up** ()
 

*u\_phi:*
- subroutine **render\_rotu\_p** ()
 

*rot(u)\_phi:*
- subroutine **render\_ut** ()
 

*u\_theta:*
- subroutine **render\_rotu\_t** ()
 

*rot(u)\_theta:*
- subroutine **render\_uz** ()
 

*u\_z*
- subroutine **render\_rotu\_z** ()



*rot(u)\_z:*

- subroutine **render\_temperature\_perturbation** ()

*Renders the temperature perturbation.*

- subroutine **render\_temperature** ()

*Renders the total temperature.*

- subroutine **render\_helicity** ()

*Renders helicity.*

- subroutine **render\_temprature\_grad\_r** ()

- subroutine **render\_streamlines\_t** ()

- subroutine **render\_poloidal\_streamlines** ()

*Renders the poloidal flow streamlines.*

- subroutine **render\_radial\_streamfunction** ()

*radial stream function for the flow*

## Variables

- double precision, dimension(:, :, :), allocatable **render\_out**
- double precision, allocatable **XX**
- double precision, allocatable **YY**
- double precision, allocatable **ZZ**

### 5.20.1 Function Documentation

#### 5.20.1.1 subroutine drs\_renderers::drs\_renderers\_allocation (integer,intent(in) what)

References drs\_dims::Np, drs\_dims::Nr, drs\_dims::Nt, render\_out, XX, YY, and ZZ.

Referenced by init().

#### 5.20.1.2 subroutine drs\_renderers::render (integer,intent(in) what)

Makes a decision about what to render.

Numbers are coded as:

~~~~~ a b c d e | | | |>e - component 1, 2 or 3 for vectors, irrelevant for scalars | | |> d - coordinate system or stream lines | |> c - quantity to be plotted | |> b - curl, gradient or divergence or 0 |> a - scalar product with selection or 0. e = 1, 2 or 3 for first second or third coordinate or meridional, azimuthal and poloidal streamlines 1 or 2 for total or anomaly scalar fields 4 for all three coordinates d = 1, 2 or 3 for cartesian (x,y,x), spherical (r,t,p) or cyllindrical (s, p, z) components respectively, 4 for streamlines, 0 for none c = 1 for the flow 2 for the magetic field 3 for the temperature field 4 for the composition field 5 for the magetic field outside the core (up to ro+1) b = 1 for the curl 2 for the gradient 3 for the divergence 0 for nothing a = 1 for scalar product with flow 2 for scalar product with field 0 for nothing ~~~~~

For example, if I want the meridional (spherical coordinates) component of the curl of the flow, a=0, b=1, c=1, d=2, e=2 so

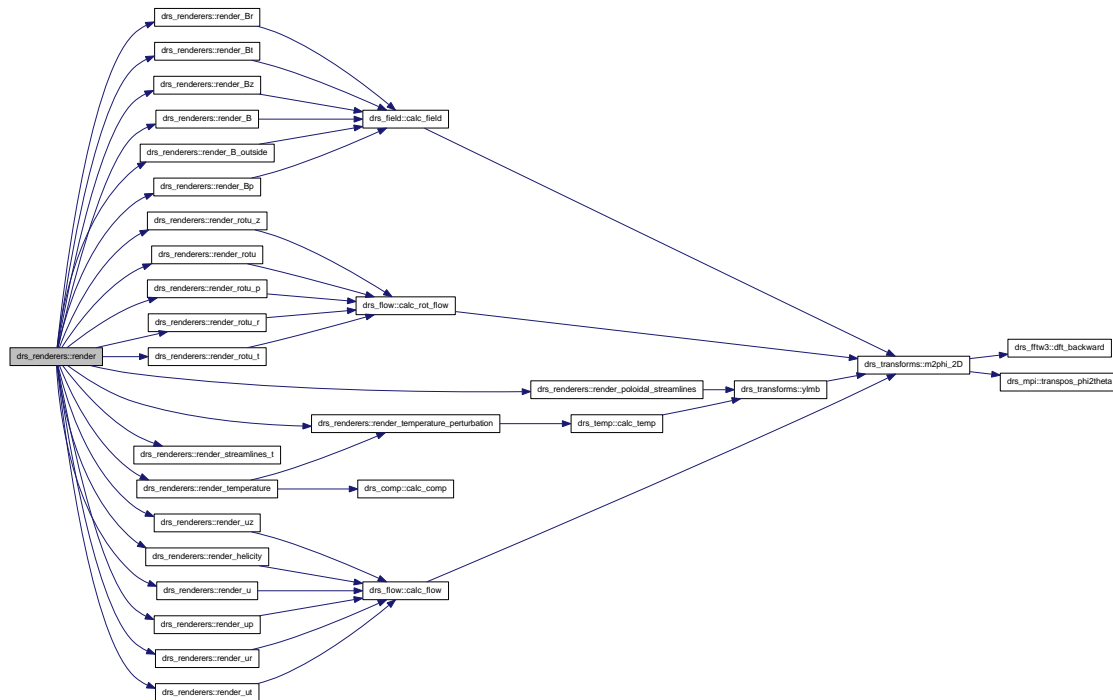
Parameters:

what = 01122

References `render_B()`, `render_B_outside()`, `render_Bp()`, `render_Br()`, `render_Bt()`, `render_Bz()`, `render_-helicity()`, `render_poloidal_streamlines()`, `render_rotu()`, `render_rotu_p()`, `render_rotu_r()`, `render_rotu_-t()`, `render_rotu_z()`, `render_streamlines_t()`, `render_temperature()`, `render_temperature_perturbation()`, `render_u()`, `render_up()`, `render_ur()`, `render_ut()`, and `render_uz()`.

Referenced by `drs2dx()`.

Here is the call graph for this function:

**5.20.13 subroutine `drs_renderers::render_B()`**

Render all three spherical components of the magnetic field.

References `drs_field::calc_field()`, `drs_field::field_pol`, `drs_field::field_pol_ddr`, `drs_field::field_pol_dr`, `drs_field::field_tor`, `drs_field::field_tor_ddr`, `drs_field::field_tor_dr`, `XX`, `YY`, and `ZZ`.

Referenced by `render()`.

Here is the call graph for this function:



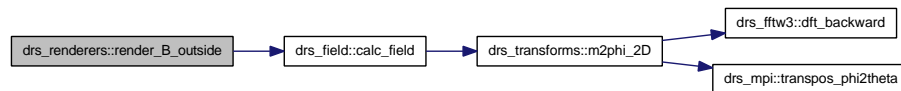
5.20.1.4 subroutine drs_renderers::render_B_outside ()

Render all three spherical components of the magnetic field outside the outer core.

References drs_mpi::blk_ps_size, drs_field::calc_field(), drs_field::field_pol, drs_field::field_pol_ddr, drs_field::field_pol_dr, drs_field::field_tor, drs_field::field_tor_ddr, drs_field::field_tor_dr, drs_mpi::mpi_rank, drs_dims::Nt_s, XX, YY, and ZZ.

Referenced by render().

Here is the call graph for this function:

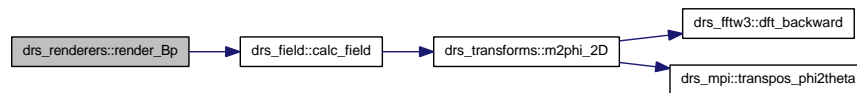
**5.20.1.5 subroutine drs_renderers::render_Bp ()**

Renders the azimuthal component of the magnetic field.

References drs_field::calc_field(), drs_field::field_pol, drs_field::field_pol_ddr, drs_field::field_pol_dr, drs_field::field_tor, drs_field::field_tor_ddr, drs_field::field_tor_dr, and render_out.

Referenced by render().

Here is the call graph for this function:

**5.20.1.6 subroutine drs_renderers::render_Br ()**

References drs_field::calc_field(), drs_field::field_pol, drs_field::field_pol_ddr, drs_field::field_pol_dr, drs_field::field_tor, drs_field::field_tor_ddr, drs_field::field_tor_dr, and render_out.

Referenced by render().

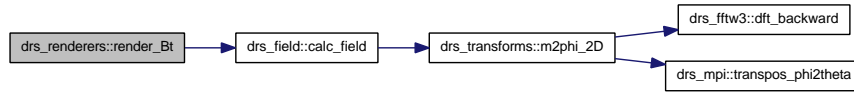
Here is the call graph for this function:

**5.20.1.7 subroutine drs_renderers::render_Bt ()**

References drs_field::calc_field(), drs_field::field_pol, drs_field::field_pol_ddr, drs_field::field_pol_dr, drs_field::field_tor, drs_field::field_tor_ddr, drs_field::field_tor_dr, and render_out.

Referenced by render().

Here is the call graph for this function:



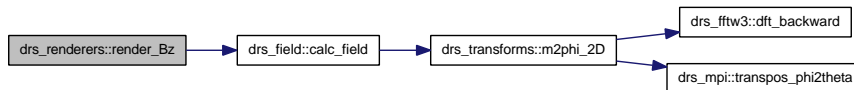
5.20.1.8 subroutine drs_renderers::render_Bz ()

Renders the z component of the magnetic field.

References `drs_field::calc_field()`, `drs_legendre::costheta`, `drs_field::field_pol_ddr`, `drs_field::field_pol_dr`, `drs_field::field_tor_ddr`, `drs_field::field_tor_dr`, `drs_legendre::pi`, and `render_out`.

Referenced by `render()`.

Here is the call graph for this function:



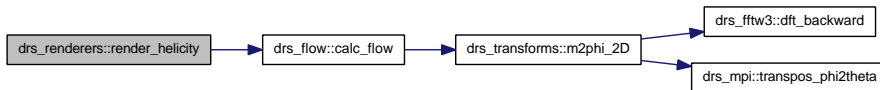
5.20.1.9 subroutine drs_renderers::render_helicity ()

Renders helicity.

References `drs_flow::calc_flow()`, `drs_flow::flow_pol`, `drs_flow::flow_pol_ddr`, `drs_flow::flow_pol_dr`, `drs_flow::flow_tor`, `drs_flow::flow_tor_ddr`, `drs_flow::flow_tor_dr`, and `render_out`.

Referenced by `render()`.

Here is the call graph for this function:



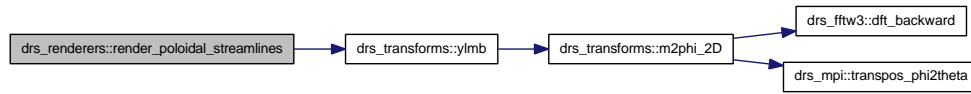
5.20.1.10 subroutine drs_renderers::render_poloidal_streamlines ()

Renders the poloidal flow streamlines.

References `drs_flow::flow_pol`, `drs_dims::m0`, `drs_dims::Np_s`, `drs_dims::Nt_s`, `render_out`, and `drs_transforms::ylmb()`.

Referenced by `render()`.

Here is the call graph for this function:

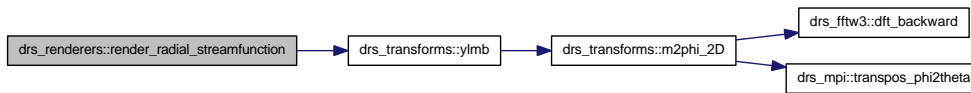


5.20.1.11 subroutine drs_renderers::render_radial_streamfunction ()

radial stream function for the flow

References drs_flow::flow_tor, render_out, and drs_transforms::ylmb().

Here is the call graph for this function:



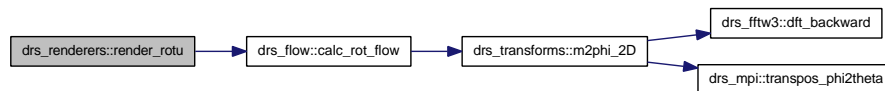
5.20.1.12 subroutine drs_renderers::render_rotu ()

Renders all three spherical components of the curl of the flow (vorticity).

References drs_flow::calc_rot_flow(), drs_flow::flow_pol, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, drs_flow::flow_tor, drs_flow::flow_tor_ddr, drs_flow::flow_tor_dr, XX, YY, and ZZ.

Referenced by render().

Here is the call graph for this function:



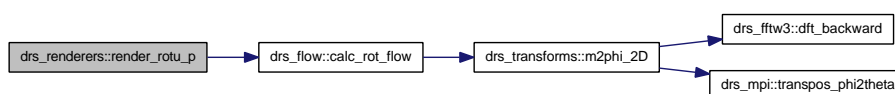
5.20.1.13 subroutine drs_renderers::render_rotu_p ()

rot(u)_phi:

References drs_flow::calc_rot_flow(), drs_flow::flow_pol, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, drs_flow::flow_tor, drs_flow::flow_tor_ddr, drs_flow::flow_tor_dr, and render_out.

Referenced by render().

Here is the call graph for this function:



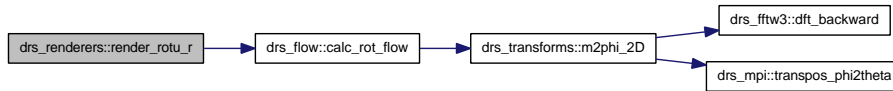
5.20.1.14 subroutine `drs_renderers::render_rotu_r ()`

Renders the radial component of the curl of the flow.

References `drs_flow::calc_rot_flow()`, `drs_flow::flow_pol`, `drs_flow::flow_pol_ddr`, `drs_flow::flow_pol_dr`, `drs_flow::flow_tor`, `drs_flow::flow_tor_ddr`, `drs_flow::flow_tor_dr`, and `render_out`.

Referenced by `render()`.

Here is the call graph for this function:



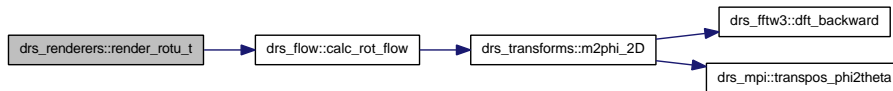
5.20.1.15 subroutine `drs_renderers::render_rotu_t ()`

`rot(u)_theta`:

References `drs_flow::calc_rot_flow()`, `drs_flow::flow_pol`, `drs_flow::flow_pol_ddr`, `drs_flow::flow_pol_dr`, `drs_flow::flow_tor`, `drs_flow::flow_tor_ddr`, `drs_flow::flow_tor_dr`, and `render_out`.

Referenced by `render()`.

Here is the call graph for this function:



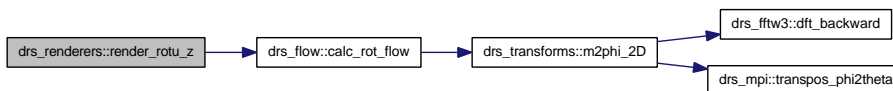
5.20.1.16 subroutine `drs_renderers::render_rotu_z ()`

`rot(u)_z`:

References `drs_flow::calc_rot_flow()`, `drs_legendre::costheta`, `drs_flow::flow_pol`, `drs_flow::flow_pol_ddr`, `drs_flow::flow_pol_dr`, `drs_flow::flow_tor`, `drs_flow::flow_tor_ddr`, `drs_flow::flow_tor_dr`, `drs_legendre::pi`, and `render_out`.

Referenced by `render()`.

Here is the call graph for this function:



5.20.1.17 subroutine `drs_renderers::render_streamlines_t ()`

References `drs_legendre::costheta`, `drs_legendre::dleg`, `drs_flow::flow_pol`, `drs_dims::m0`, `drs_dims::Nr`, `drs_dims::Nt_s`, and `render_out`.

Referenced by render().

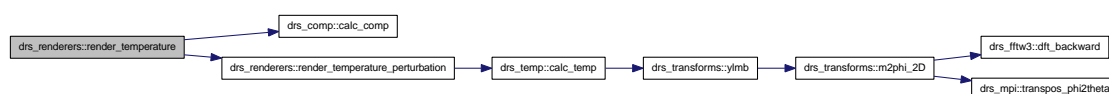
5.20.1.18 subroutine drs_renderers::render_temperature ()

Renders the total temperature.

References drs_comp::calc_comp(), drs_comp::comp, drs_comp::comp_profile, drs_dims::Np, drs_dims::Nr, drs_dims::Nt, render_out, render_temperature_perturbation(), and drs_temp::temp_profile.

Referenced by render().

Here is the call graph for this function:



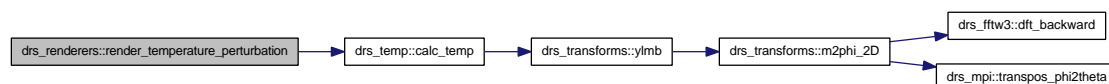
5.20.1.19 subroutine drs_renderers::render_temperature_perturbation ()

Renders the temperature perturbation.

References drs_temp::calc_temp(), and render_out.

Referenced by render(), and render_temperature().

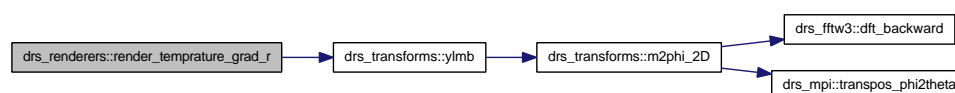
Here is the call graph for this function:



5.20.1.20 subroutine drs_renderers::render_temprature_grad_r ()

References drs_dims::Np, drs_dims::Nr, drs_dims::Nt, render_out, drs_temp::temp, and drs_transforms::ylmb().

Here is the call graph for this function:



5.20.1.21 subroutine drs_renderers::render_u ()

References drs_flow::calc_flow(), drs_flow::flow_pol, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, drs_flow::flow_tor, drs_flow::flow_tor_ddr, drs_flow::flow_tor_dr, XX, YY, and ZZ.

Referenced by render().

Here is the call graph for this function:



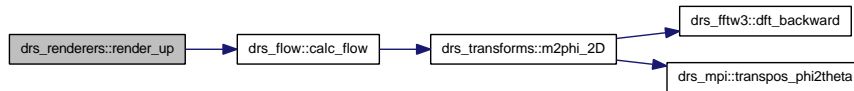
5.20.1.22 subroutine drs_renderers::render_up ()

u_phi:

References drs_flow::calc_flow(), drs_flow::flow_pol, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, drs_flow::flow_tor, drs_flow::flow_tor_ddr, drs_flow::flow_tor_dr, and render_out.

Referenced by render().

Here is the call graph for this function:



5.20.1.23 subroutine drs_renderers::render_ur ()

References drs_flow::calc_flow(), drs_flow::flow_pol, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, drs_flow::flow_tor, drs_flow::flow_tor_ddr, drs_flow::flow_tor_dr, and render_out.

Referenced by render().

Here is the call graph for this function:



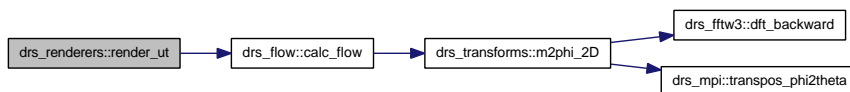
5.20.1.24 subroutine drs_renderers::render_ut ()

u_theta:

References drs_flow::calc_flow(), drs_flow::flow_pol, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, drs_flow::flow_tor, drs_flow::flow_tor_ddr, drs_flow::flow_tor_dr, and render_out.

Referenced by render().

Here is the call graph for this function:



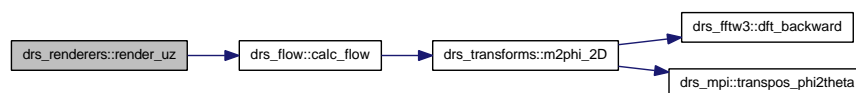
5.20.1.25 subroutine drs_renderers::render_uz ()

u_z

References drs_flow::calc_flow(), drs_legendre::costheta, drs_flow::flow_pol, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, drs_flow::flow_tor, drs_flow::flow_tor_ddr, drs_flow::flow_tor_dr, drs_legendre::pi, and render_out.

Referenced by render().

Here is the call graph for this function:

**5.20.2 Variable Documentation****5.20.2.1 double precision,dimension(:, :, :),allocatable drs_renderers::render_out**

Referenced by drs2dx(), drs_renderers_allocation(), render_Bp(), render_Br(), render_Bt(), render_Bz(), render_helicity(), render_poloidal_streamlines(), render_radial_streamfunction(), render_rotu_p(), render_rotu_r(), render_rotu_t(), render_rotu_z(), render_streamlines_t(), render_temperature(), render_temperature_perturbation(), render_temprature_grad_r(), render_up(), render_ur(), render_ut(), and render_uz().

5.20.2.2 double precision,allocatable drs_renderers::XX

Referenced by drs2dx(), drs_renderers_allocation(), render_B(), render_B_outside(), render_rotu(), and render_u().

5.20.2.3 double precision,allocatable drs_renderers::YY

Referenced by drs2dx(), drs_renderers_allocation(), render_B(), render_B_outside(), render_rotu(), and render_u().

5.20.2.4 double precision,allocatable drs_renderers::ZZ

Referenced by drs2dx(), drs_renderers_allocation(), render_B(), render_B_outside(), render_rotu(), and render_u().

5.21 drs_temp Namespace Reference

Temperature related operations.

Functions

- subroutine **drs_temp_allocation** ()
Allocates the temperature related variables.
- subroutine **drs_temp_init** ()
Precomputes the adimensional radial temperature profile.
- character(len=16) **tempProfName** ()
Outputs a human readable name for the temperature profiles.
- subroutine **drs_temp_reset** ()
Resets the temperature and its derivatives to zero.
- subroutine **update_temp_lap** ()
Recomputes and caches the laplacian of the temperature.
- subroutine **drs_temp_randomize** (noise)
- subroutine **apply_temp_BC_RHS** (val)
Boundary conditions for the temperature. For fixed temperature, the value of the anomaly is set to be zero.
- subroutine **calc_temp** (temp_t)
Computes the temperature anomaly in real space.

Variables

- double precision, dimension(:,:), allocatable **temp**
- double precision, dimension(:,:), allocatable **temp_dr**
- double precision, dimension(:,:), allocatable **temp_dds**
- double precision, dimension(:,:), allocatable **temp_lap**
- double precision, dimension(:,:), allocatable **temp_avg**
- double precision, dimension(:), allocatable **temp_dr_avg**
- double precision, dimension(:), allocatable **temp_profile**
- double precision, dimension(:), allocatable **temp_profile_dr**

5.21.1 Detailed Description

Temperature related operations.

5.21.2 Function Documentation

5.21.2.1 subroutine drs_temp::apply_temp_BC_RHS (double precision,dimension(nr),intent(inout) *val*)

Boundary conditions for the temperature. For fixed temperature, the value of the anomaly is set to be zero.

Todo

make this value depend on l and m when we can specify the full 2D anomaly at the boundaries.

Referenced by drs(), and update_temp().

5.21.2.2 subroutine drs_temp::calc_temp (double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *temp_t*)

Computes the temperature anomaly in real space.

References temp, and drs_transforms::ylmb().

Referenced by Benchmarkv1(), Benchmarkv2(), drs_nonlinear::evaluate_real_space(), and drs_renderers::render_temperature_perturbation().

Here is the call graph for this function:



5.21.2.3 subroutine drs_temp::drs_temp_allocation ()

Allocates the temperature related variables.

References drs_mpi::blk_ps_size, drs_mpi::mpi_rank, drs_dims::Nr, drs_dims::Nt_s, temp, temp_avg, temp_ddr, temp_dr, temp_dr_avg, temp_lap, temp_profile, and temp_profile_dr.

Referenced by drs_init(), and init().

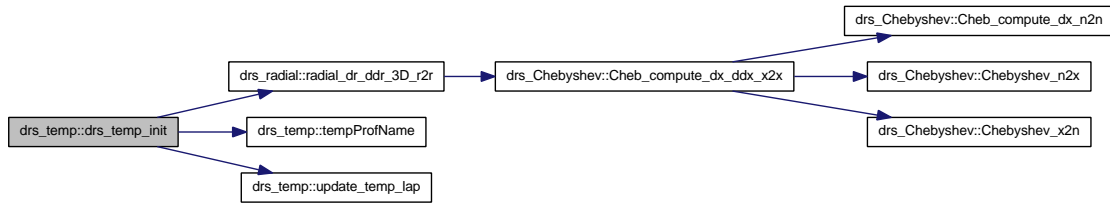
5.21.2.4 subroutine drs_temp::drs_temp_init ()

Precomputes the adimensional radial temperature profile.

References drs_dims::Nr, drs_radial::radial_dr_ddr_3D_r2r(), drs_radial::rcoll, drs_radial::rcoll2, temp, temp_ddr, temp_dr, temp_profile, temp_profile_dr, tempProfName(), and update_temp_lap().

Referenced by drs_init(), getProfile(), and init().

Here is the call graph for this function:



5.21.2.5 subroutine drs_temp::drs_temp_randomize (double precision,intent(in) *noise*)

References drs_mpi::blk_ps_start, drs_dims::m0, drs_mpi::mpi_rank, drs_dims::Np_s, drs_dims::Nr, drs_dims::Nt_s, drs_legendre::pi, drs_radial::rcoll, and temp.

5.21.2.6 subroutine drs_temp::drs_temp_reset ()

Resets the temperature and its derivatives to zero.

References temp, temp_ddr, and temp_dr.

5.21.2.7 character(len=16) drs_temp::tempProfName ()

Outputs a human readable name for the temperature profiles.

Since:

1.6.1

Referenced by drs_temp_init().

5.21.2.8 subroutine drs_temp::update_temp_lap ()

Recomputes and caches the laplacian of the temperature.

References drs_legendre::llp1, drs_dims::Nr, drs_radial::rcoll, drs_radial::rcoll2, temp, temp_ddr, temp_dr, and temp_lap.

Referenced by drs_temp_init(), and update_temp().

5.21.3 Variable Documentation

5.21.3.1 double precision,dimension(:, :, :),allocatable drs_temp::temp

Referenced by Benchmarkv1(), calc_temp(), drs(), drs_io::drs_load_state(), drs_temp_allocation(), drs_temp_init(), drs_temp_randomize(), drs_temp_reset(), getProfile(), drs_probes::measure_lm(), drs_renderers::render_temperature_grad_r(), drs_io::save_l_spec(), drs_io::save_m_spec(), drs_io::save_n_spec(), drs_io::save_state(), drs_nonlinear::save_stuff(), StateAverage(), update_temp(), and update_temp_lap().

5.21.3.2 double precision,dimension(:, :, :),allocatable drs_temp::temp_avg

Referenced by `drs_temp_allocation()`, `drs_io::dump_state()`, `drs_probes::measure_lm()`, and `StateAverage()`.

5.21.3.3 double precision,dimension(:, :, :),allocatable drs_temp::temp_ddr

Referenced by `drs()`, `drs_temp_allocation()`, `drs_temp_init()`, `drs_temp_reset()`, `update_temp()`, and `update_temp_lap()`.

5.21.3.4 double precision,dimension(:, :, :),allocatable drs_temp::temp_dr

Referenced by `drs()`, `drs_temp_allocation()`, `drs_temp_init()`, `drs_temp_reset()`, `drs_probes::measure_lm()`, `drs_probes::nusselt()`, `update_temp()`, and `update_temp_lap()`.

5.21.3.5 double precision,dimension(:),allocatable drs_temp::temp_dr_avg

Referenced by `drs_temp_allocation()`, `drs_io::dump_state()`, and `drs_probes::measure_lm()`.

5.21.3.6 double precision,dimension(:, :, :),allocatable drs_temp::temp_lap

Referenced by `drs_temp_allocation()`, `update_temp()`, and `update_temp_lap()`.

5.21.3.7 double precision,dimension(:),allocatable drs_temp::temp_profile

Referenced by `cacheTemperatureProfile()`, `drs_probes::compute_advection()`, `drs_temp_allocation()`, `drs_temp_init()`, `drs_io::dump_state()`, `getProfile()`, and `drs_renderers::render_temperature()`.

5.21.3.8 double precision,dimension(:),allocatable drs_temp::temp_profile_dr

Referenced by `drs_temp_allocation()`, `drs_temp_init()`, `drs_probes::nusselt()`, and `drs_nonlinear::save_stuff()`.

5.22 drs_time Namespace Reference

Module to deal with time. It deals with both wall time and simulation time. It also deals with the time-stepping.

Functions

- subroutine **drs_time_init** ()
• subroutine **drs_time_update** ()
Updates the current simulation time and step index.
- subroutine **update_timestep** (cfl, h, h_old, stat)
Updates the time-step according to the cfl numbers.
- subroutine **update_time_last_sample** (tnew)
Updates the time we last took a sample of some quantities.

Variables

- double precision **time_start**
The simulation time before the first step.
- double precision **time**
The simulation time.
- double precision **max_time**
The maximum simulation time.
- double precision **h**
The simulation time-step.
- double precision **h_old**
The simulation time step at the previous step.
- double precision **drift**
The drift rate in radians per simulation time unit.
- double precision **time_last_sample**
Time we last saved the probes values.
- double precision **time_since_last_sample**
Time since we last saved the probes values.
- logical **variable_h**
Do we have a variable time step?
- real **cpu_time_start**
Wall time when we started the run in seconds.

- real **cpu_time_now**
Wall time now, in seconds.
- real **cpu_time_first_step**
Wall time after we finished the first Newton step, in seconds.
- real **cpu_max_time**
The maximum wall time in hours.
- integer **transient**
How many steps to consider as a transient and discard?
- integer **sampling_rate**
The sampling rate in integer steps.
- integer **stepmax**
The maximum number of steps to take.
- integer **nsample**
How many samples we took so far.
- integer **steps**
Steps taken so far in total (includes stepstart).
- integer **stepstart**
The step count we start the run with.
- double precision, dimension(3), target **dtimestep**
- integer, dimension(5), target **imeasure**

5.22.1 Detailed Description

Module to deal with time. It deals with both wall time and simulation time. It also deals with the time-stepping.

5.22.2 Function Documentation

5.22.2.1 subroutine drs_time::drs_time_init ()

References drift, dtimestep, h, imeasure, max_time, nsample, sampling_rate, stepmax, steps, time, and transient.

Referenced by drs_init(), and init().

5.22.2.2 subroutine drs_time::drs_time_update ()

Updates the current simulation time and step index.

References h, steps, and time.

Referenced by drs().

5.22.2.3 subroutine drs_time::update_time_last_sample (double precision *tnew*)

Updates the time we last took a sample of some quantities.

References time_last_sample.

Referenced by drs_probes::drs_probes_init(), and drs_nonlinear::save_stuff().

5.22.2.4 subroutine drs_time::update_timestep (double precision,dimension(:),intent(in) *cfl*, double precision,intent(inout) *h*, double precision,intent(out) *h_old*, integer,intent(out) *stat*)

Updates the time-step according to the cfl numbers.

References variable_h.

Referenced by drs_nonlinear::rhs().

5.22.3 Variable Documentation

5.22.3.1 real drs_time::cpu_max_time

The maximum wall time in hours.

Referenced by drs_init(), drs_io_par::drs_read_conf(), drs_io_par::drs_read_conf_v2(), and need_to_step().

5.22.3.2 real drs_time::cpu_time_first_step

Wall time after we finished the first Newton step, in seconds.

Referenced by drs().

5.22.3.3 real drs_time::cpu_time_now

Wall time now, in seconds.

Referenced by drs().

5.22.3.4 real drs_time::cpu_time_start

Wall time when we started the run in seconds.

Referenced by drs_init(), and need_to_step().

5.22.3.5 double precision drs_time::drift

The drift rate in radians per simulation time unit.

Referenced by drs_init(), drs_time_init(), init(), and drs_io_par::write_parp().

5.22.3.6 double precision,dimension(3),target drs_time::dtimestep

Referenced by `drs_init()`, and `drs_time_init()`.

5.22.3.7 double precision drs_time::h

The simulation time-step.

Referenced by `drs()`, `drs_init()`, `drs_io_par::drs_read_conf()`, `drs_io_par::drs_read_conf_v2()`, `drs_time_init()`, `drs_time_update()`, and `drs_io_par::write_parp()`.

5.22.3.8 double precision drs_time::h_old

The simulation time step at the previous step.

Referenced by `drs()`, and `drs_init()`.

5.22.3.9 integer,dimension(5),target drs_time::imeasure

Referenced by `drs_init()`, and `drs_time_init()`.

5.22.3.10 double precision drs_time::max_time

The maximum simulation time.

Referenced by `drs_time_init()`, and `need_to_step()`.

5.22.3.11 integer drs_time::nsample

How many samples we took so far.

Referenced by `drs_init()`, `drs_time_init()`, `drs_io::dump_state()`, and `drs_nonlinear::rhs()`.

5.22.3.12 integer drs_time::sampling_rate

The sampling rate in integer steps.

Referenced by `drs()`, `drs_init()`, `drs_io_par::drs_read_conf()`, `drs_io_par::drs_read_conf_v2()`, `drs_time_init()`, `drs_nonlinear::rhs()`, and `drs_io_par::write_parp()`.

5.22.3.13 integer drs_time::stepmax

The maximum number of steps to take.

Referenced by `drs_init()`, `drs_io_par::drs_read_conf()`, `drs_io_par::drs_read_conf_v2()`, `drs_time_init()`, `need_to_step()`, and `drs_io_par::write_parp()`.

5.22.3.14 integer drs_time::steps

Steps taken so far in total (includes stepstart).

Referenced by `drs()`, `drs_init()`, `drs_io::drs_load_state()`, `drs_time_init()`, `drs_time_update()`, `drs_io::dump_state()`, `kd_grothrate()`, `drs_probes::measure()`, `need_to_step()`, `drs_nonlinear::rhs()`, `update_flow()`, and `drs_io_par::write_parp()`.

5.22.3.15 integer `drs_time::stepstart`

The step count we start the run with.

Referenced by `drs()`, `drs_init()`, `drs_io::drs_load_state()`, `need_to_step()`, and `drs_io_par::read_input_parp()`.

5.22.3.16 double precision `drs_time::time`

The simulation time.

Referenced by `Benchmarkv1()`, `Benchmarkv2()`, `drs()`, `drs_init()`, `drs_io::drs_load_state()`, `drs_time_init()`, `drs_time_update()`, `drs_io::dump_state()`, `init()`, `drs_probes::measure()`, `need_to_step()`, `drs_io_par::read_input_parp()`, `drs_probes::save_angular_momentum()`, `drs_probes::save_field_coeffs()`, `drs_probes::save_flow_coeffs()`, `drs_probes::save_flow_dissipation()`, `drs_probes::save_magnetic_dissipation()`, `drs_nonlinear::save_stuff()`, and `drs_io_par::write_parp()`.

5.22.3.17 double precision `drs_time::time_last_sample`

Time we last saved the probes values.

Referenced by `drs_nonlinear::save_stuff()`, and `update_time_last_sample()`.

5.22.3.18 double precision `drs_time::time_since_last_sample`

Time since we last saved the probes values.

Referenced by `drs_probes::average_unnormalised_field_l_spectrum()`, `drs_probes::average_unnormalised_flow_l_spectrum()`, `drs_probes::average_unnormalised_scalar_l_spectrum()`, `drs_probes::measure()`, `drs_probes::measure_lm()`, and `drs_nonlinear::save_stuff()`.

5.22.3.19 double precision `drs_time::time_start`

The simulation time before the first step.

Referenced by `drs_io::drs_load_state()`, and `drs_io::dump_state()`.

5.22.3.20 integer `drs_time::transient`

How many steps to consider as a transient and discard?

Referenced by `drs_init()`, `drs_io_par::drs_read_conf()`, `drs_io_par::drs_read_conf_v2()`, `drs_time_init()`, `drs_nonlinear::rhs()`, and `drs_io_par::write_parp()`.

5.22.3.21 logical `drs_time::variable_h`

Do we have a variable time step?

Referenced by `drs()`, `drs_init()`, and `update_timestep()`.

5.23 drs_transforms Namespace Reference

Spherical transforms.

Functions

- subroutine **ylmt** (input, output)

This routine performs three steps: ~~~~~ 1. transformation $tt_t(\theta, \phi) \rightarrow tt_t(\theta, m)$ transposed. 2. redistribution $dist(\theta) \rightarrow dist(m)$ 3. transformation $tt(\theta, m) \rightarrow t(l, m)$ distrib. in m .

- subroutine **ylmt_3D** (input, output)

This routine performs three steps: ~~~~~ 1. transformation $tt_t(\theta, \phi) \rightarrow tt_t(\theta, m)$ transposed. 2. redistribution $dist(\theta) \rightarrow dist(m)$ 3. transformation $tt(\theta, m) \rightarrow t(l, m)$ distrib. in m .

- subroutine **ylmb** (input, output)
- subroutine **m2phi_2D** (input, output)
- subroutine **my_div** (vec_r, vec_t, vec_p, nonlin)

*Computes the spectral coefficients of the divergence of the vector field \vec{u} .
On input:.*

- subroutine **my_rot** (vec_t, vec_p, nonlin)
- subroutine **my_rotrot** (vec_r, vec_t, vec_p, nonlin)
- subroutine **vectorField2PolTor_common** (vr, vt, vp, pol, tor)

*Computes the poloidal and toroidal scalar coefficients of a 3D vector field.
This is the part of the calculation that is common to both the flow and magnetic field.*

- subroutine **PolTor_common2PolTor_flow** (pol, tor)

Multiplies the poloidal and toroidal scalar coefficients by the appropriate factors of r and $l(l+1)$ to make them the poloidal and toroidal scalars of the flow as defined for this program.*

- subroutine **PolTor_common2PolTor_field** (pol, tor)

Multiplies the poloidal and toroidal scalar coefficients by the appropriate factors of r and $l(l+1)$ to make them the poloidal and toroidal scalars of the magnetic field as defined for this program.*

- subroutine **vectorField2Divergence** (ur, ut, up, div)

Computes the spherical harmonic coefficients of the divergence of a 3D vector field in real space.

Variables

- double precision, parameter **pi** = 3.141592653589793d0

5.23.1 Detailed Description

Spherical transforms.

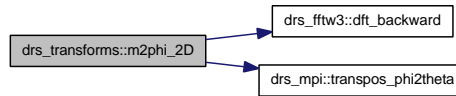
5.23.2 Function Documentation

5.23.2.1 subroutine `drs_transforms::m2phi_2D` (double precision,dimension(0:nt,blk_ps_size(mpi_rank),intent(in) *input*, double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *output*)

References `drs_fftw3::dft_backward()`, `drs_dims::Np_s`, and `drs_mpi::transpos_phi2theta()`.

Referenced by `drs_field::calc_field()`, `drs_flow::calc_flow()`, `drs_field::calc_rot_field()`, `drs_flow::calc_rot_flow()`, and `ylmb()`.

Here is the call graph for this function:



5.23.2.2 subroutine `drs_transforms::my_div` (double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(in) *vec_r*, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(in) *vec_t*, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(in) *vec_p*, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(out) *nonlin*)

Computes the spectral coefficients of the divergence of the vector field \vec{u} .

On input:.

Parameters:

vec_r is $u_r * r^2$ in (l,m,r) space.

vec_t is $\frac{u_\theta}{r \sin \theta}$ in (l,m,r) space.

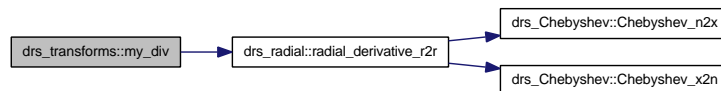
vec_p is $\frac{u_\phi}{r \sin \theta}$ in (l,m,r) space. On output:

nonlin is the divergence in (l,m,r) space.

References `drs_mpi::blk_ps_start`, `drs_mpi::blk_ts_start`, `drs_radial::radial_derivative_r2r()`, and `drs_radial::rcoll2`.

Referenced by `vectorField2Divergence()`.

Here is the call graph for this function:



5.23.2.3 subroutine `drs_transforms::my_rot` (double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(in) *vec_t*, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(in) *vec_p*, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(out) *nonlin*)

References `drs_mpi::blk_ps_start`, and `drs_mpi::blk_ts_start`.

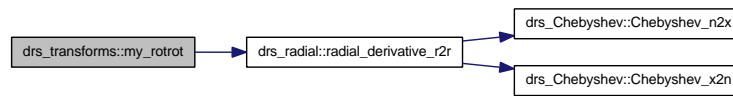
Referenced by `vectorField2PolTor_common()`.

5.23.2.4 subroutine `drs_transforms::my_rotrot` (`double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(in) vec_r, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(in) vec_t, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(in) vec_p, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(out) nonlin)`

References `drs_mpi::blk_ps_start`, `drs_mpi::blk_ts_start`, `drs_legendre::llp1`, `drs_radial::radial_derivative_r2r()`, and `drs_radial::rcoll2`.

Referenced by `vectorField2PolTor_common()`.

Here is the call graph for this function:



5.23.2.5 subroutine `drs_transforms::PolTor_common2PolTor_field` (`double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(inout) pol, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(inout) tor)`

Multiplies the poloidal and toroidal scalar coefficients by the appropriate factors of r and $l*(l+1)$ to make them the poloidal and toroidal scalars of the magnetic field as defined for this program.

References `drs_legendre::llp1`, and `drs_radial::rcoll2`.

Referenced by `drs_nonlinear::save_stuff()`.

5.23.2.6 subroutine `drs_transforms::PolTor_common2PolTor_flow` (`double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(inout) pol, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(inout) tor)`

Multiplies the poloidal and toroidal scalar coefficients by the appropriate factors of r and $l*(l+1)$ to make them the poloidal and toroidal scalars of the flow as defined for this program.

References `drs_legendre::llp1`, and `drs_radial::rcoll`.

Referenced by `drs_nonlinear::save_stuff()`.

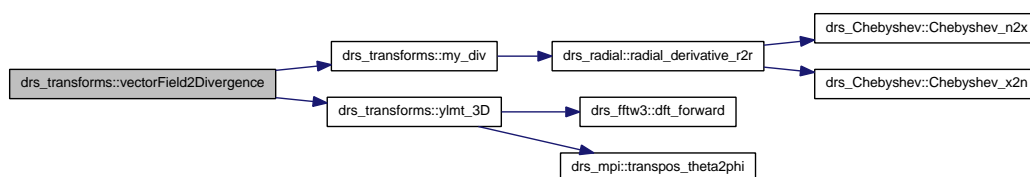
5.23.2.7 subroutine `drs_transforms::vectorField2Divergence` (`double precision,dimension(0:blk_t_size(mpi_rank),intent(inout) ur, double precision,dimension(0:blk_t_size(mpi_rank),intent(inout) ut, double precision,dimension(0:blk_t_size(mpi_rank),intent(inout) up, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(out) div)`

Computes the spherical harmonic coefficients of the divergence of a 3D vector field in real space.

References `drs_mpi::blk_t_start`, `my_div()`, `drs_radial::rcoll`, `drs_radial::rcoll2`, `drs_legendre::sintheta`, and `ylmt_3D()`.

Referenced by `drs_nonlinear::save_stuff()`, and `test_vectorField2Divergence()`.

Here is the call graph for this function:



5.23.2.8 subroutine drs_transforms::vectorField2PolTor_common (double precision,dimension(0:blk_t_size(mpi_rank),intent(inout) *vr*, double precision,dimension(0:blk_t_size(mpi_rank),intent(inout) *vt*, double precision,dimension(0:blk_t_size(mpi_rank),intent(inout) *vp*, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(inout) *pol*, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(inout) *tor*)

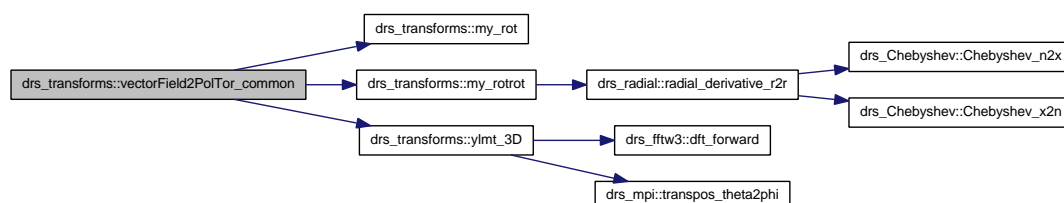
Computes the poloidal and toroidal scalar coefficients of a 3D vector field.

This is the part of the calculation that is common to both the flow and magnetic field.

References drs_mpi::blk_t_start, my_rot(), my_rotrot(), drs_radial::rcoll, drs_radial::rcoll2, drs_legendre::sintheta, and ylmt_3D().

Referenced by drs_nonlinear::save_stuff().

Here is the call graph for this function:



5.23.2.9 subroutine drs_transforms::ylmb (double precision,dimension(0:nt_s,1:blk_ps_size(mpi_rank),intent(in) *input*, double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *output*)

References drs_legendre::legendre, m2phi_2D(), and drs_mpi::mm.

Referenced by drs_temp::calc_temp(), drs_renderers::render_poloidal_streamlines(), drs_renderers::render_radial_streamfunction(), and drs_renderers::render_temperature_grad_r().

Here is the call graph for this function:



5.23.2.10 subroutine `drs_transforms::ylmt` (double precision,dimension(0:blk_t_size(mpi_rank),intent(in) *input*, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(out) *output*)

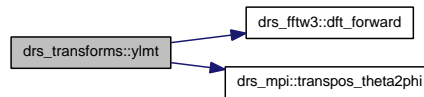
This routine performs three steps: ~~~~~ 1. transformation `tt_t(theta,phi) --> tt_t(theta,m)` transposed. 2. redistribution `dist(theta) --> dist(m)` 3. transformation `tt(theta,m) --> t(l,m)` distrib. in m. input: `tt_t(0:(blk_t_max_size-1),Np)` (transposed) has to be in direct space, in phi-direction `t(:,1:Np)` only one period is stored: `tt_t(:,j)` means position $\phi = (2\pi/m_0) * (j-1)/Np$.

output: `t(0:drs_Nt,drs_Np)` dims: `(0:Nt_s,Np_s)` dealiased in (lm)! in m-dir.: `real(m=0),real(m=m0),im(m=m0),real(m=2m0),im(m=2*m0),...` in l-dir.: includes normalization factors!

~~~~~

References `drs_fftw3::dft_forward()`, `drs_legendre::leg_neg`, `drs_mpi::mm`, `drs_dims::Np_s`, and `drs_mpi::transpos_theta2phi()`.

Here is the call graph for this function:



### 5.23.2.11 subroutine `drs_transforms::ylmt_3D` (double precision,dimension(0:blk\_t\_size(mpi\_rank),intent(in) *input*, double precision,dimension(0:nt\_s,blk\_ps\_size(mpi\_rank),intent(out) *output*)

This routine performs three steps: ~~~~~ 1. transformation `tt_t(theta,phi) --> tt_t(theta,m)` transposed. 2. redistribution `dist(theta) --> dist(m)` 3. transformation `tt(theta,m) --> t(l,m)` distrib. in m. input: `tt_t(0:(blk_t_max_size-1),Np)` (transposed) has to be in direct space, in phi-direction `t(:,1:Np)` only one period is stored: `tt_t(:,j)` means position  $\phi = (2\pi/m_0) * (j-1)/Np$ .

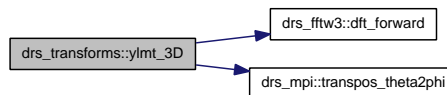
output: `t(0:drs_Nt,drs_Np)` dims: `(0:Nt_s,Np_s)` dealiased in (lm)! in m-dir.: `real(m=0),real(m=m0),im(m=m0),real(m=2m0),im(m=2*m0),...` in l-dir.: includes normalization factors!

~~~~~

References `drs_fftw3::dft_forward()`, `drs_legendre::leg_neg`, `drs_mpi::mm`, `drs_dims::Np_s`, and `drs_mpi::transpos_theta2phi()`.

Referenced by `vectorField2Divergence()`, and `vectorField2PolTor_common()`.

Here is the call graph for this function:



5.23.3 Variable Documentation

5.23.3.1 double precision,parameter `drs_transforms::pi = 3.141592653589793d0`

5.24 parser Namespace Reference

This module provides a simple **parser** (p. 123) for input files of the type 'key = val' eventually separated by '[Sections]'.

Classes

- struct **parser_vars**
A description of the keys we will find and their properties.
- struct **parser_section**
A description of the sections we will find and their properties.
- interface **read_val**
Reads a value.

Functions

- subroutine **parse** (unit_in, variable, line, error)
Reads one line from the specified unit and outputs the variable name and possible values as a string.

5.24.1 Detailed Description

This module provides a simple **parser** (p. 123) for input files of the type 'key = val' eventually separated by '[Sections]'. Typical use of this module goes like: ~~~~~~ open(unit=444, file='myconfig.conf', status='old', iostat=error) if (error.ne.0) return do call parse(444, varname, line, error) select case(varname) case('var1') call read_val(line, var1) case('var2') call read_val(line, var2) case default cycle end select enddo close(444) ~~~~~~

5.24.2 Function Documentation

5.24.2.1 subroutine **parser::parse** (integer,intent(in) *unit_in*, character(len=*),intent(out) *variable*, character(len=256),intent(out) *line*, integer,intent(out) *error*)

Reads one line from the specified unit and outputs the variable name and possible values as a string.

variable is the variable name.

line is the rhs of the attribution.

unit_in is the unit to read from

error is an error code.

Referenced by `drs_io_par::drs_read_conf_v2()`, `parse_drs2dx()`, and `parseConfig()`.

Chapter 6

Class Documentation

6.1 drs_mpi::drs_bcast Interface Reference

Encapsulates broadcast of several types and ranks.

Public Member Functions

- subroutine **drs_bcast_int** (array, num)
- subroutine **drs_bcast_dble** (array, num)
- subroutine **drs_bcast_int_scal** (val)
- subroutine **drs_bcast_dble_scal** (val)
- subroutine **drs_bcast_logical_scal** (val)

6.1.1 Detailed Description

Encapsulates broadcast of several types and ranks.

6.1.2 Member Function Documentation

6.1.2.1 subroutine **drs_mpi::drs_bcast::drs_bcast_dble** (double precision,dimension(:),intent(inout) *array*, integer,intent(in) *num*)

6.1.2.2 subroutine **drs_mpi::drs_bcast::drs_bcast_dble_scal** (double precision,intent(inout) *val*)

6.1.2.3 subroutine **drs_mpi::drs_bcast::drs_bcast_int** (integer,dimension(:),intent(inout) *array*, integer,intent(in) *num*)

6.1.2.4 subroutine **drs_mpi::drs_bcast::drs_bcast_int_scal** (integer,intent(inout) *val*)

6.1.2.5 subroutine **drs_mpi::drs_bcast::drs_bcast_logical_scal** (logical,intent(inout) *val*)

The documentation for this interface was generated from the following file:

- **drs_mpi.f90**

6.2 drs_mpi::drs_maximize Interface Reference

Encapsulates maximization of several types and ranks.

Public Member Functions

- subroutine **drs_maximize_dble** (array)
- subroutine **drs_maximize_dble_scal** (val)

6.2.1 Detailed Description

Encapsulates maximization of several types and ranks.

6.2.2 Member Function Documentation

6.2.2.1 subroutine **drs_mpi::drs_maximize::drs_maximize_dble** (double precision,dimension(:),intent(inout) *array*)

6.2.2.2 subroutine **drs_mpi::drs_maximize::drs_maximize_dble_scal** (double precision,intent(inout) *val*)

The documentation for this interface was generated from the following file:

- **drs_mpi.f90**

6.3 drs_mpi::drs_minimize Interface Reference

Encapsulates minimization of several types and ranks.

Public Member Functions

- subroutine **drs_minimize_dble** (array)
- subroutine **drs_minimize_dble_scal** (val)

6.3.1 Detailed Description

Encapsulates minimization of several types and ranks.

6.3.2 Member Function Documentation

6.3.2.1 subroutine **drs_mpi::drs_minimize::drs_minimize_dble** (double precision,dimension(:),intent(inout) *array*)

6.3.2.2 subroutine **drs_mpi::drs_minimize::drs_minimize_dble_scal** (double precision,intent(inout) *val*)

The documentation for this interface was generated from the following file:

- **drs_mpi.f90**

6.4 drs_legendre::interface Interface Reference

Public Member Functions

- subroutine **PlmBar_d1** (*p*, *dp*, *lmax*, *z*, *csphase*, *cnorm*)
- subroutine **PLegendreA_d1** (*p*, *dp*, *lmax*, *z*, *csphase*)
- integer **PlmIndex** (*l*, *m*)

6.4.1 Member Function Documentation

6.4.1.1 subroutine `drs_legendre::interface::PLegendreA_d1` (*real*8,dimension(:),intent(out)* *p*, *real*8,dimension(:),intent(out)* *dp*, *integer,intent(in)* *lmax*, *real*8,intent(in)* *z*, *integer,intent(in),optional* *csphase*)

6.4.1.2 subroutine `drs_legendre::interface::PlmBar_d1` (*real*8,dimension(:),intent(out)* *p*, *real*8,dimension(:),intent(out)* *dp*, *integer,intent(in)* *lmax*, *real*8,intent(in)* *z*, *integer,intent(in),optional* *csphase*, *integer,intent(in),optional* *cnorm*)

6.4.1.3 integer `drs_legendre::interface::PlmIndex` (*integer,intent(in)* *l*, *integer,intent(in)* *m*)

The documentation for this interface was generated from the following file:

- `drs_legendre.f90`

6.5 drs_io_DX::save2DX Interface Reference

Public Member Functions

- subroutine **save2DXscalar** (field, filename)
- subroutine **save2DXvector** (XX, YY, ZZ, filename)

6.5.1 Member Function Documentation

6.5.1.1 subroutine **drs_io_DX::save2DX::save2DXscalar** (double precision,dimension(0:(blk_t_size(mpi_rank),intent(in) *field*, character(len=*),intent(in) *filename*)

6.5.1.2 subroutine **drs_io_DX::save2DX::save2DXvector** (double precision,dimension(0:(blk_t_size(mpi_rank),intent(in) *XX*, double precision,dimension(0:(blk_t_size(mpi_rank),intent(in) *YY*, double precision,dimension(0:(blk_t_size(mpi_rank),intent(in) *ZZ*, character(len=*),intent(in) *filename*)

The documentation for this interface was generated from the following file:

- utilities/**drs_io_DX.f90**

6.6 drs_mpi::sum_over_all_cpus Interface Reference

Encapsulates sums of several types and ranks.

Public Member Functions

- subroutine **sum_over_all_cpus_scal** (*val*)
- subroutine **sum_over_all_cpus_vect** (*val*)

6.6.1 Detailed Description

Encapsulates sums of several types and ranks.

6.6.2 Member Function Documentation

6.6.2.1 subroutine **drs_mpi::sum_over_all_cpus::sum_over_all_cpus_scal** (double precision,intent(inout) *val*)

6.6.2.2 subroutine **drs_mpi::sum_over_all_cpus::sum_over_all_cpus_vect** (double precision,dimension(:),intent(inout) *val*)

The documentation for this interface was generated from the following file:

- **drs_mpi.f90**

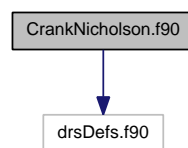
Chapter 7

File Documentation

7.1 CrankNicholson.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for CrankNicholson.f90:



Namespaces

- namespace **CrankNicholson**

Provides routines that compute the Crank-Nicholson inverse operators and variables to store them.

Functions

- subroutine **CrankNicholson::CrankNicholson_init** ()
Allocates memory for the Crank-Nicholson inverse operators.
- subroutine **CrankNicholson::updateCrankNicholson_matrices** (h, Pt, Pm, Pc)
Convenience subroutine that generates all of the Crank-Nicholson inverse operators.

Variables

- double precision, dimension(:, :, :), allocatable **CrankNicholson::field_lap_inv_tor**
The Crank-Nicholson inverse operator for the toroidal flow.
- double precision, dimension(:, :, :), allocatable **CrankNicholson::field_lap_inv_pol**

The Crank-Nicholson inverse operator for the poloidal flow.

- double precision, dimension(:, :, :), allocatable **CrankNicholson::flow_lap_inv_tor**

The Crank-Nicholson inverse operator for the toroidal field.

- double precision, dimension(:, :, :), allocatable **CrankNicholson::flow_lap_inv_pol**

The Crank-Nicholson inverse operator for the poloidal field.

- double precision, dimension(:, :, :), allocatable **CrankNicholson::temp_lap_inv**

The Crank-Nicholson inverse operator for the temperature.

- double precision, dimension(:, :, :), allocatable, target **CrankNicholson::pinv**

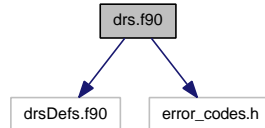
The inverse laplacian operator in real space.

7.2 drs.f90 File Reference

```
#include "drsDefs.f90"
```

```
#include "error_codes.h"
```

Include dependency graph for drs.f90:



Functions

- program **drs**
- subroutine **drs_init** (error)
 - Initialize quantities and modules necessary for the program to run.*
- subroutine **update_Green_functions** ()
 - Encapsulates dealing with the Green's functions.*
- logical **need_to_step** ()
 - This function will determine whether we need to take another time-step or not. There are several stopping conditions:.*
- subroutine **mk_green** (ib, ob, green, greenD, greenS)
 - a stopping condition on number of steps;
- subroutine **applyGreen** ()
 - Apply Green's functions.*
- subroutine **kd_grothrate** ()
 - Computes the grothrate of the kinamatic dynamo.*
- subroutine **update_flow** (h, h1, h2)
 - Updates the flow using a hybrid Crank-Nicholson/Addams-Bashford implicit integration scheme.*
- subroutine **update_temp** (h, h1, h2, Pt)
 - Updates the temperature using a hybrid Crank-Nicholson/Addams-Bashford implicit integration scheme.*
- subroutine **update_field** (h, h1, h2, Pm)
 - Updates the magnetic field using a hybrid Crank-Nicholson/Addams-Bashford implicit integration scheme.*

7.2.1 Function Documentation

7.2.1.1 subroutine drs::applyGreen ()

Apply Green's functions.

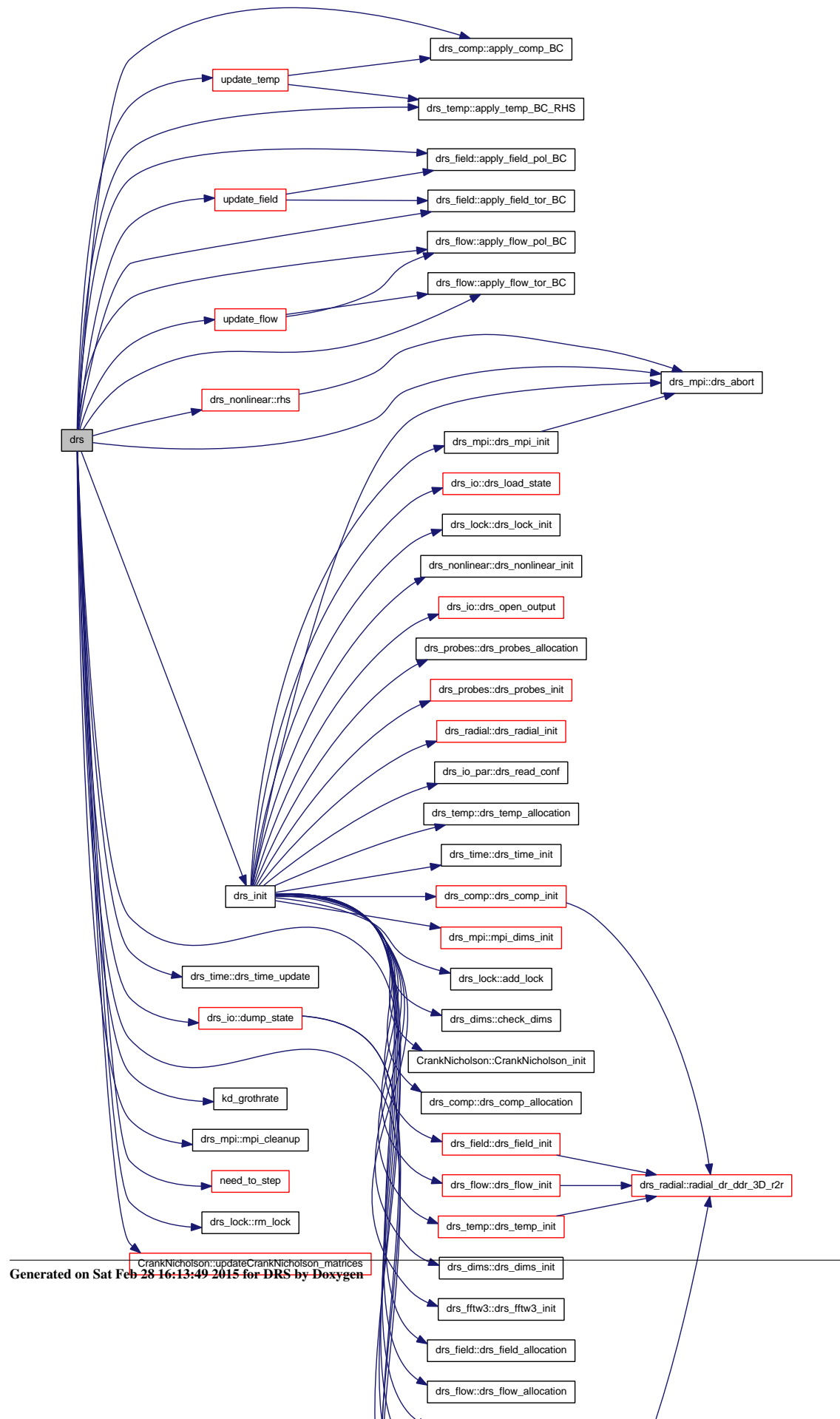
References `drs_flow::flow_pol`, `drs_flow::flow_pol_ddr`, `drs_flow::flow_pol_dr`, and `drs_dims::Nr`.

Referenced by `update_flow()`.

7.2.1.2 program `drs ()`

References `drs_comp::apply_comp_BC()`, `drs_field::apply_field_pol_BC()`, `drs_field::apply_field_tor_BC()`, `drs_flow::apply_flow_pol_BC()`, `drs_flow::apply_flow_tor_BC()`, `drs_temp::apply_temp_BC_RHS()`, `drs_comp::comp`, `drs_comp::comp_ddr`, `drs_comp::comp_dr`, `drs_time::cpu_time_first_step`, `drs_time::cpu_time_now`, `drs_mpi::drs_abort()`, `drs_init()`, `drs_time::drs_time_update()`, `drs_io::dump_state()`, `drs_nonlinear::evaluate_real_space()`, `drs_field::field_pol`, `drs_field::field_pol_ddr`, `drs_field::field_pol_dr`, `drs_field::field_tor`, `drs_field::field_tor_ddr`, `drs_field::field_tor_dr`, `drs_flow::flow_pol`, `drs_flow::flow_pol_ddr`, `drs_flow::flow_pol_dr`, `drs_flow::flow_tor`, `drs_flow::flow_tor_ddr`, `drs_flow::flow_tor_dr`, `drs_time::h`, `drs_time::h_old`, `kd_grothrate()`, `drs_mpi::mm`, `drs_mpi::mpi_cleanup()`, `drs_mpi::mpi_rank`, `need_to_step()`, `drs_dims::Nr`, `drs_radial::radial_dr_ddr_3D_r2r()`, `drs_nonlinear::rhs()`, `drs_lock::rm_lock()`, `drs_time::sampling_rate`, `drs_time::steps`, `drs_time::stepstart`, `drs_temp::temp`, `drs_temp::temp_ddr`, `drs_temp::temp_dr`, `drs_time::time`, `update_field()`, `update_flow()`, `update_temp()`, `CrankNicholson::updateCrankNicholson_matrices()`, and `drs_time::variable_h`.

Here is the call graph for this function:



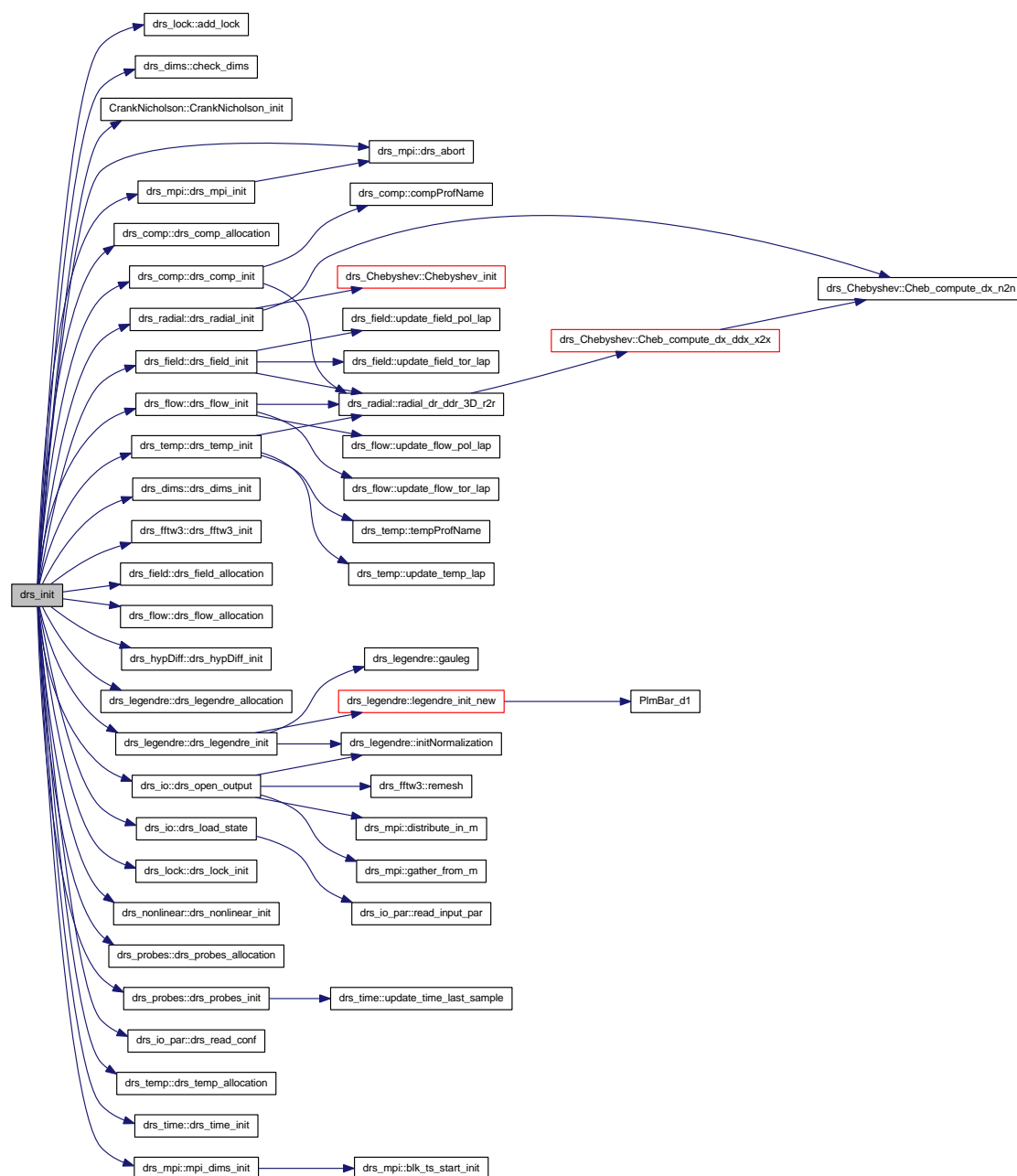
7.2.1.3 subroutine drs::drs_init (integer,intent(inout) error)

Initialize quantities and modules necessary for the program to run.

References drs_lock::add_lock(), drs_mpi::blk_ps_size, drs_mpi::blk_ps_start, drs_-
mpi::blk_t_size, drs_dims::check_dims(), drs_time::cpu_max_time, drs_time::cpu_time_start,
CrankNicholson::CrankNicholson_init(), drs_time::drift, drs_mpi::drs_abort(), drs_comp::drs_-
comp_allocation(), drs_comp::drs_comp_init(), drs_dims::drs_dims_init(), drs_fftw3::drs_fftw3_-
init(), drs_field::drs_field_allocation(), drs_field::drs_field_init(), drs_flow::drs_flow_allocation(),
drs_flow::drs_flow_init(), drs_hypDiff::drs_hypDiff_init(), drs_legendre::drs_legendre_allocation(), drs_-
legendre::drs_legendre_init(), drs_io::drs_load_state(), drs_lock::drs_lock_init(), drs_mpi::drs_mpi_init(),
drs_nonlinear::drs_nonlinear_init(), drs_io::drs_open_output(), drs_probes::drs_probes_allocation(),
drs_probes::drs_probes_init(), drs_radial::drs_radial_init(), drs_io_par::drs_read_conf(), drs_temp::drs_-
temp_allocation(), drs_temp::drs_temp_init(), drs_time::drs_time_init(), drs_hypDiff::drs_want_-
hypDiff, drs_time::dtimestep, drs_field::field_pol_ddr, drs_field::field_pol_dr, drs_field::field_tor_ddr,
drs_field::field_tor_dr, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, drs_flow::flow_tor_ddr, drs_-
flow::flow_tor_dr, drs_time::h, drs_time::h_old, drs_io_par::hi, drs_time::imeasure, drs_io::io_calc_file_-
in, drs_io::io_calc_file_out, drs_dims::lsymm, drs_dims::m0, drs_mpi::mpi_dims_init(), drs_mpi::mpi_-
rank, drs_dims::Np, drs_dims::Np_s, drs_dims::Nr, drs_dims::Nr_s, drs_time::nsample, drs_dims::Nt,
drs_dims::Nt_s, drs_legendre::pi, drs_time::sampling_rate, drs_time::stepmax, drs_time::steps, drs_-
time::stepstart, drs_time::time, drs_time::transient, drs_dims::usr_dims, and drs_time::variable_h.

Referenced by drs().

Here is the call graph for this function:



7.2.1.4 subroutine drs::kd_grothrate ()

Computes the grothrate of the kinamatic dynamo.

References `drs_mpi::blk_ps_size`, `drs_mpi::blk_ps_start`, `drs_field::field_pol_ddr`, `drs_field::field_tor`, `drs_field::field_tor_ddr`, `drs_legendre::llp1`, `drs_dims::m0`, `drs_mpi::mpi_rank`, `drs_dims::Np_s`, `drs_dims::Nr`, `drs_dims::Nt_s`, `drs_radial::rcoll`, `drs_nonlinear::rhs_IE_pol`, `drs_nonlinear::rhs_IE_tor`, and `drs_time::steps`.

Referenced by drs().

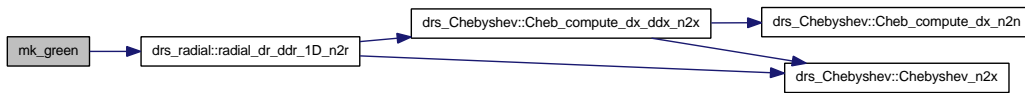
7.2.1.5 subroutine drs::mk_green (double precision,intent(in) *ib*, double precision,intent(in) *ob*, double precision,dimension(nr, 0:nt_s),intent(out) *green*, double precision,dimension(nr, 0:nt_s),intent(out) *greenD*, double precision,dimension(nr, 0:nt_s),intent(out) *greenS*)

- a stopping condition on number of steps;
- a stopping condition on cpu time; - a stopping condition on the existence of a lock file; - a stopping condition on simulation time; Generates the appropriate Green's function for a given boundary value.

References CrankNicholson::flow_lap_inv_pol, CrankNicholson::pinv, and drs_radial::radial_dr_ddr_1D_n2r().

Referenced by update_Green_functions().

Here is the call graph for this function:



7.2.1.6 logical drs::need_to_step ()

This function will determine whether we need to take another time-step or not. There are several stopping conditions:.

References drs_time::cpu_max_time, drs_time::cpu_time_start, drs_lock::lockExists(), drs_time::max_time, drs_mpi::mpi_rank, drs_time::stepmax, drs_time::steps, drs_time::stepstart, drs_time::time, and drs_mpi::wait_for_everyone().

Referenced by drs().

Here is the call graph for this function:



7.2.1.7 subroutine drs::update_field (double precision,intent(in) *h*, double precision,intent(in) *h1*, double precision,intent(in) *h2*, double precision,intent(in) *Pm*)

Updates the magnetic field using a hybrid Crank-Nicholson/Addams-Bashford implicit integration scheme.

Parameters:

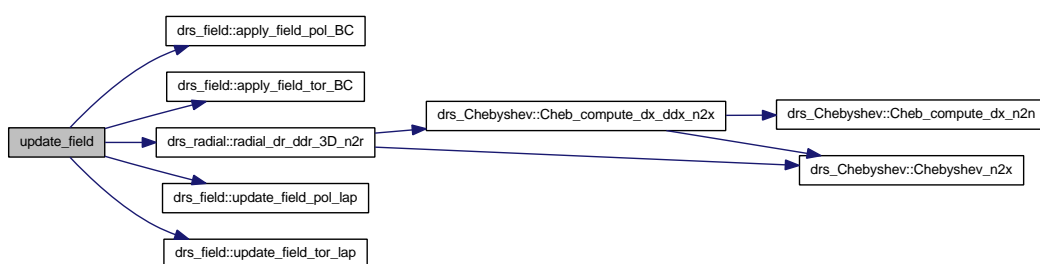
- h* is the time step;
- h1* and
- h2* are the Addams-Bashford weights;

Pm is the magnetic Prandtl number;

References `drs_field::apply_field_pol_BC()`, `drs_field::apply_field_tor_BC()`, `CrankNicholson::field_lap_inv_pol`, `CrankNicholson::field_lap_inv_tor`, `drs_field::field_pol`, `drs_field::field_pol_ddr`, `drs_field::field_pol_dr`, `drs_field::field_pol_lap`, `drs_field::field_tor`, `drs_field::field_tor_ddr`, `drs_field::field_tor_dr`, `drs_field::field_tor_lap`, `drs_mpi::mm`, `drs_dims::Nr`, `drs_radial::radial_dr_ddr_3D_n2r()`, `drs_nonlinear::rhs_IE_pol`, `drs_nonlinear::rhs_IE_tor`, `drs_field::update_field_pol_lap()`, and `drs_field::update_field_tor_lap()`.

Referenced by `drs()`.

Here is the call graph for this function:



7.2.1.8 subroutine `drs::update_flow` (double precision,intent(in) h , double precision,intent(in) $h1$, double precision,intent(in) $h2$)

Updates the flow using a hybrid Crank-Nicholson/Addams-Bashford implicit integration scheme.

Parameters:

h is the time step;

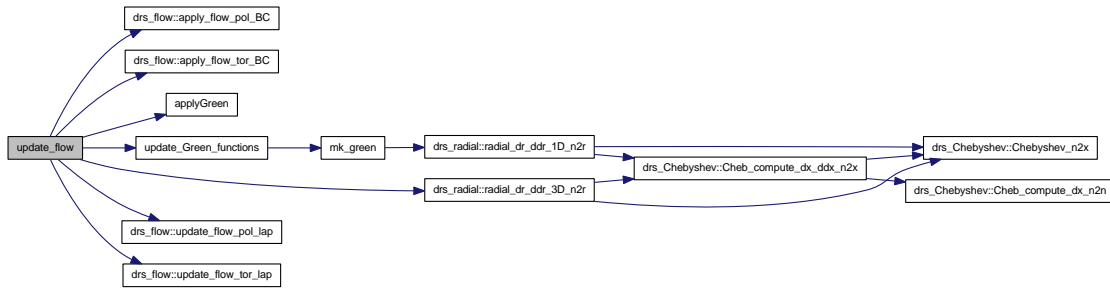
$h1$ and

$h2$ are the Addams-Bashford weights

References `drs_flow::apply_flow_pol_BC()`, `drs_flow::apply_flow_tor_BC()`, `applyGreen()`, `CrankNicholson::flow_lap_inv_pol`, `CrankNicholson::flow_lap_inv_tor`, `drs_flow::flow_pol`, `drs_flow::flow_pol_ddr`, `drs_flow::flow_pol_dr`, `drs_flow::flow_pol_lap`, `drs_flow::flow_tor`, `drs_flow::flow_tor_ddr`, `drs_flow::flow_tor_dr`, `drs_flow::flow_tor_lap`, `drs_mpi::mm`, `CrankNicholson::pinv`, `drs_radial::radial_dr_ddr_3D_n2r()`, `drs_radial::rcoll`, `drs_nonlinear::rhs_NS_pol`, `drs_nonlinear::rhs_NS_tor`, `drs_time::steps`, `drs_flow::update_flow_pol_lap()`, `drs_flow::update_flow_tor_lap()`, and `update_Green_functions()`.

Referenced by `drs()`.

Here is the call graph for this function:



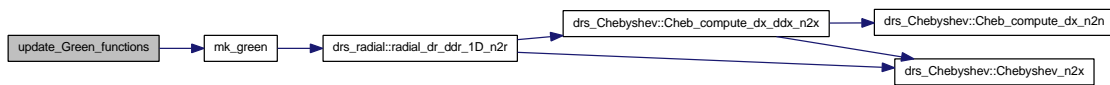
7.2.1.9 subroutine drs::update_Green_functions ()

Encapsulates dealing with the Green's functions.

References mk_green().

Referenced by update_flow().

Here is the call graph for this function:



7.2.1.10 subroutine drs::update_temp (double precision,intent(in) *h*, double precision,intent(in) *h1*, double precision,intent(in) *h2*, double precision,intent(in) *Pt*)

Updates the temperature using a hybrid Crank-Nicholson/Addams-Bashford implicit integration scheme.

Parameters:

h is the time step;

h1 and

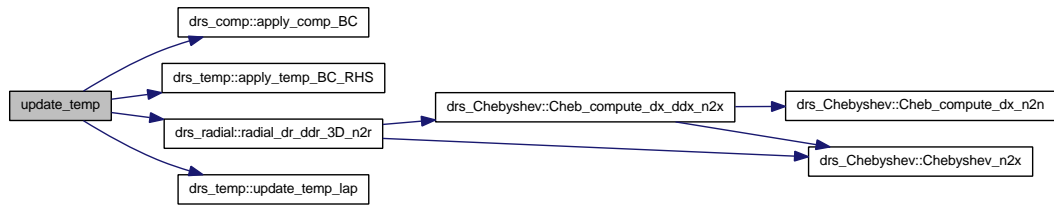
h2 are the Addams-Bashford weights;

Pt is the Thermal Prandtl number;

References drs_comp::apply_comp_BC(), drs_temp::apply_temp_BC_RHS(), drs_comp::comp, drs_comp::comp_ddr, drs_comp::comp_dr, drs_dims::Nr, drs_radial::radial_dr_ddr_3D_n2r(), drs_nonlinear::rhs_TE, drs_temp::temp, drs_temp::temp_ddr, drs_temp::temp_dr, drs_temp::temp_lap, CrankNicholson::temp_lap_inv, and drs_temp::update_temp_lap().

Referenced by drs().

Here is the call graph for this function:



7.3 drs_Chebyshev.f90 File Reference

Namespaces

- namespace **drs_Chebyshev**
Module containing the implementation of the Chebyshev polynomials.

Functions

- subroutine **drs_Chebyshev::Chebyshev_init** (N, N_s)
Computes the Chebyshev polynomials of order up to N as a function of r.
- subroutine **drs_Chebyshev::Chebyshev_cleanup** ()
- subroutine **drs_Chebyshev::Cheb_compute_dx_ddx_n2x** (f, dfdx, d2fdx2)
Returns second radial derivative in d2fdx2, first derivative in dfdx Input f is supposed to be given in Chebyshev space, derivatives are returned in direct space.
- subroutine **drs_Chebyshev::Cheb_compute_dx_ddx_x2x** (f, dfdx, d2fdx2)
Returns second radial derivative in d2fdx2, first derivative in dfdx Input f is supposed to be given in real space, derivatives are returned in real space.
- subroutine **drs_Chebyshev::Cheb_compute_dx_n2n** (f, dfdx)
Computes the Chebyshev coefficients of the first derivative of f with respect to x.
- subroutine **drs_Chebyshev::Chebyshev_x2n** (input)
The forward real to spectral cosinus transform Since $T_n(\cos(t)) = \cos(nt)$, the forward cosinus transform gives us the coefficients of order n of the expansion of a scalar function $f(x)$ in terms of Chebyshev polynomials.
- subroutine **drs_Chebyshev::Chebyshev_n2x** (input)
The backward spectral to real cosinus transform Since $T_n(\cos(t)) = \cos(nt)$, the backward cosinus transform gives us the value of a scalar function $f(x)$ in terms of Chebyshev polynomials.

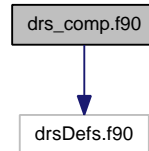
Variables

- double precision, parameter **drs_Chebyshev::pi** = 3.141592653589793d0
It makes use of fftw3.
- integer *8 **drs_Chebyshev::plan_x**
- integer **drs_Chebyshev::Nx**
- integer **drs_Chebyshev::Nx_s**
- double precision, dimension(:), allocatable **drs_Chebyshev::ct_buffer**
- double precision, allocatable **drs_Chebyshev::Cheb_x**
- double precision, allocatable, target **drs_Chebyshev::Chebyshev**
- double precision, allocatable, target **drs_Chebyshev::Chebyshev_dx**
- double precision, allocatable, target **drs_Chebyshev::Chebyshev_ddx**
First index is radial point, second index is mode index.

7.4 drs_comp.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs_comp.f90:



Namespaces

- namespace **drs_comp**
Module dealing with the composition.

Functions

- subroutine **drs_comp::drs_comp_allocation** ()
Allocates the variables required for computations involving composition.
- subroutine **drs_comp::drs_comp_init** ()
Initialises the composition boundary conditions, derivatives and profiles.
- character(len=16) **drs_comp::compProfName** ()
Outputs a human readable name for the composition profiles.
- subroutine **drs_comp::drs_comp_reset** ()
Resets the composition and its derivatives to 0.
- subroutine **drs_comp::drs_comp_randomize** (noise)
Computes the laplacian of the composition.
- subroutine **drs_comp::apply_comp_BC** (comp)
These lines take care of boundary conditions If the value at a boundary is bc different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.
- subroutine **drs_comp::calc_comp** (comp_spec, comp_real)
Computes the composition in real space from the composition in spectral space.

Variables

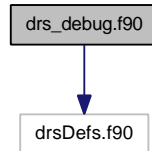
- double precision, dimension(:, :, :), allocatable **drs_comp::comp**
- double precision, dimension(:, :, :), allocatable **drs_comp::comp_dr**

- double precision, dimension(:, :, :), allocatable **drs_comp::comp_ddr**
- double precision, dimension(:, :, :), allocatable **drs_comp::comp_avg**
- double precision, dimension(:), allocatable **drs_comp::comp_dr_avg**
- double precision, dimension(:), allocatable **drs_comp::comp_profile**
- double precision, dimension(:), allocatable **drs_comp::comp_profile_dr**

7.5 drs_debug.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs_debug.f90:



Namespaces

- namespace **drs_debug**
Module with helper subroutines for debug.

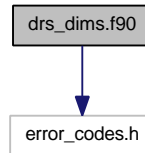
Functions

- subroutine **drs_debug::save_lmr_quantity** (t, tname)
- subroutine **drs_debug::save_tpr_quantity** (t, tname)

7.6 drs_dims.f90 File Reference

```
#include "error_codes.h"
```

Include dependency graph for drs_dims.f90:



Namespaces

- namespace **drs_dims**
Provides variables to store the real space and spectral space dimensions of the problem.

Functions

- subroutine **drs_dims::drs_dims_init** (error)
- subroutine **drs_dims::check_dims** (error)
Checks consistency of input parameters.

Variables

- integer, dimension(8), target **drs_dims::usr_dims**
- integer **drs_dims::Nr**
Number of radial points.
- integer **drs_dims::Nt**
Number of meridional points.
- integer **drs_dims::Np**
Number of azimuthal points.
- integer **drs_dims::Nr_s**
Highest index for the polynomials in the radial direction.
- integer **drs_dims::Nt_s**
Number of spherical harmonic degrees to use, including 0.
- integer **drs_dims::Np_s**
Number of spherical harmonic orders (positive, negative and zero) to use.
- integer **drs_dims::lsymm**
Equatorial symmetry.

- integer **drs_dims::m0**
Axial symmetry to use.

7.7 drs_fftw3.f90 File Reference

Namespaces

- namespace **drs_fftw3**

This module abstracts the computation of Fourier and cosinus transforms.

Functions

- subroutine **drs_fftw3::drs_fftw3_init** (Nr, Nt, Np)
Initialises all the fftw3 plans for forward and backward Fourier and cosinus transforms.
- subroutine **drs_fftw3::drs_fftw3_cleanup** ()
Destroys the plans.
- subroutine **drs_fftw3::dft_forward** (input, output)
The forward real to spectral DFT.
- subroutine **drs_fftw3::dft_backward** (input, output)
The backward, spectral to real DFT.
- subroutine **drs_fftw3::cos_r2r_1_r2n** (input)
The forward real to spectral cosinus transform.
- subroutine **drs_fftw3::cos_r2r_1_n2r** (input)
The backward spectral to real cosinus transform.
- subroutine **drs_fftw3::remesh** (Nr1, f1, Nr2, f2)
Given a field f1 described at Nr1 points in an interval, remesh outputs the same field, in the same interval at Nr2 points as f2.

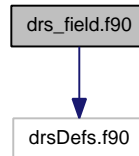
Variables

- integer *8 **drs_fftw3::plan_r**
It makes use of fftw3.
- integer *8 **drs_fftw3::plan_pf**
- integer *8 **drs_fftw3::plan_pb**
- double precision, dimension(:,:), allocatable **drs_fftw3::in_p**
- double precision, dimension(:), allocatable **drs_fftw3::inout_r**
- integer **drs_fftw3::drs_fftw3_Nr**
- integer **drs_fftw3::drs_fftw3_Np**
- integer **drs_fftw3::drs_fftw3_Nt**

7.8 drs_field.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs_field.f90:



Namespaces

- namespace **drs_field**

Functions

- subroutine **drs_field::drs_field_allocation** ()
- subroutine **drs_field::drs_field_init** (field_tor_dr, field_tor_ddr, field_pol_dr, field_pol_ddr)
- subroutine **drs_field::update_field_tor_lap** ()
- subroutine **drs_field::update_field_pol_lap** ()
- subroutine **drs_field::drs_field_random_init** (noise)
- subroutine **drs_field::apply_field_pol_BC** (pol, l, m)

These lines take care of boundary conditions If the value at a boundary is different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

- subroutine **drs_field::apply_field_tor_BC** (tor, l, m)

These lines take care of boundary conditions If the value at a boundary is different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

- subroutine **drs_field::calc_B** (Br_t, Bt_t, Bp_t, rot_Br_t, rot_Bt_t, rot_Bp_t)
- subroutine **drs_field::calc_field** (Br, Bt, Bp)
- subroutine **drs_field::calc_rot_field** (rotB_r, rotB_t, rotB_p)
- subroutine **drs_field::calc_field_lspec** (Bspec)
- subroutine **drs_field::calc_field_mspec** (Bspec)
- subroutine **drs_field::calc_field_nspec** (Bspec)

Variables

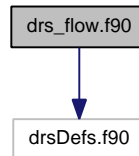
- double precision, dimension(:, :, :), allocatable **drs_field::field_pol**
- double precision, dimension(:, :, :), allocatable **drs_field::field_tor**
- double precision, dimension(:, :, :), allocatable **drs_field::field_pol_dr**
- double precision, dimension(:, :, :), allocatable **drs_field::field_tor_dr**
- double precision, dimension(:, :, :), allocatable **drs_field::field_pol_ddr**

- double precision, dimension(:, :, :), allocatable **drs_field::field_tor_ddr**
- double precision, dimension(:, :, :), allocatable **drs_field::field_pol_lap**
- double precision, dimension(:, :, :), allocatable **drs_field::field_tor_lap**
- double precision, dimension(:, :, :), allocatable **drs_field::field_pol_avg**
- double precision, dimension(:, :, :), allocatable **drs_field::field_tor_avg**

7.9 drs_flow.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs_flow.f90:



Namespaces

- namespace **drs_flow**

Functions

- subroutine **drs_flow::drs_flow_allocation** ()
- subroutine **drs_flow::drs_flow_init** (flow_tor_dr, flow_tor_ddr, flow_pol_dr, flow_pol_ddr)
- subroutine **drs_flow::update_flow_tor_lap** ()
- subroutine **drs_flow::update_flow_pol_lap** ()
- subroutine **drs_flow::apply_flow_pol_BC** (pol)

These lines take care of boundary conditions If the value at a boundary is bc different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

- subroutine **drs_flow::apply_flow_tor_BC** (tor)

These lines take care of boundary conditions If the value at a boundary is bc different from 0, insert bc/2 because of the factor 2 between Chebychev and radial differentiation If the boundary condition is not on the derivative but the value of the function itself (as is the case for the temperature), no factor of 2 is needed if the boundary value bc is different from 0.

- subroutine **drs_flow::calc_u** (ur, ut, up, rotu_r, rotu_t, rotu_p)

Abstracts computing the flow and its curl in real space.

- subroutine **drs_flow::calc_flow** (ur_t, ut_t, up_t)

This routine computes:.

- subroutine **drs_flow::calc_rot_flow** (rotu_r, rotu_t, rotu_p)

This routine computes:.

- subroutine **drs_flow::calc_flow_lspec** (uspec)

Computes the l-spectrum of the radial flow.

- subroutine **drs_flow::calc_flow_mspec** (uspec)

Computes the m-spectrum of the radial flow.

- subroutine **drs_flow::calc_flow_nspec** (uspec)

Computes the n-spectrum of the radial flow.

Variables

- double precision, dimension(:, :, :), allocatable **drs_flow::flow_pol**
- double precision, dimension(:, :, :), allocatable **drs_flow::flow_tor**
- double precision, dimension(:, :, :), allocatable **drs_flow::flow_pol_dr**
- double precision, dimension(:, :, :), allocatable **drs_flow::flow_tor_dr**
- double precision, dimension(:, :, :), allocatable **drs_flow::flow_pol_ddr**
- double precision, dimension(:, :, :), allocatable **drs_flow::flow_tor_ddr**
- double precision, dimension(:, :, :), allocatable **drs_flow::flow_pol_lap**
- double precision, dimension(:, :, :), allocatable **drs_flow::flow_tor_lap**
- double precision, dimension(:, :, :), allocatable **drs_flow::flow_pol_avg**
- double precision, dimension(:, :, :), allocatable **drs_flow::flow_tor_avg**

7.10 drs_hypDiff.f90 File Reference

Classes

- interface **drs_hypDiff::drs_apply_hypDiff**

Namespaces

- namespace **drs_hypDiff**

Functions

- subroutine **drs_hypDiff::drs_hypDiff_init** (Nt)

Variables

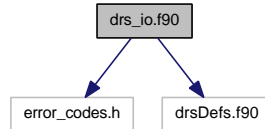
- double precision, allocatable **drs_hypDiff::hypDiff**
- logical **drs_hypDiff::drs_want_hypDiff** = .FALSE.

7.11 drs_io.f90 File Reference

```
#include "error_codes.h"
```

```
#include "drsDefs.f90"
```

Include dependency graph for drs_io.f90:



Namespaces

- namespace **drs_io**

Deals with input and output of state files and derived quantities.

Functions

- subroutine **drs_io::drs_load_state** (error)

Reads a state performing interpolation as needed. The state is stored in the files with name given by `io_calc_file_in` and are described by the file with extension `.par`.

- subroutine **drs_io::drs_open_output** ()

Opens units for regularly probed quantities to be saved.

- subroutine **drs_io::dump_state** ()

- subroutine **drs_io::save_state** ()

Saves the present state to file. At this point all files are saved with the file name given by `io_calc_file_out`.

- subroutine **drs_io::save_l_spec** ()

Saves the normalized power spectra with respect to `l`.

- subroutine **drs_io::save_m_spec** ()

Saves the normalized power spectra with respect to `m`.

- subroutine **drs_io::save_n_spec** ()

Saves the normalized power spectra of all quantities with respect to `n`.

Variables

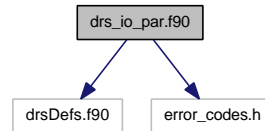
- character(len=60) **drs_io::io_calc_file_in**
- character(len=60) **drs_io::io_calc_file_out**
- character(len=13), parameter **drs_io::deflate**
- character(len=15), parameter **drs_io::inflate**

7.12 drs_io_par.f90 File Reference

```
#include "drsDefs.f90"
```

```
#include "error_codes.h"
```

Include dependency graph for drs_io_par.f90:



Namespaces

- namespace **drs_io_par**
Module to read and write parameter and configuration files.

Functions

- subroutine **drs_io_par::drs_read_conf_v2** (io_calc_file_in, io_calc_file_out, comment, error)
reads parameters for the calculation from the standard input
- subroutine **drs_io_par::drs_read_conf** (io_calc_file_in, io_calc_file_out, comment, error)
reads parameters for the calculation from the standard input
- subroutine **drs_io_par::read_input_par** (unit_in)
reads the parameterfile 'file'.par
- subroutine **drs_io_par::write_parp** (unit_out)
writes the parameter file 'file'.par

Variables

- integer, dimension(8), target **drs_io_par::usr_dimsi**
- integer **drs_io_par::lformi**
- integer **drs_io_par::drs_calc_typei**
- integer **drs_io_par::tempBCi**
- integer **drs_io_par::flowBCi**
- integer **drs_io_par::magBCi**
- integer **drs_io_par::Nri**
- integer **drs_io_par::Nti**
- integer **drs_io_par::Npi**
- integer **drs_io_par::Nri_s**
- integer **drs_io_par::Nti_s**
- integer **drs_io_par::Npi_s**
- integer **drs_io_par::lsymmi**
- integer **drs_io_par::m0i**
- double precision **drs_io_par::etai**

- double precision **drs_io_par::Pti**
- double precision **drs_io_par::Tai**
- double precision **drs_io_par::Ra_ti**
- double precision **drs_io_par::Pmi**
- double precision **drs_io_par::hi**
- double precision **drs_io_par::drifti**
- double precision **drs_io_par::noise**
- integer **drs_io_par::stepmaxi**
- integer **drs_io_par::sampling_ratei**
- integer **drs_io_par::transienti**
- character(len=60) **drs_io_par::commenti**
- integer **drs_io_par::magici**
- integer, parameter **drs_io_par::magic** = 10205
- integer, parameter **drs_io_par::MAGICC1** = 10101
- integer, parameter **drs_io_par::MAGICC2** = 10102
- integer, parameter **drs_io_par::MAGICC3** = 10103
- integer, parameter **drs_io_par::MAGICC4** = 10104
- integer, parameter **drs_io_par::MAGICC5** = 10105
- integer, parameter **drs_io_par::MAGICC6** = 10106
- integer, parameter **drs_io_par::MAGICC7** = 10107
- integer, parameter **drs_io_par::MAGICC9** = 10109

7.13 drs_io_units.f90 File Reference

Namespaces

- namespace **drs_io_units**
Manages the I/O units of DRS.

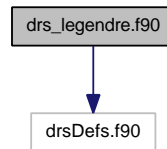
Variables

- integer, parameter **drs_io_units::unit_ek** = 11
- integer, parameter **drs_io_units::unit_ur** = 12
- integer, parameter **drs_io_units::unit_uzon** = 13
- integer, parameter **drs_io_units::unit_koeu** = 14
- integer, parameter **drs_io_units::unit_uaz** = 15
- integer, parameter **drs_io_units::unit_u_mid** = 16
- integer, parameter **drs_io_units::unit_am** = 17
- integer, parameter **drs_io_units::unit_nu** = 21
- integer, parameter **drs_io_units::unit_adv** = 22
- integer, parameter **drs_io_units::unit_t** = 23
- integer, parameter **drs_io_units::unit_eb** = 31
- integer, parameter **drs_io_units::unit_koeb** = 32
- integer, parameter **drs_io_units::unit_dissu** = 33
- integer, parameter **drs_io_units::unit_dissB** = 34
- integer, parameter **drs_io_units::unit_mspect** = 41
- integer, parameter **drs_io_units::unit_lspect** = 42
- integer, parameter **drs_io_units::unit_nspect** = 43
- integer, parameter **drs_io_units::unit_evp** = 51
- integer, parameter **drs_io_units::unit_evt** = 52
- integer, parameter **drs_io_units::unit_cfl** = 99

7.14 drs_legendre.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs_legendre.f90:



Classes

- interface **drs_legendre::interface**

Namespaces

- namespace **drs_legendre**

Functions

- subroutine **drs_legendre::drs_legendre_allocation** ()
- subroutine **drs_legendre::drs_legendre_init** ()
Initialize the Legendre associated Polynomials and the Gauss-Legendre co-location points.
- subroutine **drs_legendre::initNormalization** (normType, lmax, norms)
Computes the normalization factors for the the Legendre associated Polynomials.
- subroutine **drs_legendre::legendre_init_new** ()
Initializes the tables of Associated Legendre Polynomials.
- subroutine **drs_legendre::gauleg** (x1, x2, x, w, n)
Computes the Gauss-Legendre quadrature points and weights.

Variables

- double precision, dimension(:, :, :), allocatable **drs_legendre::legendre**
The unnormalised Legendre polynomials.
- double precision, dimension(:, :, :), allocatable **drs_legendre::leg_neg**
The unnormalised Legendre polynomials for negative m multiplied by the integration factors.
- double precision, dimension(:, :, :), allocatable **drs_legendre::dleg**
d Plm(cos(theta))/d theta
- double precision, dimension(:, :, :), allocatable **drs_legendre::leg_sin**

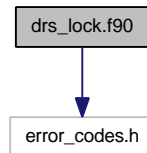
$P_l m / \sin(\theta)$.

- double precision, dimension(:,), allocatable **drs_legendre::plmfac**
 $\text{sqrt}((l+m)!/(l-m)!/(2l+1)) = \text{sqrt}((l-m+1)*(l-m+2)*\dots*((l+m)/(2l+1))$
- double precision, dimension(:,), allocatable, target **drs_legendre::costheta**
Gauss-Legendre integration points.
- double precision, dimension(:,), allocatable, target **drs_legendre::sintheta**
Gauss-Legendre integration points.
- double precision, dimension(:,), allocatable **drs_legendre::w**
Gauss-Legendre integration weights.
- integer, dimension(:,), allocatable **drs_legendre::llp1**
Table of $l(l+1)$.
- double precision, parameter **drs_legendre::pi** = 3.141592653589793d0

7.15 drs_lock.f90 File Reference

```
#include "error_codes.h"
```

Include dependency graph for drs_lock.f90:



Namespaces

- namespace **drs_lock**
This module provides a locking mechanism for the dynamo code.

Functions

- subroutine **drs_lock::drs_lock_init** (u, f)
Sets the lock file name to f and manages it on unit u.
- subroutine **drs_lock::add_lock** (error)
Creates the lock file.
- subroutine **drs_lock::rm_lock** (error)
Removes the lock file.
- logical **drs_lock::lockExists** ()
Checks whether the lock file exists.

Variables

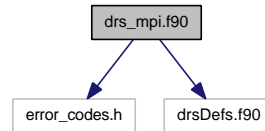
- character(len=128) **drs_lock::lockFileName**
- integer **drs_lock::lockFileUnit** = -1

7.16 drs_mpi.f90 File Reference

```
#include "error_codes.h"
```

```
#include "drsDefs.f90"
```

Include dependency graph for drs_mpi.f90:



Classes

- interface **drs_mpi::sum_over_all_cpus**
Encapsulates sums of several types and ranks.
- interface **drs_mpi::drs_minimize**
Encapsulates minimization of several types and ranks.
- interface **drs_mpi::drs_maximize**
Encapsulates maximization of several types and ranks.
- interface **drs_mpi::drs_bcast**
Encapsulates broadcast of several types and ranks.

Namespaces

- namespace **drs_mpi**
Provides initialisation and variables to be used with the mpi implementation.

Functions

- subroutine **drs_mpi::drs_mpi_init** ()
Gets initial values for mpi_size and mpi_rank. Allocates block indices accordingly.
- subroutine **drs_mpi::mpi_dims_init** (Nt, Np_s, m0, error)
Initializes mpi variables and sizes.
- subroutine **drs_mpi::transpos_phi2theta** (input, Nt, output, Np)
transposition: $t \text{ distrib}(\phi) \rightarrow tt_t \text{ distrib}(\theta)$:
- subroutine **drs_mpi::transpos_theta2phi** (input, Np_s, output, Nt)
transposition: $tt_t \text{ distrib}(\theta) \rightarrow t \text{ distrib}(\phi)$:
- subroutine **drs_mpi::distribute_in_m** (buffer, Nt, Nr)

Performs a one-to-all communication of the contents of buffer. It is essentially a targeted version of mpi_scatter.

- subroutine **drs_mpi::gather_from_m** (buffer, Nt, Nr)
Performs an all-to-one communication of the contents of buffer. It is essentially a targeted version of mpi_gather.
- subroutine **drs_mpi::blk_ts_start_init** (m0)
Initialises blk_ts_start.
- subroutine **drs_mpi::sum_over_all_cpus_scal** (val)
Subroutine to encapsulate sums across all the cpu's.
- subroutine **drs_mpi::sum_over_all_cpus_vect** (val)
Subroutine to encapsulate mpi calls that sum arrays over all cpu's.
- subroutine **drs_mpi::wait_for_everyone** ()
Encapsulate mpi barrier.
- subroutine **drs_mpi::drs_minimize_dble** (array)
Encapsulate mpi_reduce min.
- subroutine **drs_mpi::drs_minimize_dble_scal** (val)
Encapsulate mpi_reduce min (scalars).
- subroutine **drs_mpi::drs_maximize_dble** (array)
Encapsulate mpi_reduce max.
- subroutine **drs_mpi::drs_maximize_dble_scal** (val)
Encapsulate mpi_reduce max (scalars).
- subroutine **drs_mpi::drs_gather_vars** (rank, val)
- subroutine **drs_mpi::drs_bcast_dble** (array, num)
- subroutine **drs_mpi::drs_bcast_int** (array, num)
- subroutine **drs_mpi::drs_bcast_dble_scal** (val)
- subroutine **drs_mpi::drs_bcast_int_scal** (val)
- subroutine **drs_mpi::drs_bcast_logical_scal** (val)
- subroutine **drs_mpi::drs_abort** (error)
- subroutine **drs_mpi::mpi_cleanup** ()

Variables

- integer **drs_mpi::mpi_size**
How many CPU's are in use.
- integer **drs_mpi::mpi_rank**
The rank of the present CPU.
- integer, dimension(:), allocatable, target **drs_mpi::blk_ps_start**

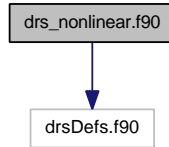
Start index of the blocks in m for each CPU.

- integer, dimension(:), allocatable, target **drs_mpi::blk_ps_size**
Size of the blocks in m for each CPU.
- integer, dimension(:), allocatable, target **drs_mpi::blk_t_start**
Start index of the blocks in theta dor each CPU.
- integer, dimension(:), allocatable, target **drs_mpi::blk_t_size**
Size of the blocks in theta for each CPU.
- integer, dimension(:), allocatable, target **drs_mpi::blk_ts_start**
Stores the index of the first nonzero l value in the block.
- integer, dimension(:), pointer **drs_mpi::mm**
A convinience shorthand for blk_ts_start.
- integer **drs_mpi::blk_t_max_size**
Maximum size of theta block per cpu.
- integer **drs_mpi::blk_ps_max_size**
Maximum size of phi block per cpu.

7.17 drs_nonlinear.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs_nonlinear.f90:



Namespaces

- namespace **drs_nonlinear**

Takes care of constructing the nonlinear terms of all equations and other quantities in real space.

Functions

- subroutine **drs_nonlinear::drs_nonlinear_init** ()
- subroutine **drs_nonlinear::evaluate_real_space** ()
- subroutine **drs_nonlinear::rhs** (h_old, h)
- subroutine **drs_nonlinear::save_stuff** (nsample)

Encapsulate saving quantities in real and spectral space.

Variables

- double precision, allocatable **drs_nonlinear::rhs_NS_tor**
NS for Navier-Stokes.
- double precision, allocatable **drs_nonlinear::rhs_NS_pol**
- double precision, allocatable **drs_nonlinear::rhs_IE_tor**
IE for Induction Equation.
- double precision, allocatable **drs_nonlinear::rhs_IE_pol**
- double precision, allocatable **drs_nonlinear::rhs_TE**
TE for Temperature Equation.
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::temp_t**
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::flow_r_t**
Quantities in real space.
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::flow_t_t**
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::flow_p_t**
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::field_r_t**
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::field_t_t**
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::field_p_t**

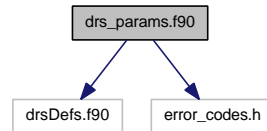
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::rot_flow_r_t**
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::rot_flow_t_t**
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::rot_flow_p_t**
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::rot_field_r_t**
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::rot_field_t_t**
- double precision, dimension(:, :, :), allocatable **drs_nonlinear::rot_field_p_t**
- integer, parameter **drs_nonlinear::ncfl** = 5
- double precision, dimension(ncfl) **drs_nonlinear::cfl**

7.18 drs_params.f90 File Reference

```
#include "drsDefs.f90"
```

```
#include "error_codes.h"
```

Include dependency graph for drs_params.f90:

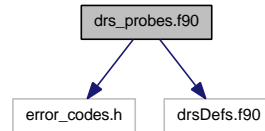


7.19 drs_probes.f90 File Reference

```
#include "error_codes.h"
```

```
#include "drsDefs.f90"
```

Include dependency graph for drs_probes.f90:



Namespaces

- namespace **drs_probes**

This module implements some prbing facilities for the running models.

Functions

- subroutine **drs_probes::drs_probes_allocation** ()
- subroutine **drs_probes::drs_probes_init** (time)
- subroutine **drs_probes::measure_lm** ()
- subroutine **drs_probes::average_unnormalised_flow_l_spectrum** (urspec_avg)
- subroutine **drs_probes::average_unnormalised_field_l_spectrum** (Brspec_avg)
- subroutine **drs_probes::average_unnormalised_scalar_l_spectrum** (scalar, scalar_spec_avg)
- subroutine **drs_probes::l_spec_of_scalar_field** (field, spec)

Calculate the normalized power spectrum with respect to l of a scalar field.

- subroutine **drs_probes::m_spec_of_scalar_field** (field, spec)

Calculates the normalized power spectrum of a scalar field with respect to m.

- subroutine **drs_probes::n_spec_of_scalar_field** (field, spec)

Calculates the normalized power spectrum of a scalar quantity f with respect to the Chebyshev polynomials.

$$R_n = \sum_{l,m} N_l^m (f_{nl}^m)^2$$

- double precision **drs_probes::integrate_r** (input)

Performs the integration of the 1d real array.

- function **drs_probes::c_integrate_r** (input)

Performs the integration of the 1d complex array.

- subroutine **drs_probes::save_magnetic_dissipation** (mmax)

Computes the magnetic dissipation truncated up to degree.

- subroutine **drs_probes::save_flow_dissipation** (mmax)

Computes the viscous dissipation.

- subroutine **drs_probes::save_flow_coeffs** ()
Saves some flow coefficients at the present instant.
- subroutine **drs_probes::save_field_coeffs** ()
Saves some field coefficients at the present instant.
- subroutine **drs_probes::check_resolution_Hartman** (Rm, error)
- double precision **drs_probes::energy** (vr, vt, vp)
*Computes the energy of a vector field based on its components.
Only root contains the solution.*
- subroutine **drs_probes::measure** (temp2_t, ur_t, utheta_t, uphi_t, rotu_r_t, rotu_theta_t, rotu_phi_t, Br_t, Btheta_t, Bphi_t, cfl)
- subroutine **drs_probes::compute_helicities** (ur, ut, up, rotu_r, rotu_t, rotu_p, helicity_south, helicity_north)
- subroutine **drs_probes::compute_advection** (ur, temp, advect)
Computes the heat transported by advection. as

$$Q(r) = \int \int u_r(r, \theta, \phi) * (\Theta(r, \theta, \phi) + T_S(r)) \sin \theta d\theta d\phi$$

- double precision **drs_probes::nusselt** (r)
Computes the Nusselt number, that is, the the ratio between the convective and the diffusive heat fluxes.
- subroutine **drs_probes::integrate_power_surf** (f, n, f_int)
Performs the integration in theta and phy of a function f raised to the power n. as

$$F(r) = \int \int f(r, \theta, \phi)^n \sin \theta d\theta d\phi$$

- subroutine **drs_probes::save_angular_momentum** (u_t, u_p)
computes and saves the three cartesian components of the total angular momentum

Variables

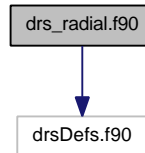
- double precision, dimension(:), allocatable **drs_probes::ur_avg**
- double precision, dimension(:), allocatable **drs_probes::ut_avg**
- double precision, dimension(:), allocatable **drs_probes::up_avg**
- double precision, dimension(:), allocatable **drs_probes::up2**
- double precision, dimension(:), allocatable **drs_probes::ut2**
- double precision, dimension(:), allocatable **drs_probes::adv_avg**
- double precision, dimension(:), allocatable **drs_probes::t2_avg**
- double precision, dimension(:), allocatable **drs_probes::tspec_avg**
- double precision, dimension(:), allocatable **drs_probes::urspec_avg**
- double precision, allocatable **drs_probes::Brspec_avg**
- double precision **drs_probes::groth**

- double precision **drs_probes::Ekin**
- double precision **drs_probes::EB**
- double precision **drs_probes::nkes**
energies from measure_lm:
- double precision **drs_probes::nkea**
- double precision **drs_probes::etors**
- double precision **drs_probes::etora**
- double precision **drs_probes::drkes**
- double precision **drs_probes::drkea**
- double precision **drs_probes::mckes**
- double precision **drs_probes::mckea**
- double precision **drs_probes::Bnkes**
- double precision **drs_probes::Bnkea**
- double precision **drs_probes::Betors**
- double precision **drs_probes::Betora**
- double precision **drs_probes::Bdrkes**
- double precision **drs_probes::Bdrkea**
- double precision **drs_probes::Bmckes**
- double precision **drs_probes::Bmckea**
- double precision, allocatable **drs_probes::dOmega**
Weights for volume integration.
- double precision **drs_probes::Rm** = 1.0d0

7.20 drs_radial.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs_radial.f90:



Namespaces

- namespace **drs_radial**

This module implements the radial domain and operations in it.

Functions

- subroutine **drs_radial::drs_radial_init** (riro)
Initializes the radial domain and the Chebyshev polynomials and their derivatives.
- double precision, dimension(nr) **drs_radial::radial_derivative_r2r** (radarr)
Returns the first derivative of radarr. radarr is supposed to be given in direct space, derivative is returned in direct space.
- subroutine **drs_radial::radial_dr_ddr_1D_n2r** (t, t1, t2)
Returns first radial derivative in t1, second derivative in t2. Input t is supposed to be given in spectral space. On output both t and its derivatives are returned in real space.
- subroutine **drs_radial::radial_dr_ddr_1D_r2r** (t, t1, t2)
*A factor of 2 for each derivative is due to the mapping from the radial coordinate r to the Chebyshev coordinate x, where x runs from -1 to 1. The interrelation is $r = \eta/(1-\eta) + 0.5(x+1)$ see the def. of rcoll in the initialization routine. Obviously, $d/dr = 2*d/dx$.*
- subroutine **drs_radial::radial_dr_ddr_3D_r2r** (t, t1, t2)
Calculates first and second radial derivatives of 3D-array in lmr space. Includes dealiasing in n.
- subroutine **drs_radial::radial_dr_ddr_3D_n2r** (t0, t1, t2)
Calculates first and second radial derivatives of 3D-array in lmn space. includes dealiasing in n. Transforms the original field to lmr space.

Variables

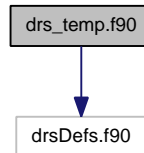
- double precision, dimension(:), allocatable **drs_radial::rcoll**
Radial collocation points for Chebychev polynomials.

- double precision, dimension(:), allocatable **drs_radial::rcoll2**
Squares of radial collocation points for Chebychev polynomials.
- double precision, dimension(:), allocatable **drs_radial::drcoll**
Differences for radial collocation points for Chebychev polynomials.
- double precision, dimension(:, :), allocatable **drs_radial::poly**
- double precision, dimension(:, :), allocatable **drs_radial::poly_dr**
- double precision, dimension(:, :), allocatable **drs_radial::poly_ddr**
- integer, dimension(2) **drs_radial::b**
Index of the boundaries: b(1)=inner boundary; b(2)=outer boundary.

7.21 drs_temp.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs_temp.f90:



Namespaces

- namespace **drs_temp**
Temperature related operations.

Functions

- subroutine **drs_temp::drs_temp_allocation** ()
Allocates the temperature related variables.
- subroutine **drs_temp::drs_temp_init** ()
Precomputes the adimensional radial temperature profile.
- character(len=16) **drs_temp::tempProfName** ()
Outputs a human readable name for the temperature profiles.
- subroutine **drs_temp::drs_temp_reset** ()
Resets the temperature and its derivatives to zero.
- subroutine **drs_temp::update_temp_lap** ()
Recomputes and caches the laplacian of the temperature.
- subroutine **drs_temp::drs_temp_randomize** (noise)
- subroutine **drs_temp::apply_temp_BC_RHS** (val)
Boundary conditions for the temperature. For fixed temperature, the value of the anomaly is set to be zero.
- subroutine **drs_temp::calc_temp** (temp_t)
Computes the temperature anomaly in real space.

Variables

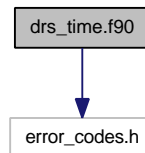
- double precision, dimension(:, :, :), allocatable **drs_temp::temp**
- double precision, dimension(:, :, :), allocatable **drs_temp::temp_dr**
- double precision, dimension(:, :, :), allocatable **drs_temp::temp_ddr**

- double precision, dimension(:, :, :), allocatable **drs_temp::temp_lap**
- double precision, dimension(:, :, :), allocatable **drs_temp::temp_avg**
- double precision, dimension(:), allocatable **drs_temp::temp_dr_avg**
- double precision, dimension(:), allocatable **drs_temp::temp_profile**
- double precision, dimension(:), allocatable **drs_temp::temp_profile_dr**

7.22 drs_time.f90 File Reference

```
#include "error_codes.h"
```

Include dependency graph for drs_time.f90:



Namespaces

- namespace **drs_time**
Module to deal with time. It deals with both wall time and simulation time. It also deals wit the time-stepping.

Functions

- subroutine **drs_time::drs_time_init** ()
- subroutine **drs_time::drs_time_update** ()
Updates the current simulation time and step index.
- subroutine **drs_time::update_timestep** (cfl, h, h_old, stat)
Updates the time-step according to the cfl numbers.
- subroutine **drs_time::update_time_last_sample** (tnew)
Updates the time we last took a sample of some quantities.

Variables

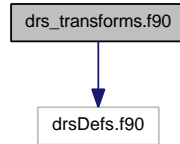
- double precision **drs_time::time_start**
The simulation time before the first step.
- double precision **drs_time::time**
The simulation time.
- double precision **drs_time::max_time**
The maximum simulation time.
- double precision **drs_time::h**
The simulation time-step.
- double precision **drs_time::h_old**
The simulation time step at the previous step.

- double precision **drs_time::drift**
The drift rate in radians per simulation time unit.
- double precision **drs_time::time_last_sample**
Time we last saved the probes values.
- double precision **drs_time::time_since_last_sample**
Time since we last saved the probes values.
- logical **drs_time::variable_h**
Do we have a variable time step?
- real **drs_time::cpu_time_start**
Wall time when we started the run in seconds.
- real **drs_time::cpu_time_now**
Wall time now, in seconds.
- real **drs_time::cpu_time_first_step**
Wall time after we finished the first Newton step, in seconds.
- real **drs_time::cpu_max_time**
The maximum wall time in hours.
- integer **drs_time::transient**
How many steps to consider as a transient and discard?
- integer **drs_time::sampling_rate**
The sampling rate in integer steps.
- integer **drs_time::stepmax**
The maximum number of steps to take.
- integer **drs_time::nsample**
How many samples we took so far.
- integer **drs_time::steps**
Steps taken so far in total (includes stepstart).
- integer **drs_time::stepstart**
The step count we start the run with.
- double precision, dimension(3), target **drs_time::dtimestep**
- integer, dimension(5), target **drs_time::imeasure**

7.23 drs_transforms.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs_transforms.f90:



Namespaces

- namespace **drs_transforms**
Spherical transforms.

Functions

- subroutine **drs_transforms::ylmt** (input, output)
This routine performs three steps: ~~~~~ 1. transformation $tt_t(\theta, \phi) \rightarrow tt_t(\theta, m)$ transposed. 2. redistribution $dist(\theta) \rightarrow dist(m)$ 3. transformation $tt(\theta, m) \rightarrow t(l, m)$ distrib. in m .
- subroutine **drs_transforms::ylmt_3D** (input, output)
This routine performs three steps: ~~~~~ 1. transformation $tt_t(\theta, \phi) \rightarrow tt_t(\theta, m)$ transposed. 2. redistribution $dist(\theta) \rightarrow dist(m)$ 3. transformation $tt(\theta, m) \rightarrow t(l, m)$ distrib. in m .
- subroutine **drs_transforms::ylmb** (input, output)
- subroutine **drs_transforms::m2phi_2D** (input, output)
- subroutine **drs_transforms::my_div** (vec_r, vec_t, vec_p, nonlin)
*Computes the spectral coefficients of the divergence of the vector field \vec{u} .
On input:.*
- subroutine **drs_transforms::my_rot** (vec_t, vec_p, nonlin)
- subroutine **drs_transforms::my_rotrot** (vec_r, vec_t, vec_p, nonlin)
- subroutine **drs_transforms::vectorField2PolTor_common** (vr, vt, vp, pol, tor)
*Computes the poloidal and toroidal scalar coefficients of a 3D vector field.
This is the part of the calculation that is common to both the flow and magnetic field.*
- subroutine **drs_transforms::PolTor_common2PolTor_flow** (pol, tor)
Multiplies the poloidal and toroidal scalar coefficients by the appropriate factors of r and $l(l+1)$ to make them the poloidal and toroidal scalars of the flow as defined for this program.*
- subroutine **drs_transforms::PolTor_common2PolTor_field** (pol, tor)
Multiplies the poloidal and toroidal scalar coefficients by the appropriate factors of r and $l(l+1)$ to make them the poloidal and toroidal scalars of the magnetic field as defined for this program.*
- subroutine **drs_transforms::vectorField2Divergence** (ur, ut, up, div)
Computes the spherical harmonic coefficients of the divergence of a 3D vector field in real space.

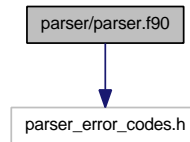
Variables

- double precision, parameter **drs_transforms::pi** = 3.141592653589793d0

7.24 parser/parser.f90 File Reference

```
#include "parser_error_codes.h"
```

Include dependency graph for parser.f90:



Classes

- struct **parser::parser_vars**
A description of the keys we will find and their properties.
- struct **parser::parser_section**
A description of the sections we will find and their properties.
- interface **parser::read_val**
Reads a value.

Namespaces

- namespace **parser**
*This module provides a simple **parser** (p. 123) for input files of the type 'key = val' eventually separated by '[Sections]'.*

Functions

- subroutine **parser::parse** (unit_in, variable, line, error)
Reads one line from the specified unit and outputs the variable name and possible values as a string.

7.25 SHTOOLS/PLegendreA_d1.f90 File Reference

Functions

- subroutine **PLegendreA_d1** (*p*, *dp*, *lmax*, *z*, *csphase*)

7.25.1 Function Documentation

- 7.25.1.1** subroutine **PLegendreA_d1** (*real*8,dimension(:),intent(out)* *p*,
*real*8,dimension(:),intent(out)* *dp*, *integer,intent(in)* *lmax*, *real*8,intent(in)* *z*,
integer,intent(in),optional *csphase*)

7.26 SHTOOLS/PlmBar_d1.f90 File Reference

Functions

- subroutine **PlmBar_d1** (*p*, *dp*, *lmax*, *z*, *csphase*, *cnorm*)

7.26.1 Function Documentation

7.26.1.1 subroutine **PlmBar_d1** (*real*8,dimension(:),intent(out)* *p*, *real*8,dimension(:),intent(out)* *dp*, *integer,intent(in)* *lmax*, *real*8,intent(in)* *z*, *integer,intent(in),optional* *csphase*, *integer,intent(in),optional* *cnorm*)

Referenced by `drs_legendre::legendre_init_new()`.

7.27 SHTOOLS/PlmIndex.f90 File Reference

Functions

- integer **PlmIndex** (*l*, *m*)

7.27.1 Function Documentation

7.27.1.1 integer **PlmIndex** (integer,intent(in) *l*, integer,intent(in) *m*)

Parameters:

- l* This function will return the index corresponding to a given *l* and *m* in the arrays of Legendre Polynomials generated by routines such as PlmBar and PlmSchmidt.

Calling Parameters: *l* Spherical harmonic angular degree. *m* Spherical harmonic angular order.

Dependencies: None

Written by Mark Wieczorek (May 2004)

Copyright (c) 2005, Mark A. Wieczorek All rights reserved.

Parameters:

- m* This function will return the index corresponding to a given *l* and *m* in the arrays of Legendre Polynomials generated by routines such as PlmBar and PlmSchmidt.

Calling Parameters: *l* Spherical harmonic angular degree. *m* Spherical harmonic angular order.

Dependencies: None

Written by Mark Wieczorek (May 2004)

Copyright (c) 2005, Mark A. Wieczorek All rights reserved.

Referenced by drs_legendre::legendre_init_new().

7.28 tests/test_drs_Chebyshev.f90 File Reference

Functions

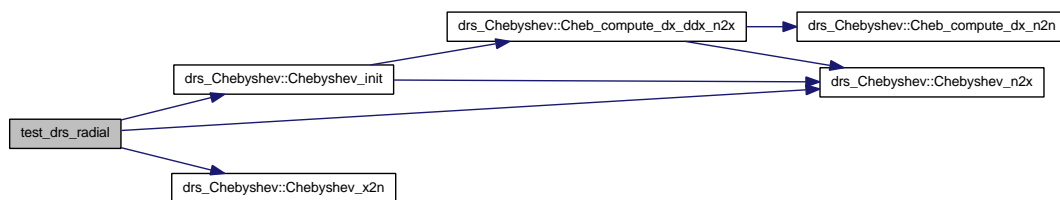
- program `test_drs_radial`

7.28.1 Function Documentation

7.28.1.1 program `test_drs_radial` ()

References `drs_Chebyshev::Chebyshev`, `drs_Chebyshev::Chebyshev_dx`, `drs_Chebyshev::Chebyshev_init`(), `drs_Chebyshev::Chebyshev_n2x`(), and `drs_Chebyshev::Chebyshev_x2n`().

Here is the call graph for this function:



7.29 tests/test_drs_fftw-r2r.f90 File Reference

Functions

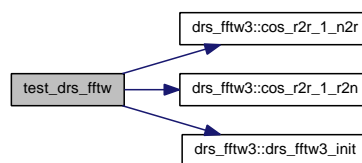
- program `test_drs_fftw`

7.29.1 Function Documentation

7.29.1.1 program `test_drs_fftw` ()

References `drs_fftw3::cos_r2r_1_n2r()`, `drs_fftw3::cos_r2r_1_r2n()`, and `drs_fftw3::drs_fftw3_init()`.

Here is the call graph for this function:



7.30 tests/test_drs_fftw.f90 File Reference

Functions

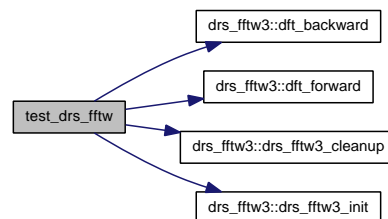
- program `test_drs_fftw`

7.30.1 Function Documentation

7.30.1.1 program `test_drs_fftw` ()

References `drs_fftw3::dft_backward()`, `drs_fftw3::dft_forward()`, `drs_fftw3::drs_fftw3_cleanup()`, and `drs_fftw3::drs_fftw3_init()`.

Here is the call graph for this function:



7.31 tests/test_drs_radial.f90 File Reference

Functions

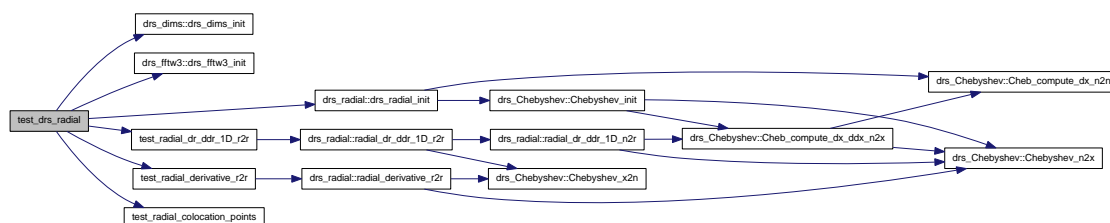
- program **test_drs_radial**
- logical **test_radial_colocation_points** ()
Tests the correctnes of the collocation points.
- logical **test_radial_derivative_r2r** ()
*Tests if the derivative of $r**2$ is $2*r$.*
- logical **test_radial_dr_ddr_1D_r2r** ()
*Tests if the first derivative of $r**2$ is $2*r$ and the second is 2.*

7.31.1 Function Documentation

7.31.1.1 program test_drs_radial ()

References drs_dims::drs_dims_init(), drs_fftw3::drs_fftw3_init(), drs_radial::drs_radial_init(), drs_dims::Np, drs_dims::Nr, drs_dims::Nr_s, drs_dims::Nt, test_radial_colocation_points(), test_radial_derivative_r2r(), and test_radial_dr_ddr_1D_r2r().

Here is the call graph for this function:



7.31.1.2 logical test_drs_radial::test_radial_colocation_points ()

Tests the correctnes of the collocation points.

References drs_dims::Nr, and drs_radial::rcoll.

Referenced by test_drs_radial().

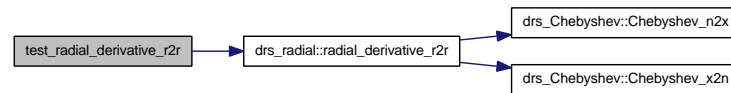
7.31.1.3 logical test_drs_radial::test_radial_derivative_r2r ()

Tests if the derivative of $r**2$ is $2*r$.

References drs_radial::radial_derivative_r2r(), drs_radial::rcoll, and drs_radial::rcoll2.

Referenced by test_drs_radial().

Here is the call graph for this function:



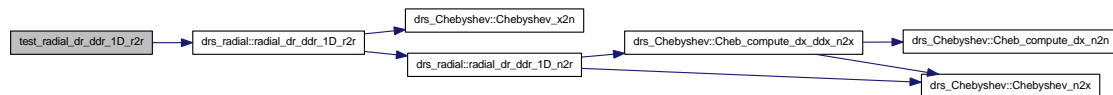
7.31.1.4 logical test_drs_radial::test_radial_dr_ddr_1D_r2r ()

Tests if the first derivative of $r**2$ is $2*r$ and the second is 2.

References `drs_radial::radial_dr_ddr_1D_r2r()`, `drs_radial::rcoll`, and `drs_radial::rcoll2`.

Referenced by `test_drs_radial()`.

Here is the call graph for this function:



7.32 tests/test_logFeature.f90 File Reference

Functions

- program `testLogFeature`

7.32.1 Function Documentation

7.32.1.1 program `testLogFeature` ()

7.33 tests/test_remesh-r2r.f90 File Reference

Functions

- program `test_drs_fftw`

7.33.1 Function Documentation

7.33.1.1 program `test_drs_fftw` ()

References `drs_fftw3::remesh()`.

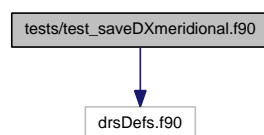
Here is the call graph for this function:



7.34 tests/test_saveDXmeridional.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for test_saveDXmeridional.f90:



Functions

- program **test_saveDXMer**
- subroutine **saveDXmeridional** (field, phi, filename)

7.34.1 Function Documentation

7.34.1.1 subroutine test_saveDXMer::saveDXmeridional (double precision,dimension(0:(blk_t_size(mpi_rank)),intent(in) *field*, double precision,intent(in) *phi*, character(len=*),intent(in) *filename*)

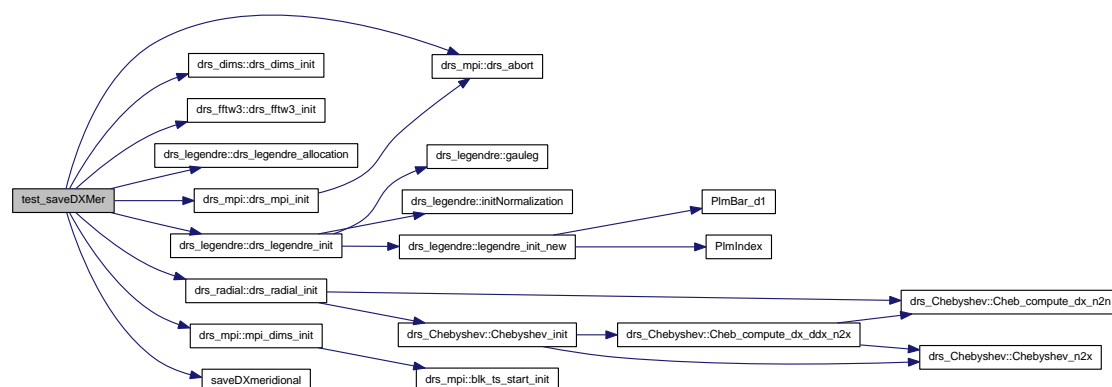
References drs_legendre::costheta, drs_dims::m0, drs_dims::Nt, drs_legendre::pi, and drs_radial::rcoll.

Referenced by drs_io_DX::save2DXscalar(), test_saveDXMer(), and YokoiPlots().

7.34.1.2 program test_saveDXMer ()

References drs_mpi::blk_t_size, drs_mpi::drs_abort(), drs_dims::drs_dims_init(), drs_fftw3::drs_fftw3_init(), drs_legendre::drs_legendre_allocation(), drs_legendre::drs_legendre_init(), drs_mpi::drs_mpi_init(), drs_radial::drs_radial_init(), drs_dims::m0, drs_mpi::mpi_dims_init(), drs_mpi::mpi_rank, drs_mpi::mpi_size, drs_dims::Np, drs_dims::Np_s, drs_dims::Nr, drs_dims::Nr_s, drs_dims::Nt, drs_dims::Nt_s, drs_legendre::pi, and saveDXmeridional().

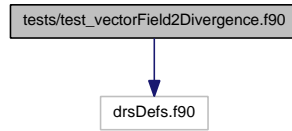
Here is the call graph for this function:



7.35 tests/test_vectorField2Divergence.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for test_vectorField2Divergence.f90:



Functions

- program **test_vectorField2Divergence**
- subroutine **init** ()

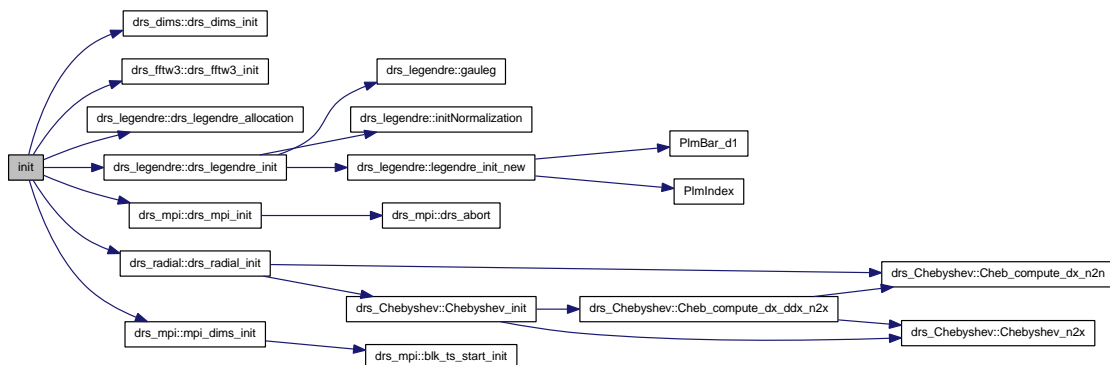
7.35.1 Function Documentation

7.35.1.1 subroutine test_vectorField2Divergence::init ()

References `drs_mpi::blk_t_size`, `drs_dims::drs_dims_init()`, `drs_fftw3::drs_fftw3_init()`, `drs_legendre::drs_legendre_allocation()`, `drs_legendre::drs_legendre_init()`, `drs_mpi::drs_mpi_init()`, `drs_radial::drs_radial_init()`, `drs_dims::lsymm`, `drs_dims::m0`, `drs_mpi::mpi_dims_init()`, `drs_mpi::mpi_rank`, `drs_dims::Np`, `drs_dims::Np_s`, `drs_dims::Nr`, `drs_dims::Nr_s`, `drs_dims::Nt`, and `drs_dims::Nt_s`.

Referenced by `Benchmarkv1()`, `Benchmarkv2()`, `drs2dx()`, `getProfile()`, `StateAverage()`, `test_vectorField2Divergence()`, and `YokoiPlots()`.

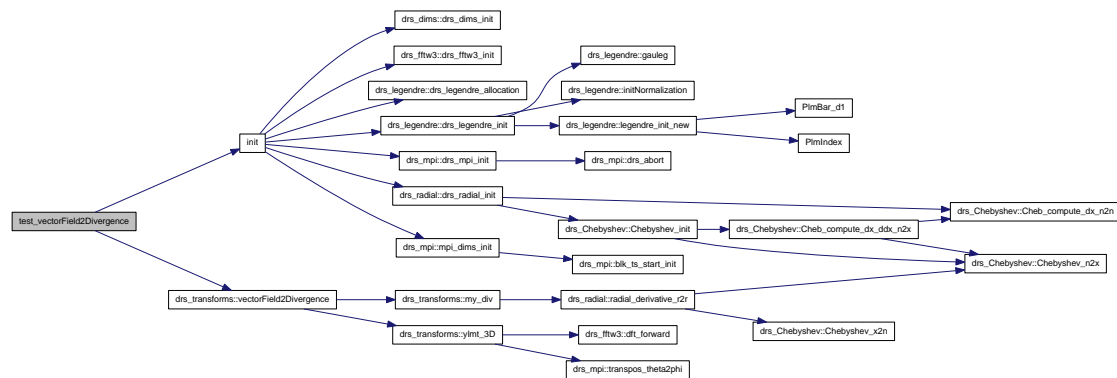
Here is the call graph for this function:



7.35.1.2 program test_vectorField2Divergence ()

References `init()`, `drs_dims::Nr`, `drs_radial::rcoll`, and `drs_transforms::vectorField2Divergence()`.

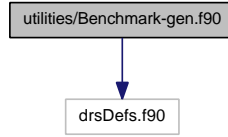
Here is the call graph for this function:



7.36 utilities/Benchmark-gen.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for Benchmark-gen.f90:



Functions

- program **Benchmarkv1**
Generate the initial conditions relevant to the 1st benchmark exercise based on a given state.
- subroutine **init** (error)

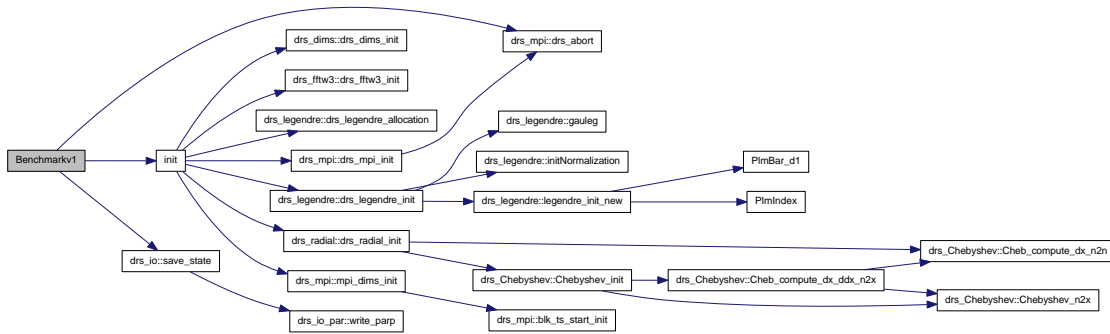
7.36.1 Function Documentation

7.36.1.1 program Benchmarkv1 ()

Generate the initial conditions relevant to the 1st benchmark exercise based on a given state.

References `drs_legendre::costheta`, `drs_mpi::drs_abort()`, `drs_field::field_pol`, `drs_field::field_tor`, `drs_flow::flow_pol`, `drs_flow::flow_tor`, `init()`, `drs_dims::Np`, `drs_dims::Nr`, `drs_dims::Nt`, `drs_legendre::pi`, `drs_radial::rcoll`, `drs_io::save_state()`, `drs_legendre::sintheta`, and `drs_temp::temp`.

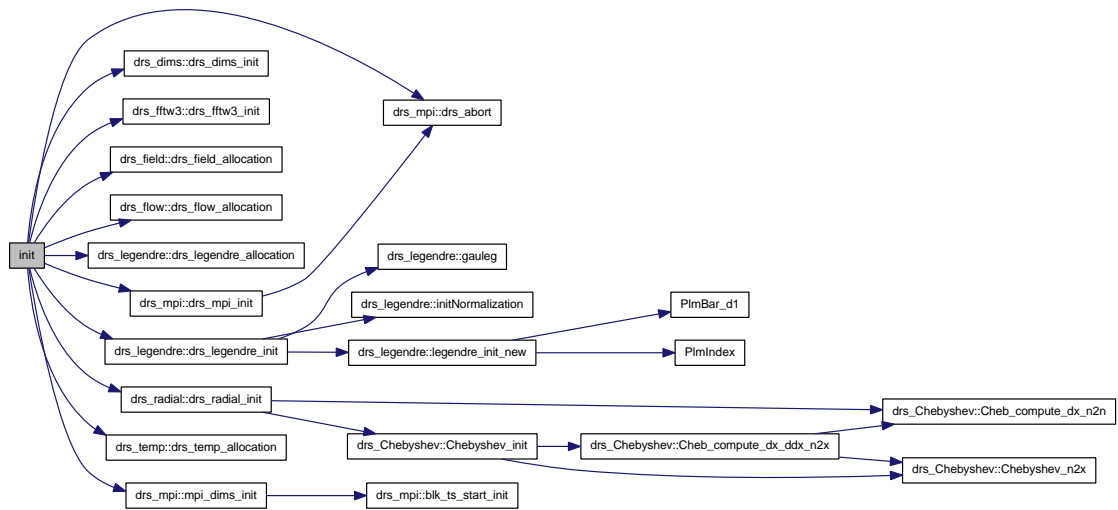
Here is the call graph for this function:



7.36.1.2 subroutine Benchmarkv1::init (integer,intent(inout) error)

References `drs_mpi::blk_t_size`, `drs_mpi::drs_abort()`, `drs_dims::drs_dims_init()`, `drs_fft3::drs_fft3_init()`, `drs_field::drs_field_allocation()`, `drs_flow::drs_flow_allocation()`, `drs_legendre::drs_legendre_allocation()`, `drs_legendre::drs_legendre_init()`, `drs_mpi::drs_mpi_init()`, `drs_radial::drs_radial_init()`, `drs_temp::drs_temp_allocation()`, `drs_io::io_calc_file_out`, `drs_dims::m0`, `drs_mpi::mpi_dims_init()`, `drs_mpi::mpi_rank`, `drs_mpi::mpi_size`, `drs_dims::Np`, `drs_dims::Np_s`, `drs_dims::Nr`, `drs_dims::Nr_s`, `drs_dims::Nt`, and `drs_dims::Nt_s`.

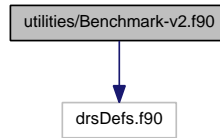
Here is the call graph for this function:



7.37 utilities/Benchmark-v2.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for Benchmark-v2.f90:



Functions

- program **Benchmarkv2**
Computes quantities relevant to the 2nd benchmark exercise based on a given state.
- subroutine **cacheTemperatureProfile** (rcut, T)
Cache the temperature from the profile.
- subroutine **selectEquatorMidShell** (field, line, rcut)
- double precision **volume** (eta)
- subroutine **init** (error)

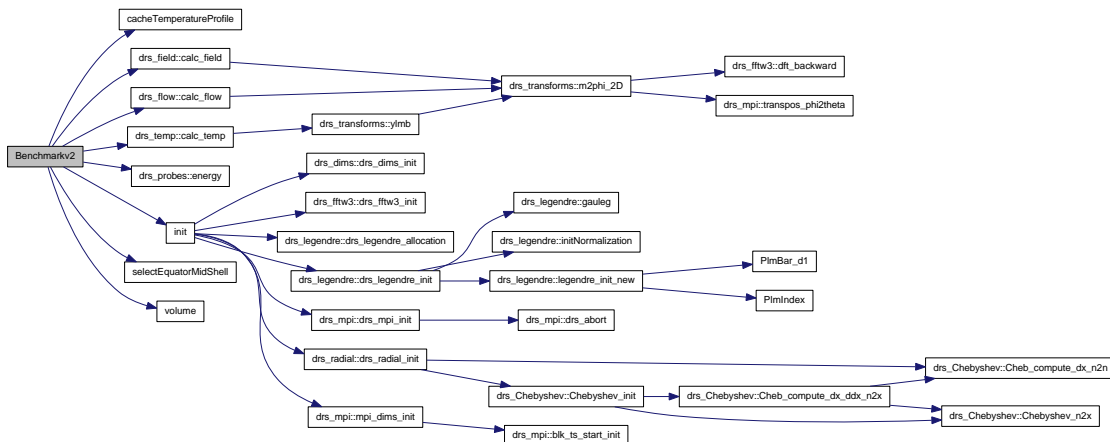
7.37.1 Function Documentation

7.37.1.1 program Benchmarkv2 ()

Computes quantities relevant to the 2nd benchmark exercise based on a given state.

References `cacheTemperatureProfile()`, `drs_field::calc_field()`, `drs_flow::calc_flow()`, `drs_temp::calc_temp()`, `drs_probes::Ekin`, `drs_probes::energy()`, `init()`, `drs_io::io_calc_file_in`, `drs_dims::Np`, `drs_dims::Np_s`, `drs_dims::Nr`, `drs_dims::Nt_s`, `drs_legendre::pi`, `selectEquatorMidShell()`, `drs_time::time`, and `volume()`.

Here is the call graph for this function:



7.37.1.2 subroutine Benchmarkv2::cacheTemperatureProfile (integer,intent(in) *rcut*, double precision,intent(out) *T*)

Cache the temperature from the profile.

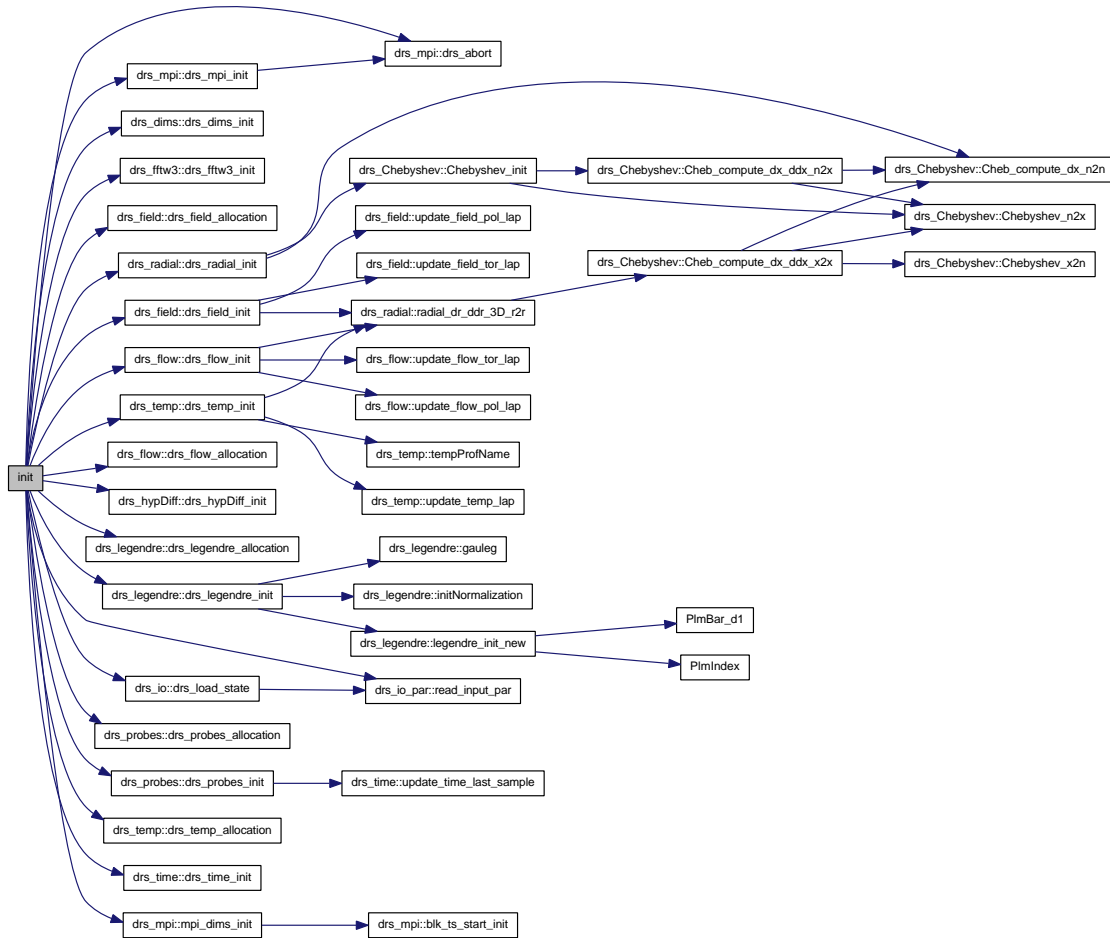
References `drs_radial::drcoll`, `drs_radial::rcoll`, and `drs_temp::temp_profile`.

Referenced by `Benchmarkv1()`, and `Benchmarkv2()`.

7.37.1.3 subroutine Benchmarkv2::init (integer,intent(inout) *error*)

References `drs_mpi::blk_t_size`, `drs_io_par::commenti`, `drs_time::drift`, `drs_io_par::drifti`, `drs_mpi::drs_abort()`, `drs_io_par::drs_calc_typei`, `drs_dims::drs_dims_init()`, `drs_fftw3::drs_fftw3_init()`, `drs_field::drs_field_allocation()`, `drs_field::drs_field_init()`, `drs_flow::drs_flow_allocation()`, `drs_flow::drs_flow_init()`, `drs_hypDiff::drs_hypDiff_init()`, `drs_legendre::drs_legendre_allocation()`, `drs_legendre::drs_legendre_init()`, `drs_io::drs_load_state()`, `drs_mpi::drs_mpi_init()`, `drs_probes::drs_probes_allocation()`, `drs_probes::drs_probes_init()`, `drs_radial::drs_radial_init()`, `drs_temp::drs_temp_allocation()`, `drs_temp::drs_temp_init()`, `drs_time::drs_time_init()`, `drs_hypDiff::drs_want_hypDiff`, `drs_io_par::etai`, `drs_field::field_pol_ddr`, `drs_field::field_pol_dr`, `drs_field::field_tor_ddr`, `drs_field::field_tor_dr`, `drs_flow::flow_pol_ddr`, `drs_flow::flow_pol_dr`, `drs_flow::flow_tor_ddr`, `drs_flow::flow_tor_dr`, `drs_io::io_calc_file_in`, `drs_io_par::lformi`, `drs_dims::m0`, `drs_io_par::m0i`, `drs_mpi::mpi_dims_init()`, `drs_mpi::mpi_rank`, `drs_mpi::mpi_size`, `drs_dims::Np`, `drs_dims::Np_s`, `drs_io_par::Npi`, `drs_io_par::Npi_s`, `drs_dims::Nr`, `drs_dims::Nr_s`, `drs_io_par::Nri`, `drs_io_par::Nri_s`, `drs_dims::Nt`, `drs_dims::Nt_s`, `drs_io_par::Nti`, `drs_io_par::Nti_s`, `drs_io_par::Pmi`, `drs_io_par::Pti`, `drs_io_par::Ra_ti`, `drs_io_par::read_input_par()`, `drs_io_par::Tai`, and `drs_time::time`.

Here is the call graph for this function:



7.37.1.4 subroutine Benchmarkv2::selectEquatorMidShell (double precision,dimension(0:(blk_t_size(mpi_rank)),intent(in) *field*, double precision,dimension(np),intent(out) *line*, integer,intent(out) *rcut*)

References drs_mpi::blk_t_start, drs_legendre::costheta, drs_radial::drcoll, drs_dims::Nt, and drs_radial::rcoll.

Referenced by Benchmarkv1(), and Benchmarkv2().

7.37.1.5 double precision Benchmarkv2::volume (double precision,intent(in) *eta*)

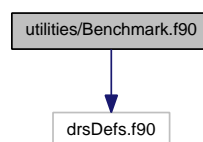
References drs_legendre::pi.

Referenced by Benchmarkv1(), Benchmarkv2(), and volume().

7.38 utilities/Benchmark.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for Benchmark.f90:



Functions

- program **Benchmarkv1**
Computes quantities relevant to the 1st benchmark exercise based on a given state.
- subroutine **cacheTemperatureProfile** (rcut, T)
Cache the temperature from the profile.
- subroutine **selectEquatorMidShell** (field, line, rcut)
- double precision **volume** (eta)
- subroutine **init** (error)

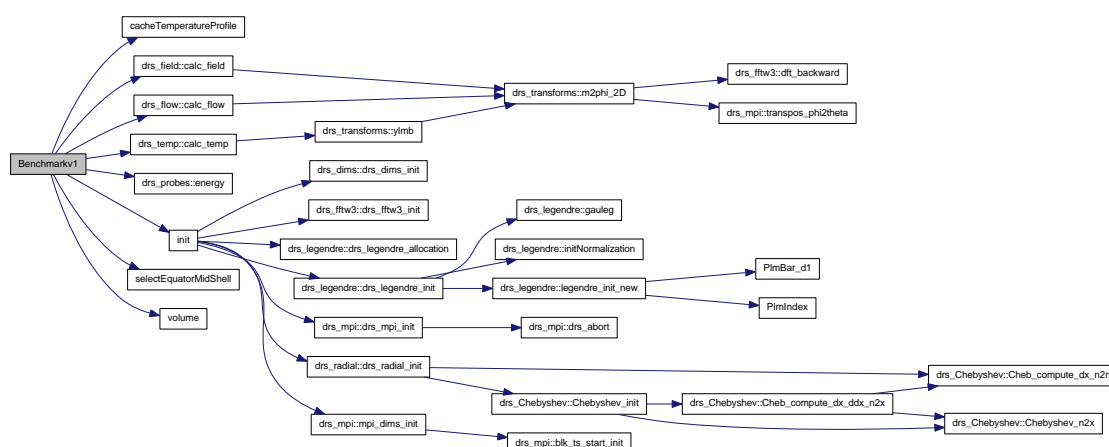
7.38.1 Function Documentation

7.38.1.1 program Benchmarkv1 ()

Computes quantities relevant to the 1st benchmark exercise based on a given state.

References `cacheTemperatureProfile()`, `drs_field::calc_field()`, `drs_flow::calc_flow()`, `drs_temp::calc_temp()`, `drs_probes::Ekin`, `drs_probes::energy()`, `init()`, `drs_io::io_calc_file_in`, `drs_dims::Np`, `drs_dims::Np_s`, `drs_dims::Nr`, `drs_dims::Nt_s`, `drs_legendre::pi`, `selectEquatorMidShell()`, `drs_time::time`, and `volume()`.

Here is the call graph for this function:



7.38.1.2 subroutine Benchmarkv1::cacheTemperatureProfile (integer,intent(in) *rcut*, double precision,intent(out) *T*)

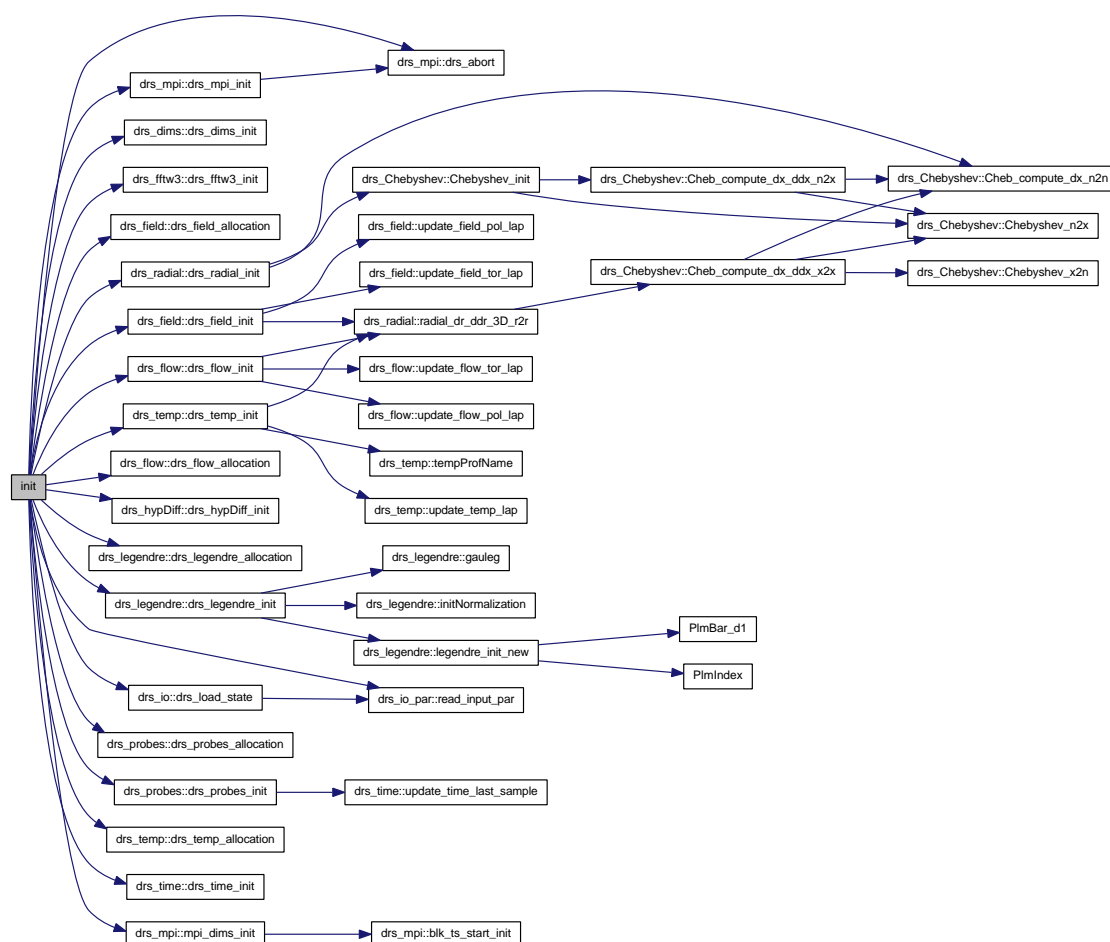
Cache the temperature from the profile.

References drs_radial::drcoll, drs_radial::rcoll, and drs_temp::temp_profile.

7.38.1.3 subroutine Benchmarkv1::init (integer,intent(inout) *error*)

References drs_mpi::blk_t_size, drs_io_par::commenti, drs_time::drift, drs_io_par::drifti, drs_mpi::drs_abort(), drs_io_par::drs_calc_typei, drs_dims::drs_dims_init(), drs_fftw3::drs_fftw3_init(), drs_field::drs_field_allocation(), drs_field::drs_field_init(), drs_flow::drs_flow_allocation(), drs_flow::drs_flow_init(), drs_hypDiff::drs_hypDiff_init(), drs_legendre::drs_legendre_allocation(), drs_legendre::drs_legendre_init(), drs_io::drs_load_state(), drs_mpi::drs_mpi_init(), drs_probes::drs_probes_allocation(), drs_probes::drs_probes_init(), drs_radial::drs_radial_init(), drs_temp::drs_temp_allocation(), drs_temp::drs_temp_init(), drs_time::drs_time_init(), drs_hypDiff::drs_want_hypDiff, drs_io_par::etai, drs_field::field_pol_ddr, drs_field::field_pol_dr, drs_field::field_tor_ddr, drs_field::field_tor_dr, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, drs_flow::flow_tor_ddr, drs_flow::flow_tor_dr, drs_io::io_calc_file_in, drs_io_par::lformi, drs_dims::m0, drs_io_par::m0i, drs_mpi::mpi_dims_init(), drs_mpi::mpi_rank, drs_mpi::mpi_size, drs_dims::Np, drs_dims::Np_s, drs_io_par::Npi, drs_io_par::Npi_s, drs_dims::Nr, drs_dims::Nr_s, drs_io_par::Nri, drs_io_par::Nri_s, drs_dims::Nt, drs_dims::Nt_s, drs_io_par::Nti, drs_io_par::Nti_s, drs_io_par::Pmi, drs_io_par::Pti, drs_io_par::Ra_ti, drs_io_par::read_input_par(), drs_io_par::Tai, and drs_time::time.

Here is the call graph for this function:



7.38.1.4 subroutine Benchmarkv1::selectEquatorMidShell (double precision,dimension(0:(blk_t_size(mpi_rank)),intent(in) *field*, double precision,dimension(np),intent(out) *line*, integer,intent(out) *rcut*)

References drs_mpi::blk_t_start, drs_legendre::costheta, drs_radial::drcoll, drs_dims::Nt, and drs_radial::rcoll.

7.38.1.5 double precision Benchmarkv1::volume (double precision,intent(in) *eta*)

References drs_legendre::pi, and volume().

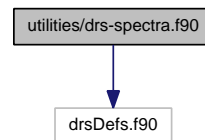
Here is the call graph for this function:



7.39 utilities/drs-spectra.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs-spectra.f90:



Functions

- program **StateAverage**

Computes the spectra of all quantities for a specific state.

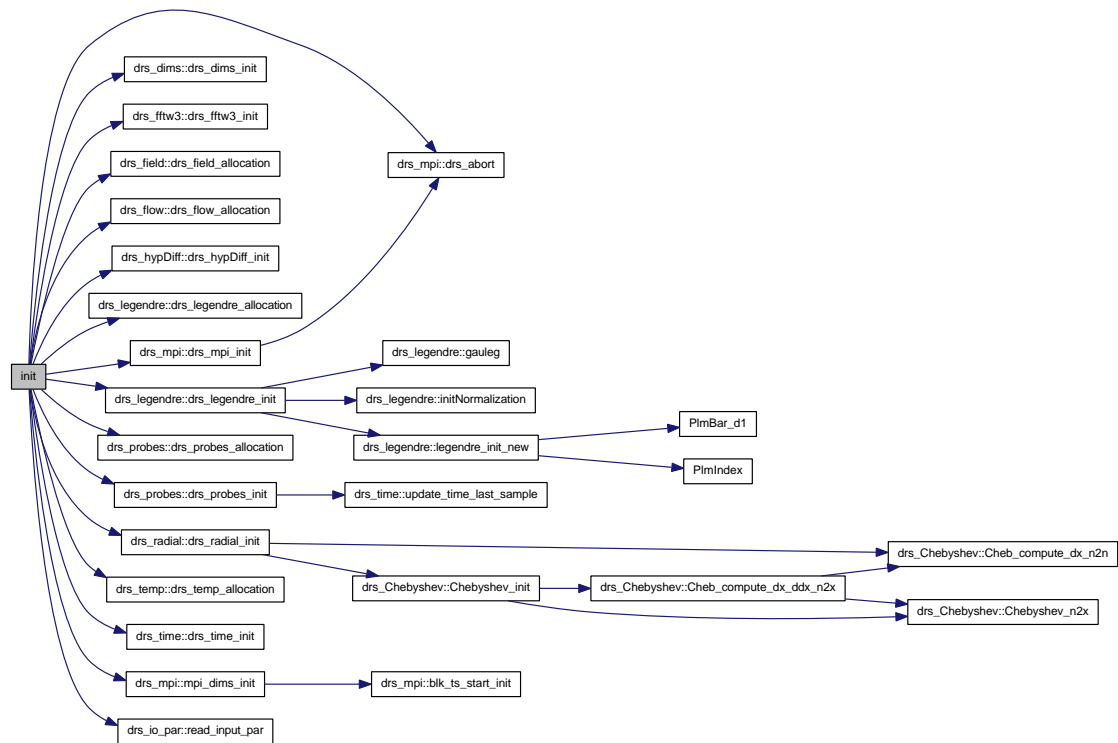
- subroutine **init** (error)

7.39.1 Function Documentation

7.39.1.1 subroutine **StateAverage::init** (integer,intent(inout) *error*)

References drs_mpi::blk_t_size, drs_io_par::commenti, drs_time::drift, drs_io_par::drifti, drs_mpi::drs_abort(), drs_io_par::drs_calc_typei, drs_dims::drs_dims_init(), drs_fftw3::drs_fftw3_init(), drs_field::drs_field_allocation(), drs_flow::drs_flow_allocation(), drs_hypDiff::drs_hypDiff_init(), drs_legendre::drs_legendre_allocation(), drs_legendre::drs_legendre_init(), drs_mpi::drs_mpi_init(), drs_probes::drs_probes_allocation(), drs_probes::drs_probes_init(), drs_radial::drs_radial_init(), drs_temp::drs_temp_allocation(), drs_time::drs_time_init(), drs_hypDiff::drs_want_hypDiff, drs_io_par::etai, drs_io::io_calc_file_in, drs_io_par::lformi, drs_dims::m0, drs_io_par::m0i, drs_mpi::mpi_dims_init(), drs_mpi::mpi_rank, drs_mpi::mpi_size, drs_dims::Np, drs_dims::Np_s, drs_io_par::Npi, drs_io_par::Npi_s, drs_dims::Nr, drs_dims::Nr_s, drs_io_par::Nri, drs_io_par::Nri_s, drs_dims::Nt, drs_dims::Nt_s, drs_io_par::Nti, drs_io_par::Nti_s, drs_io_par::Pmi, drs_io_par::Pti, drs_io_par::Ra_ti, drs_io_par::read_input_par(), drs_io_par::Tai, and drs_time::time.

Here is the call graph for this function:

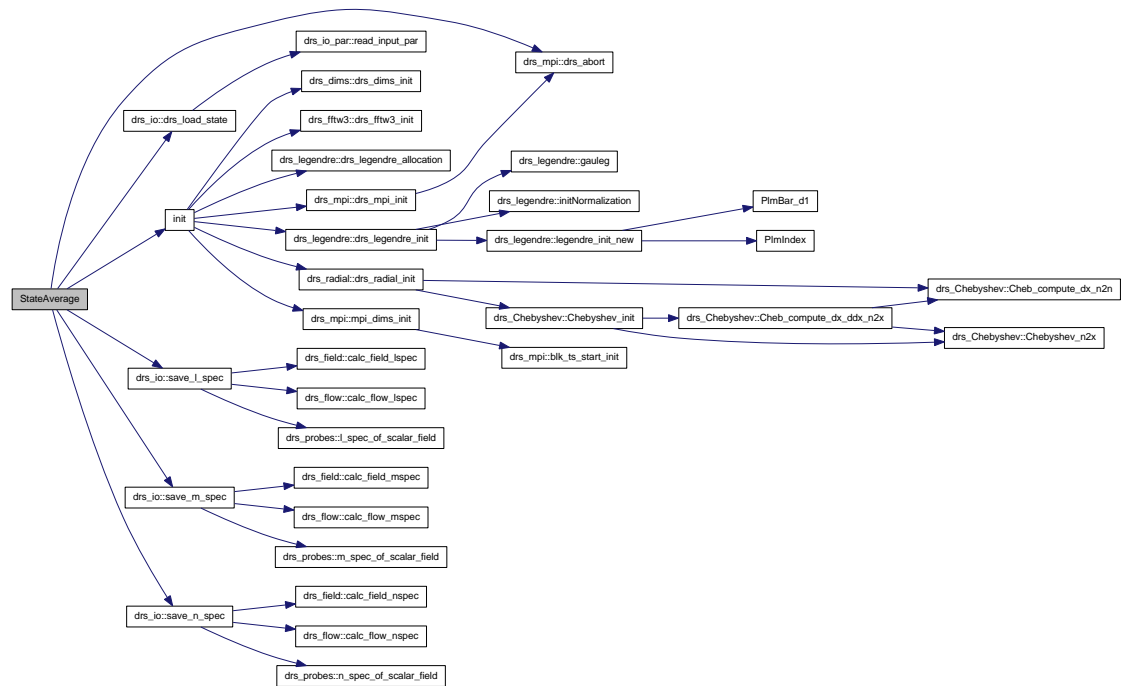


7.39.1.2 program StateAverage ()

Computes the spectra of all quantities for a specific state.

References `drs_io::deflate`, `drs_mpi::drs_abort()`, `drs_io::drs_load_state()`, `drs_io::inflate`, `init()`, `drs_io::io_calc_file_in`, `drs_io::io_calc_file_out`, `drs_io::save_l_spec()`, `drs_io::save_m_spec()`, and `drs_io::save_n_spec()`.

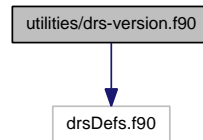
Here is the call graph for this function:



7.40 utilities/drs-version.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs-version.f90:



Functions

- program **drsVersion**

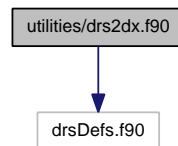
7.40.1 Function Documentation

7.40.1.1 program **drsVersion** ()

7.41 utilities/drs2dx.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs2dx.f90:



Functions

- program **drs2dx**

Computes the plots require for the Yokoi paper.

- subroutine **init** (error)
- subroutine **parse_drs2dx** ()
- subroutine **redefine_radial_coordinate** ()

7.41.1 Function Documentation

7.41.1.1 program drs2dx ()

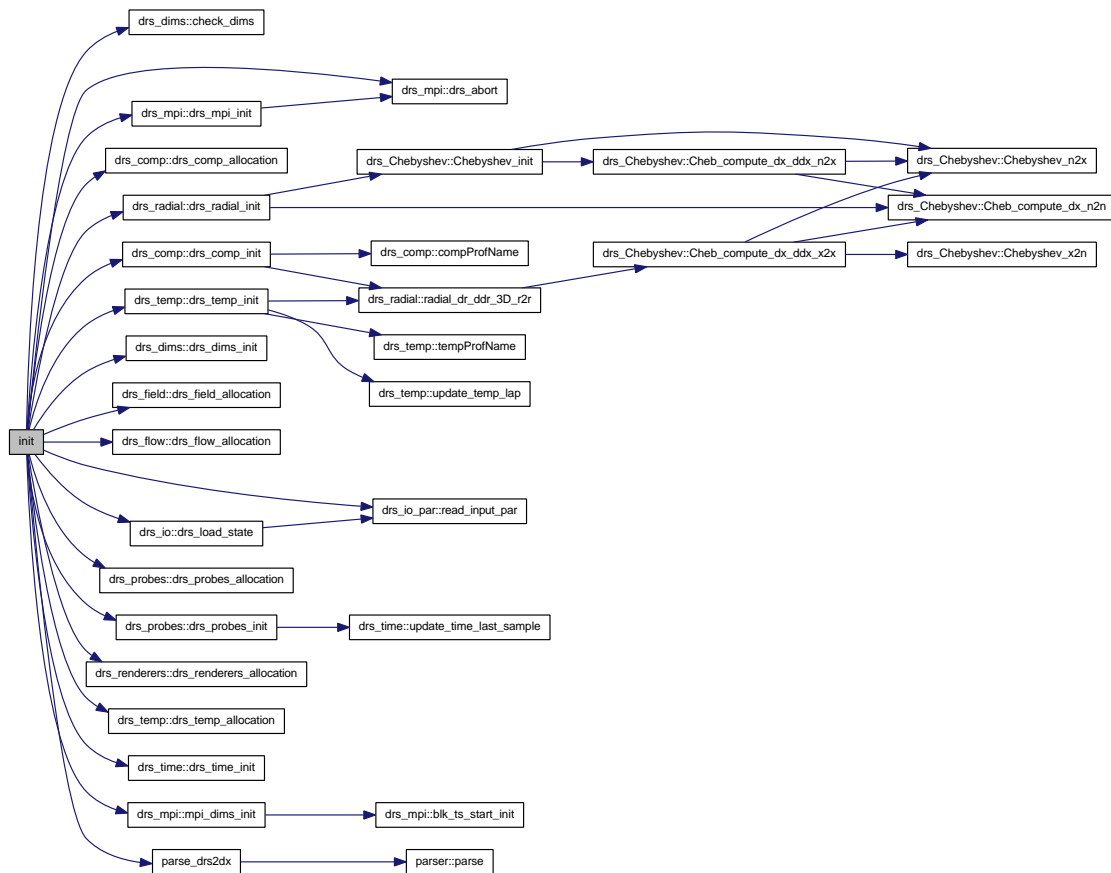
Computes the plots require for the Yokoi paper.

References `init()`, `drs_io::io_calc_file_in`, `redefine_radial_coordinate()`, `drs_renderers::render()`, `drs_renderers::render_out`, `drs_renderers::XX`, `drs_renderers::YY`, and `drs_renderers::ZZ`.

[illegible]

References drs_mpi::blk_t_size, drs_dims::check_dims(), drs_io::deflate, drs_time::drift, drs_io_par::drift, drs_mpi::drs_abort(), drs_io_par::drs_calc_typei, drs_comp::drs_comp_allocation(), drs_comp::drs_comp_init(), drs_dims::drs_dims_init(), drs_field::drs_field_allocation(), drs_flow::drs_flow_allocation(), drs_io::drs_load_state(), drs_mpi::drs_mpi_init(), drs_probes::drs_probes_allocation(), drs_probes::drs_probes_init(), drs_radial::drs_radial_init(), drs_renderers::drs_renderers_allocation(), drs_temp::drs_temp_allocation(), drs_temp::drs_temp_init(), drs_time::drs_time_init(), drs_io_par::etai, drs_io::inflate, drs_io::io_calc_file_in, drs_io_par::lformi, drs_dims::m0, drs_io_par::m0i, drs_mpi::mpi_dims_init(), drs_mpi::mpi_rank, drs_mpi::mpi_size, drs_dims::Np, drs_dims::Np_s, drs_io_par::Npi_s, drs_dims::Nr, drs_dims::Nr_s, drs_io_par::Nri_s, drs_dims::Nt, drs_dims::Nt_s, drs_io_par::Nti_s, parse_dr2dx(), drs_io_par::Pmi, drs_io_par::Pti, drs_io_par::Ra_ti, drs_io_par::read_input_par(), drs_io_par::Tai, and drs_time::time.

Here is the call graph for this function:



7.41.1.3 subroutine drs2dx::parse_drs2dx ()

References drs_io_DX::cut_type, drs_io::io_calc_file_in, drs_dims::Np, drs_dims::Nr, drs_dims::Nt, parser::parse(), and drs_io_DX::where_to_cut.

Referenced by init().

Here is the call graph for this function:



7.41.1.4 subroutine drs2dx::redefine_radial_coordinate ()

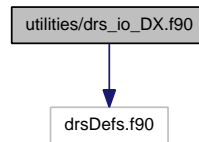
References drs_radial::drcoll, drs_dims::Nr, drs_radial::rcoll, and drs_radial::rcoll2.

Referenced by drs2dx().

7.42 utilities/drs_io_DX.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs_io_DX.f90:



Classes

- interface **drs_io_DX::save2DX**

Namespaces

- namespace **drs_io_DX**

Functions

- subroutine **drs_io_DX::save2DXscalar** (field, filename)
Saves the contents of a scalar field to file.
- subroutine **drs_io_DX::save2DXvector** (XX, YY, ZZ, filename)
Saves the contents of a vector field to file given its three components.
- subroutine **drs_io_DX::saveDXmeridional** (field, filename)
Writes a meridional slice of the field.
- subroutine **drs_io_DX::saveDXmeridional3DVec** (field_x, field_y, field_z, filename)
Writes a meridional slice of the field.
- subroutine **drs_io_DX::saveDXvolume** (field, filename)
Writes a volume render of the field.
- subroutine **drs_io_DX::saveDXvolume_v2** (field, filename)
Writes a volume render of the field.
- subroutine **drs_io_DX::saveDXvolume3DVec** (XX, YY, ZZ, filename)
Writes a volume render of the vector field components.

Variables

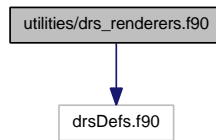
- double precision **drs_io_DX::cut_phi**
the azimuth to use on meridional cuts

- double precision **drs_io_DX::cut_z**
the azimuth to use on equator parallell cuts
- double precision **drs_io_DX::where_to_cut** = 0.0d0
- integer **drs_io_DX::cut_type**
the type of cut or render to save

7.43 utilities/drs_renderers.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs_renderers.f90:



Namespaces

- namespace **drs_renderers**

Defines

- #define **ERR_UNSUPPORTED_OPTION** 33

Functions

- subroutine **drs_renderers::drs_renderers_allocation** (what)
- subroutine **drs_renderers::render** (what)

Makes a decision about what to render.

Numbers are coded as:

~~~~~ a b c d e | | | | > e - component 1, 2 or 3 for vectors, irrelevant for scalars | | | > d - coordinate system or stream lines | | > c - quantity to be plotted | > b - curl, gradient or divergence or 0 | > a - scalar product with selection or 0.

- subroutine **drs_renderers::render_ur** ()
- subroutine **drs_renderers::render_u** ()
- subroutine **drs_renderers::render_Br** ()
- subroutine **drs_renderers::render_Bt** ()
- subroutine **drs_renderers::render_Bp** ()

Renders the azimuthal component of the magnetic field.

- subroutine **drs_renderers::render_Bz** ()

Renders the z component of the magnetic field.

- subroutine **drs_renderers::render_B** ()

Render all three spherical components of the magnetic field.

- subroutine **drs_renderers::render_B_outside** ()

Render all three spherical components of the magnetic field outside the outer core.

- subroutine **drs_renderers::render_rotu_r** ()

Renders the radial component of the curl of the flow.

- subroutine **drs_renderers::render_rotu** ()
Renders all three spherical components of the curl of the flow (vorticity).
- subroutine **drs_renderers::render_up** ()
u_phi:
- subroutine **drs_renderers::render_rotu_p** ()
rot(u)_phi:
- subroutine **drs_renderers::render_ut** ()
u_theta:
- subroutine **drs_renderers::render_rotu_t** ()
rot(u)_theta:
- subroutine **drs_renderers::render_uz** ()
u_z
- subroutine **drs_renderers::render_rotu_z** ()
rot(u)_z:
- subroutine **drs_renderers::render_temperature_perturbation** ()
Renders the temperature perturbation.
- subroutine **drs_renderers::render_temperature** ()
Renders the total temperature.
- subroutine **drs_renderers::render_helicity** ()
Renders helicity.
- subroutine **drs_renderers::render_temperature_grad_r** ()
- subroutine **drs_renderers::render_streamlines_t** ()
- subroutine **drs_renderers::render_poloidal_streamlines** ()
Renders the poloidal flow streamlines.
- subroutine **drs_renderers::render_radial_streamfunction** ()
radial stream function for the flow

Variables

- double precision, dimension(;;,:), allocatable **drs_renderers::render_out**
- double precision, allocatable **drs_renderers::XX**
- double precision, allocatable **drs_renderers::YY**
- double precision, allocatable **drs_renderers::ZZ**

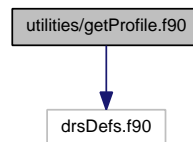
7.43.1 Define Documentation

7.43.1.1 #define ERR_UNSUPPORTED_OPTION 33

7.44 utilities/getProfile.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for getProfile.f90:



Functions

- program **getProfile**

Computes the horizontally integrated radial profile for the given quantity.

- subroutine **init** (error)
- subroutine **parseConfig** (error)

Parses the configuration file ~~~~~ state = <state base="" name>=""> what = quantity to generate a profile for ~~~~~.

- subroutine **setWhatName** ()

Sets a whuman readable name for what is being computed.

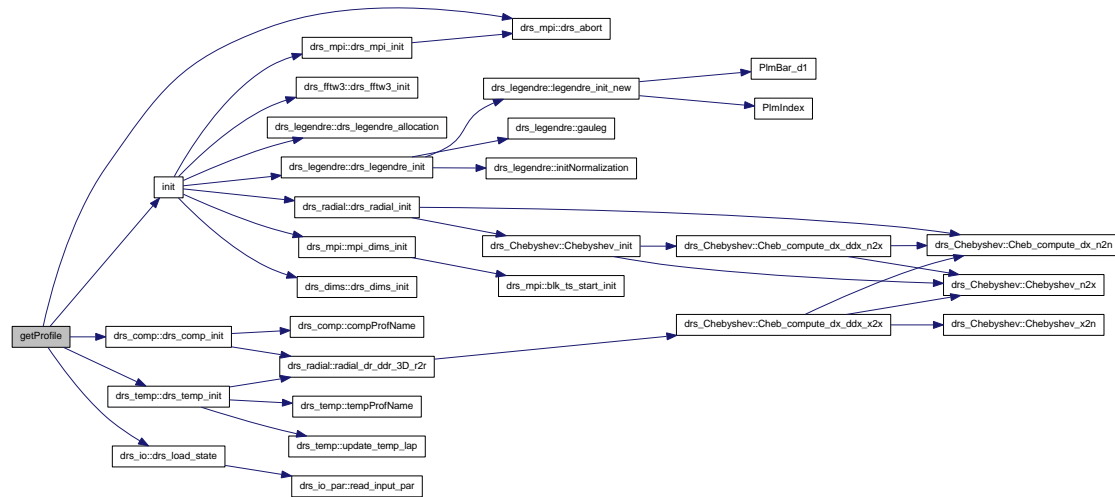
7.44.1 Function Documentation

7.44.1.1 program getProfile ()

Computes the horizontally integrated radial profile for the given quantity.

References drs_comp::comp, drs_comp::comp_profile, drs_io::deflate, drs_mpi::drs_abort(), drs_comp::drs_comp_init(), drs_io::drs_load_state(), drs_temp::drs_temp_init(), drs_io::inflate, init(), drs_io::io_calc_file_in, drs_dims::Nr, drs_radial::rcoll, drs_temp::temp, and drs_temp::temp_profile.

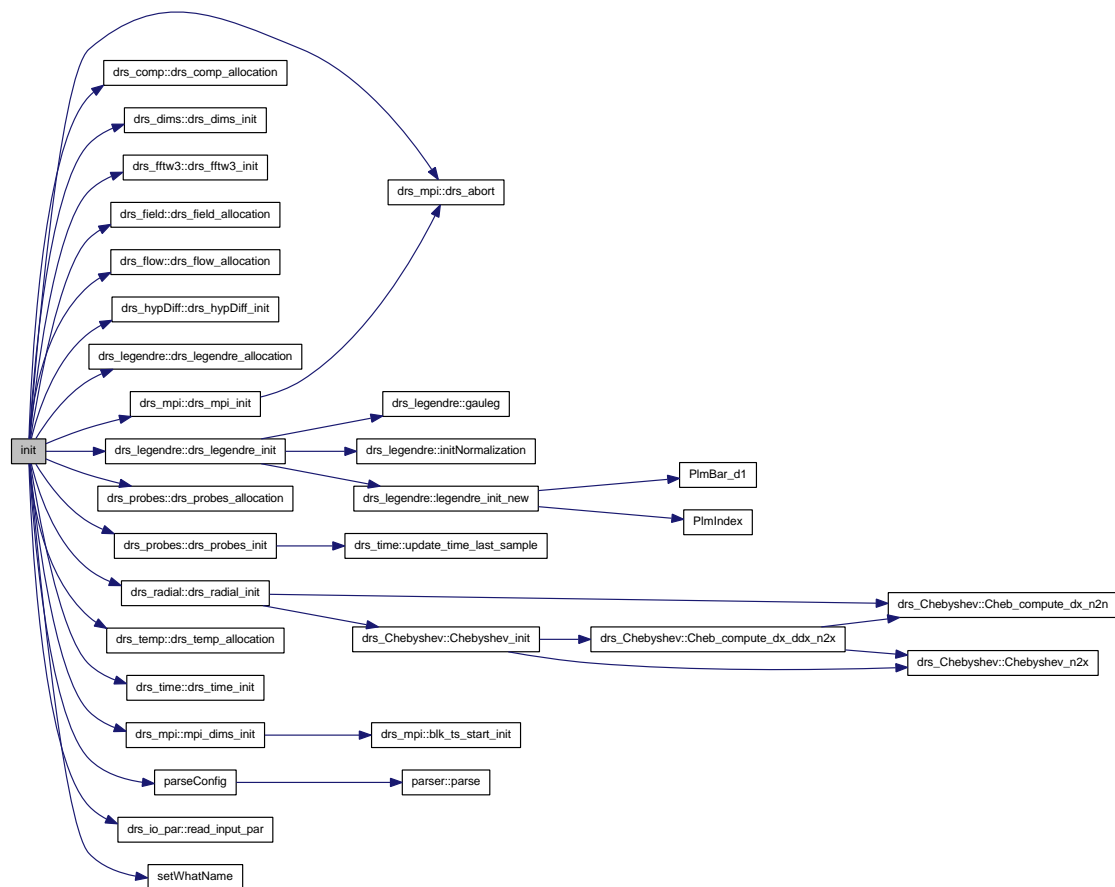
Here is the call graph for this function:



7.44.1.2 subroutine getProfile::init (integer,intent(inout) error)

References drs_mpi::blk_t_size, drs_io_par::commenti, drs_time::drift, drs_io_par::drifti, drs_mpi::drs_abort(), drs_io_par::drs_calc_typei, drs_comp::drs_comp_allocation(), drs_dims::drs_dims_init(), drs_fftw3::drs_fftw3_init(), drs_field::drs_field_allocation(), drs_flow::drs_flow_allocation(), drs_hypDiff::drs_hypDiff_init(), drs_legendre::drs_legendre_allocation(), drs_legendre::drs_legendre_init(), drs_mpi::drs_mpi_init(), drs_probes::drs_probes_allocation(), drs_probes::drs_probes_init(), drs_radial::drs_radial_init(), drs_temp::drs_temp_allocation(), drs_time::drs_time_init(), drs_hypDiff::drs_want_hypDiff, drs_io_par::etai, drs_io::io_calc_file_in, drs_io_par::lformi, drs_dims::m0, drs_io_par::m0i, drs_mpi::mpi_dims_init(), drs_mpi::mpi_rank, drs_mpi::mpi_size, drs_dims::Np, drs_dims::Np_s, drs_io_par::Npi, drs_io_par::Npi_s, drs_dims::Nr, drs_dims::Nr_s, drs_io_par::Nri, drs_io_par::Nri_s, drs_dims::Nt, drs_dims::Nt_s, drs_io_par::Nti, drs_io_par::Nti_s, parseConfig(), drs_io_par::Pmi, drs_io_par::Pti, drs_io_par::Ra_ti, drs_io_par::read_input_par(), setWhatName(), drs_io_par::Tai, and drs_time::time.

Here is the call graph for this function:



7.44.1.3 subroutine getProfile::parseConfig (integer error)

Parses the configuration file ~~~~ state = <state base="" name>=""> what = quantity to generate a profile for ~~~~.

References drs_io::io_calc_file_in, and parser::parse().

Referenced by init().

Here is the call graph for this function:



7.44.1.4 subroutine getProfile::setWhatName ()

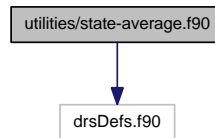
Sets a whuman readable name for what is being computed.

Referenced by init().

7.45 utilities/state-average.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for state-average.f90:



Functions

- program **StateAverage**

Takes an Average in time of all fields.

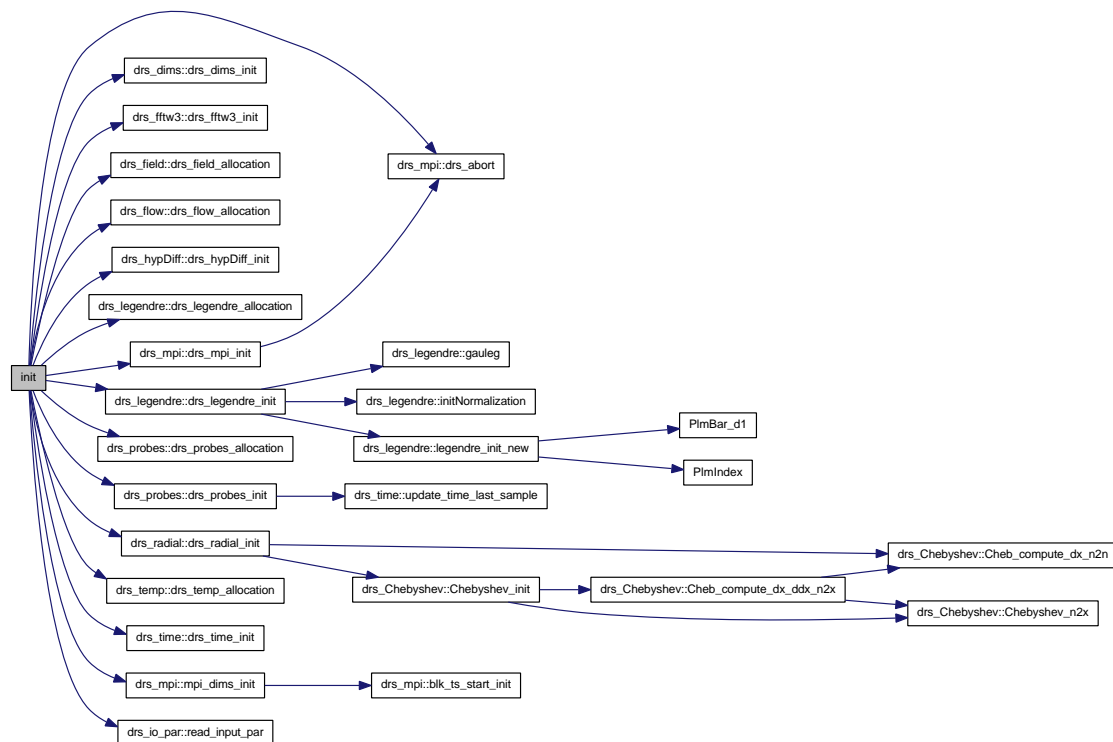
- subroutine **init** (error)

7.45.1 Function Documentation

7.45.1.1 subroutine **StateAverage::init** (integer,intent(inout) *error*)

References drs_mpi::blk_t_size, drs_io_par::commenti, drs_time::drift, drs_io_par::drifti, drs_mpi::drs_abort(), drs_io_par::drs_calc_typei, drs_dims::drs_dims_init(), drs_fftw3::drs_fftw3_init(), drs_field::drs_field_allocation(), drs_flow::drs_flow_allocation(), drs_hypDiff::drs_hypDiff_init(), drs_legendre::drs_legendre_allocation(), drs_legendre::drs_legendre_init(), drs_mpi::drs_mpi_init(), drs_probes::drs_probes_allocation(), drs_probes::drs_probes_init(), drs_radial::drs_radial_init(), drs_temp::drs_temp_allocation(), drs_time::drs_time_init(), drs_hypDiff::drs_want_hypDiff, drs_io_par::etai, drs_io::io_calc_file_in, drs_io_par::lformi, drs_dims::m0, drs_io_par::m0i, drs_mpi::mpi_dims_init(), drs_mpi::mpi_rank, drs_mpi::mpi_size, drs_dims::Np, drs_dims::Np_s, drs_io_par::Npi, drs_io_par::Npi_s, drs_dims::Nr, drs_dims::Nr_s, drs_io_par::Nri, drs_io_par::Nri_s, drs_dims::Nt, drs_dims::Nt_s, drs_io_par::Nti, drs_io_par::Nti_s, drs_io_par::Pmi, drs_io_par::Pti, drs_io_par::Ra_ti, drs_io_par::read_input_par(), drs_io_par::Tai, and drs_time::time.

Here is the call graph for this function:

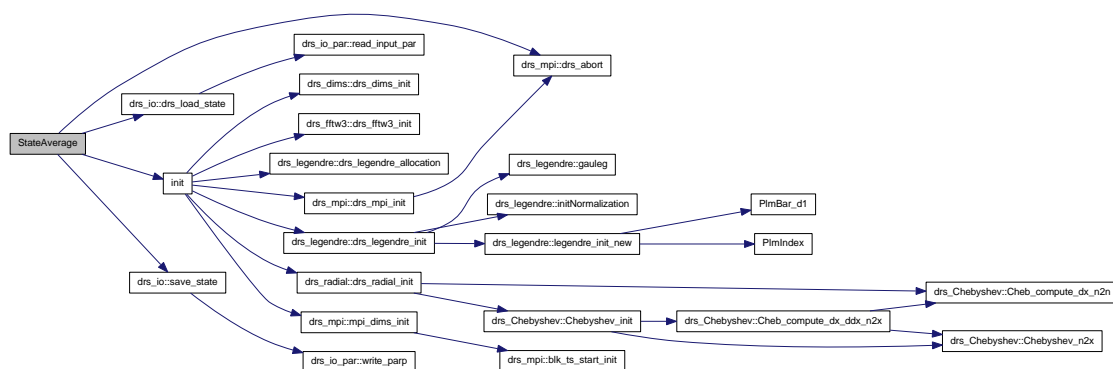


7.45.1.2 program StateAverage ()

Takes an Average in time of all fields.

References `drs_io::deflate`, `drs_mpi::drs_abort()`, `drs_io::drs_load_state()`, `drs_field::field_pol`, `drs_field::field_pol_avg`, `drs_field::field_tor`, `drs_field::field_tor_avg`, `drs_flow::flow_pol`, `drs_flow::flow_pol_avg`, `drs_flow::flow_tor`, `drs_flow::flow_tor_avg`, `drs_io::inflate`, `init()`, `drs_io::io_calc_file_in`, `drs_io::io_calc_file_out`, `drs_io::save_state()`, `drs_temp::temp`, and `drs_temp::temp_avg`.

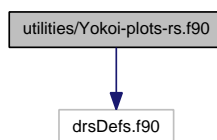
Here is the call graph for this function:



7.46 utilities/Yokoi-plots-rs.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for Yokoi-plots-rs.f90:



Functions

- program **StateAverage**

Computes the plots require for the Yokoi paper.

- subroutine **init** (error)
- subroutine **computeAndSaveAverage** (nstates)
- subroutine **computeEMF** (EMF)
- subroutine **computeAlpha** (alpha)
- subroutine **computeBeta** (beta)
- subroutine **computeGamma** (gamma)
- subroutine **saveIDLmeridional** (field, phi, filename)

Save the requested meridional cut in a format compatible with Radostin's IDL scripts.

7.46.1 Function Documentation

7.46.1.1 subroutine **StateAverage::computeAlpha** (double precision,dimension(0:(blk_t_size(mpi_rank),intent(inout) *alpha*)

References `drs_dims::Nt`.

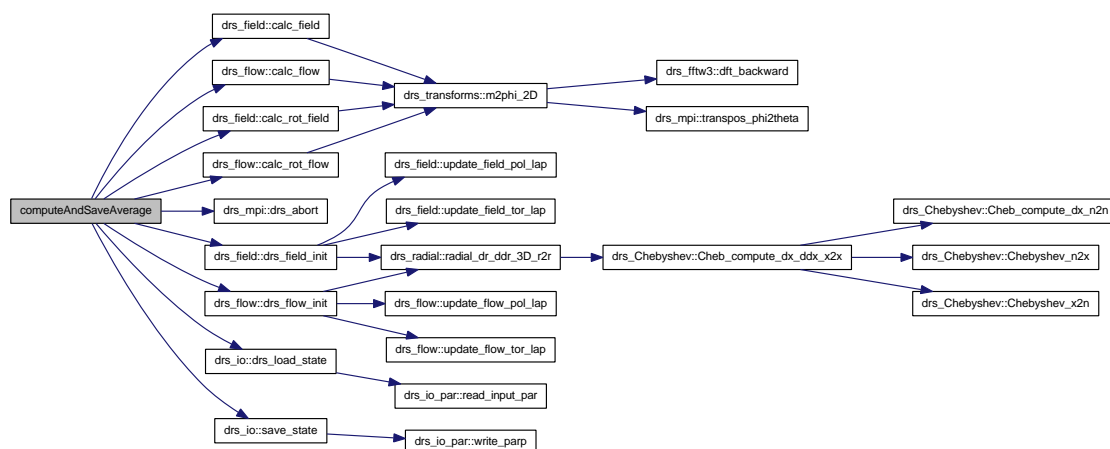
Referenced by `StateAverage()`, and `YokoiPlots()`.

7.46.1.2 subroutine **StateAverage::computeAndSaveAverage** (integer,intent(out) *nstates*)

References `drs_field::calc_field()`, `drs_flow::calc_flow()`, `drs_field::calc_rot_field()`, `drs_flow::calc_rot_flow()`, `drs_io::deflate`, `drs_mpi::drs_abort()`, `drs_field::drs_field_init()`, `drs_flow::drs_flow_init()`, `drs_io::drs_load_state()`, `drs_field::field_pol`, `drs_field::field_pol_avg`, `drs_field::field_pol_ddr`, `drs_field::field_pol_dr`, `drs_field::field_tor`, `drs_field::field_tor_avg`, `drs_field::field_tor_ddr`, `drs_field::field_tor_dr`, `drs_flow::flow_pol`, `drs_flow::flow_pol_avg`, `drs_flow::flow_pol_ddr`, `drs_flow::flow_pol_dr`, `drs_flow::flow_tor`, `drs_flow::flow_tor_avg`, `drs_flow::flow_tor_ddr`, `drs_flow::flow_tor_dr`, `drs_io::inflate`, `drs_io::io_calc_file_in`, `drs_io::io_calc_file_out`, `drs_dims::Np_s`, and `drs_io::save_state()`.

Referenced by `StateAverage()`, and `YokoiPlots()`.

Here is the call graph for this function:



7.46.1.3 subroutine StateAverage::computeBeta (double precision,dimension(0:(blk_t_size(mpi_rank),intent(inout) beta)

References drs_dims::Nt.

Referenced by StateAverage(), and YokoiPlots().

7.46.1.4 subroutine StateAverage::computeEMF (double precision,dimension(0:(blk_t_size(mpi_rank),intent(inout) EMF)

References drs_legendre::costheta, and drs_dims::Nt.

Referenced by StateAverage(), and YokoiPlots().

7.46.1.5 subroutine StateAverage::computeGamma (double precision,dimension(0:(blk_t_size(mpi_rank),intent(inout) gamma)

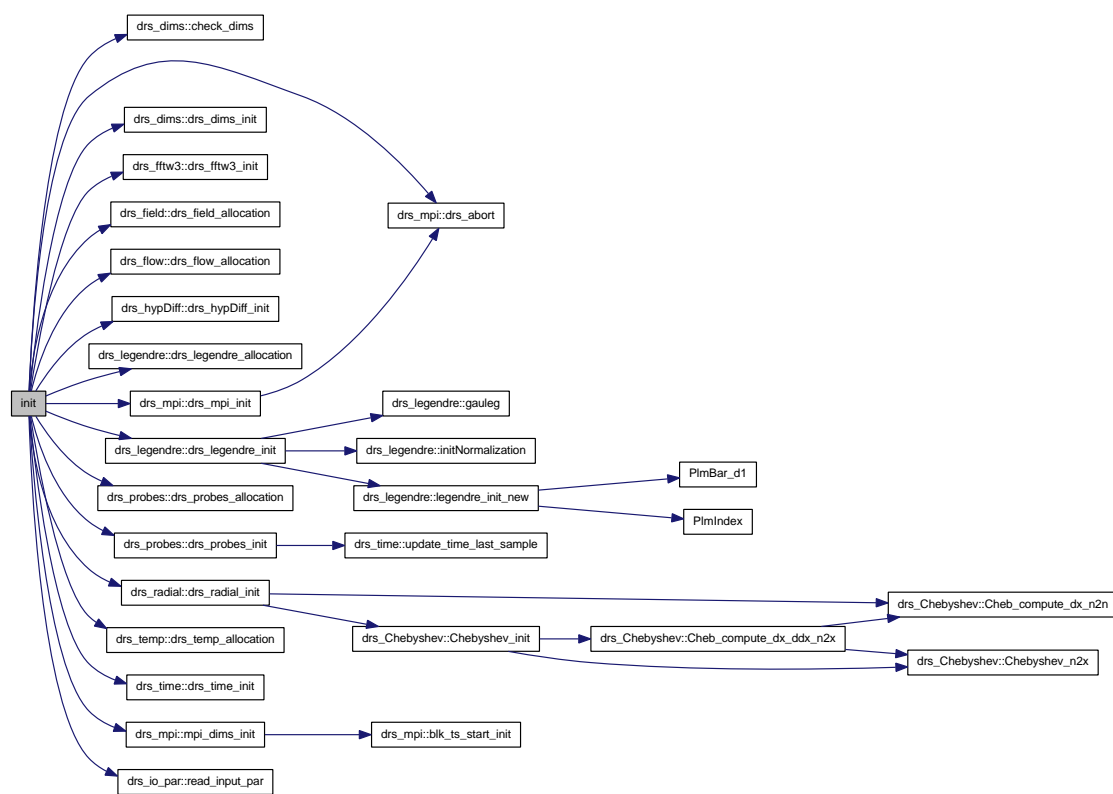
References drs_dims::Nt.

Referenced by StateAverage(), and YokoiPlots().

7.46.1.6 subroutine StateAverage::init (integer,intent(inout) error)

References drs_mpi::blk_t_size, drs_dims::check_dims(), drs_io_par::commenti, drs_time::drift, drs_io_par::drifti, drs_mpi::drs_abort(), drs_io_par::drs_calc_typei, drs_dims::drs_dims_init(), drs_fftw3::drs_fftw3_init(), drs_field::drs_field_allocation(), drs_flow::drs_flow_allocation(), drs_hypDiff::drs_hypDiff_init(), drs_legendre::drs_legendre_allocation(), drs_legendre::drs_legendre_init(), drs_mpi::drs_mpi_init(), drs_probes::drs_probes_allocation(), drs_probes::drs_probes_init(), drs_radial::drs_radial_init(), drs_temp::drs_temp_allocation(), drs_time::drs_time_init(), drs_hypDiff::drs_want_hypDiff, drs_io_par::etai, drs_io::io_calc_file_in, drs_io_par::lformi, drs_dims::m0, drs_io_par::m0i, drs_mpi::mpi_dims_init(), drs_mpi::mpi_rank, drs_mpi::mpi_size, drs_dims::Np, drs_dims::Np_s, drs_io_par::Npi, drs_io_par::Npi_s, drs_dims::Nr, drs_dims::Nr_s, drs_io_par::Nri, drs_io_par::Nri_s, drs_dims::Nt, drs_dims::Nt_s, drs_io_par::Nti, drs_io_par::Nti_s, drs_io_par::Pmi, drs_io_par::Pti, drs_io_par::Ra_ti, drs_io_par::read_input_par(), drs_io_par::Tai, and drs_time::time.

Here is the call graph for this function:



7.46.1.7 subroutine StateAverage::saveIDLmeridional (double precision,dimension(0:(blk_t_-size(mpi_rank),intent(in) *field*, double precision,intent(in) *phi*, character(len=*),intent(in) *filename*)

Save the requested meridional cut in a format compatible with Radostin's IDL scripts.

References drs_legendre::costheta, drs_probes::dOmega, drs_dims::m0, drs_dims::Nt, drs_legendre::pi, and drs_radial::rcoll.

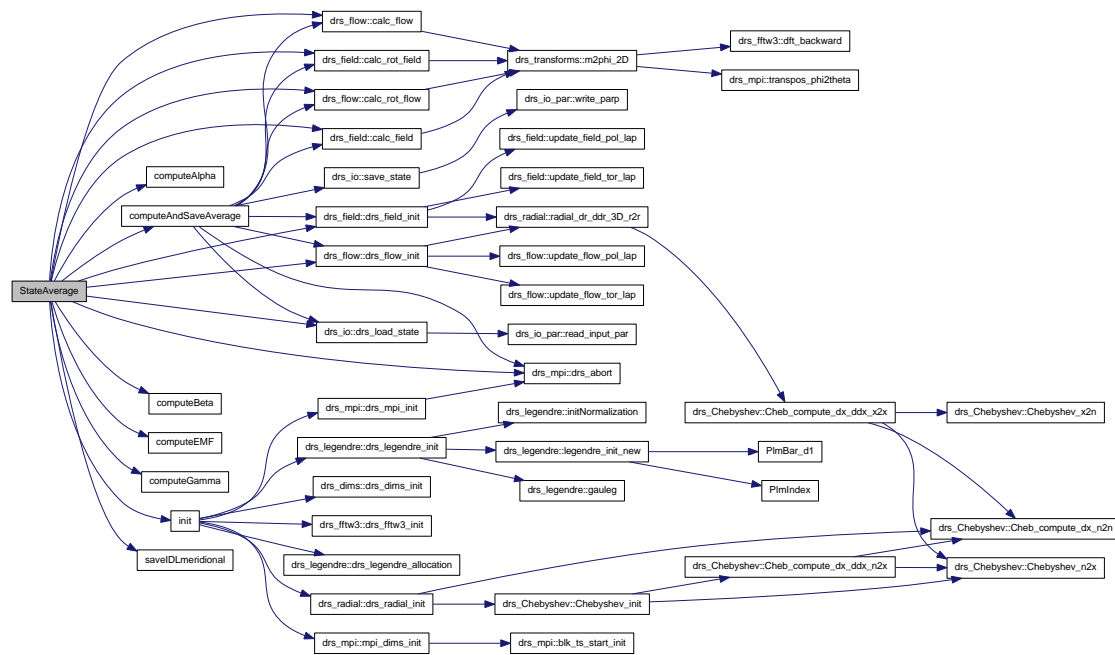
Referenced by StateAverage().

7.46.1.8 program StateAverage ()

Computes the plots require for the Yokoi paper.

References drs_field::calc_field(), drs_flow::calc_flow(), drs_field::calc_rot_field(), drs_flow::calc_rot_flow(), computeAlpha(), computeAndSaveAverage(), computeBeta(), computeEMF(), computeGamma(), drs_legendre::costheta, drs_io::deflate, drs_mpi::drs_abort(), drs_field::drs_field_init(), drs_flow::drs_flow_init(), drs_io::drs_load_state(), drs_field::field_pol, drs_field::field_pol_avg, drs_field::field_pol_ddr, drs_field::field_pol_dr, drs_field::field_tor, drs_field::field_tor_avg, drs_field::field_tor_ddr, drs_field::field_tor_dr, drs_flow::flow_pol, drs_flow::flow_pol_avg, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, drs_flow::flow_tor, drs_flow::flow_tor_avg, drs_flow::flow_tor_ddr, drs_flow::flow_tor_dr, drs_io::inflate, init(), drs_io::io_calc_file_in, drs_dims::Np, drs_dims::Nr, drs_dims::Nt, and saveIDLmeridional().

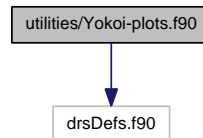
Here is the call graph for this function:



7.47 utilities/Yokoi-plots.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for Yokoi-plots.f90:



Functions

- program **YokoiPlots**
Computes the plots require for the Yokoi paper.
- subroutine **init** (error)
- subroutine **computeAndSaveAverage** (nstates)
- subroutine **computeEMF** (EMF)
- subroutine **computeAlpha** (alpha)
- subroutine **computeBeta** (beta)
- subroutine **computeGamma** (gamma)
- subroutine **saveDXmeridional** (field, phi, filename)

7.47.1 Function Documentation

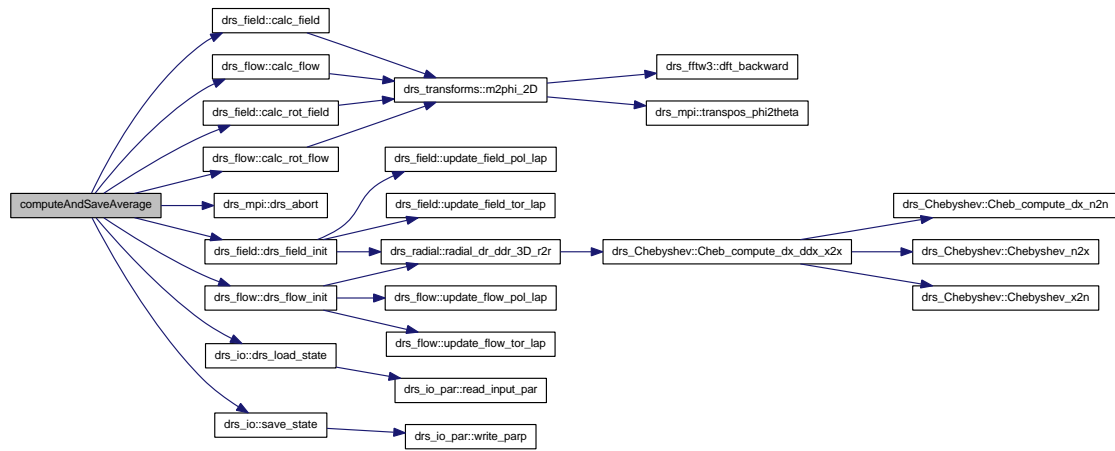
7.47.1.1 subroutine YokoiPlots::computeAlpha (double precision,dimension(0:(blk_t_size(mpi_rank),intent(inout) *alpha*)

References drs_dims::Nt.

7.47.1.2 subroutine YokoiPlots::computeAndSaveAverage (integer,intent(out) *nstates*)

References drs_field::calc_field(), drs_flow::calc_flow(), drs_field::calc_rot_field(), drs_flow::calc_rot_flow(), drs_io::deflate, drs_mpi::drs_abort(), drs_field::drs_field_init(), drs_flow::drs_flow_init(), drs_io::drs_load_state(), drs_field::field_pol, drs_field::field_pol_avg, drs_field::field_pol_ddr, drs_field::field_pol_dr, drs_field::field_tor, drs_field::field_tor_avg, drs_field::field_tor_ddr, drs_field::field_tor_dr, drs_flow::flow_pol, drs_flow::flow_pol_avg, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, drs_flow::flow_tor, drs_flow::flow_tor_avg, drs_flow::flow_tor_ddr, drs_flow::flow_tor_dr, drs_io::inflate, drs_io::io_calc_file_in, drs_io::io_calc_file_out, drs_dims::Np_s, and drs_io::save_state().

Here is the call graph for this function:



7.47.1.3 subroutine YokoiPlots::computeBeta (double precision,dimension(0:(blk_t_size(mpi_rank),intent(inout) beta)

References drs_dims::Nt.

7.47.1.4 subroutine YokoiPlots::computeEMF (double precision,dimension(0:(blk_t_size(mpi_rank),intent(inout) EMF)

References drs_legendre::costheta, and drs_dims::Nt.

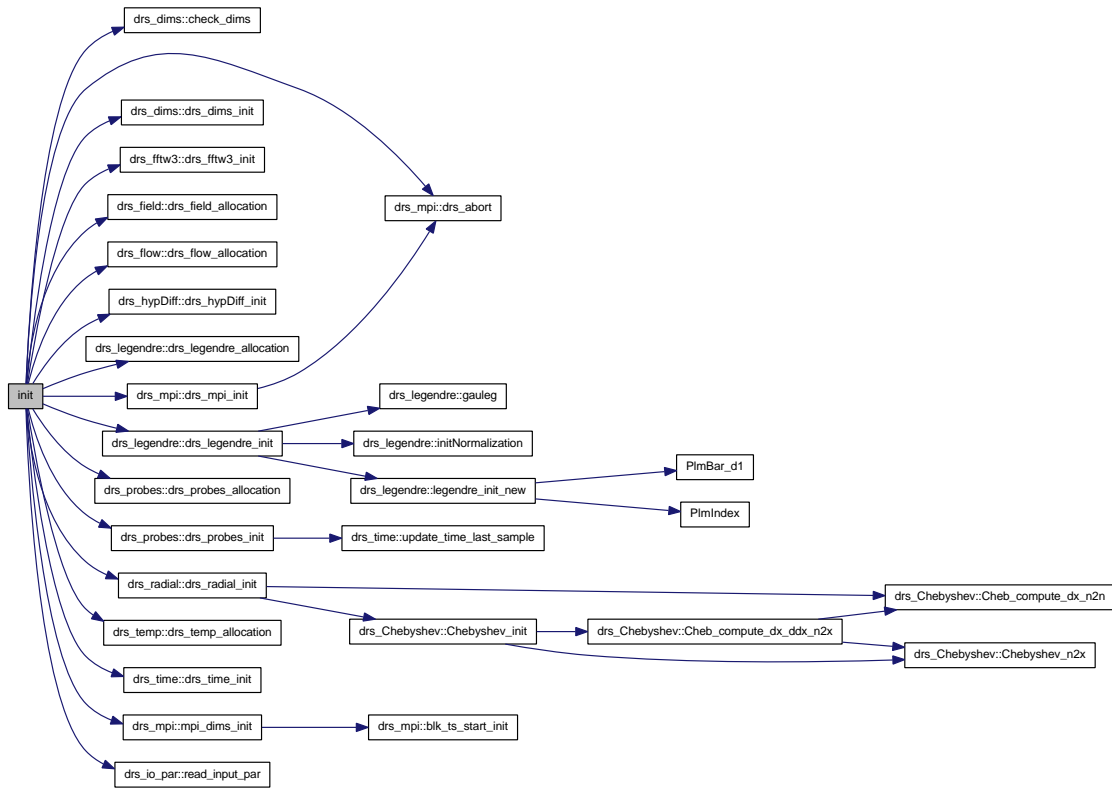
7.47.1.5 subroutine YokoiPlots::computeGamma (double precision,dimension(0:(blk_t_size(mpi_rank),intent(inout) gamma)

References drs_dims::Nt.

7.47.1.6 subroutine YokoiPlots::init (integer,intent(inout) error)

References drs_mpi::blk_t_size, drs_dims::check_dims(), drs_io_par::commenti, drs_time::drift, drs_io_par::drifti, drs_mpi::drs_abort(), drs_io_par::drs_calc_typei, drs_dims::drs_dims_init(), drs_fft3::drs_fft3_init(), drs_field::drs_field_allocation(), drs_flow::drs_flow_allocation(), drs_hypDiff::drs_hypDiff_init(), drs_legendre::drs_legendre_allocation(), drs_legendre::drs_legendre_init(), drs_mpi::drs_mpi_init(), drs_probes::drs_probes_allocation(), drs_probes::drs_probes_init(), drs_radial::drs_radial_init(), drs_temp::drs_temp_allocation(), drs_time::drs_time_init(), drs_hypDiff::drs_want_hypDiff, drs_io_par::etai, drs_io::io_calc_file_in, drs_io_par::lformi, drs_dims::m0, drs_io_par::m0i, drs_mpi::mpi_dims_init(), drs_mpi::mpi_rank, drs_mpi::mpi_size, drs_dims::Np, drs_dims::Np_s, drs_io_par::Npi, drs_io_par::Npi_s, drs_dims::Nr, drs_dims::Nr_s, drs_io_par::Nri, drs_io_par::Nri_s, drs_dims::Nt, drs_dims::Nt_s, drs_io_par::Nti, drs_io_par::Nti_s, drs_io_par::Pmi, drs_io_par::Pti, drs_io_par::Ra_ti, drs_io_par::read_input_par(), drs_io_par::Tai, and drs_time::time.

Here is the call graph for this function:



7.47.1.7 subroutine YokoiPlots::saveDXmeridional (double precision,dimension(0:(blk_t_-size(mpi_rank)),intent(in) *field*, double precision,intent(in) *phi*, character(len=*),intent(in) *filename*)

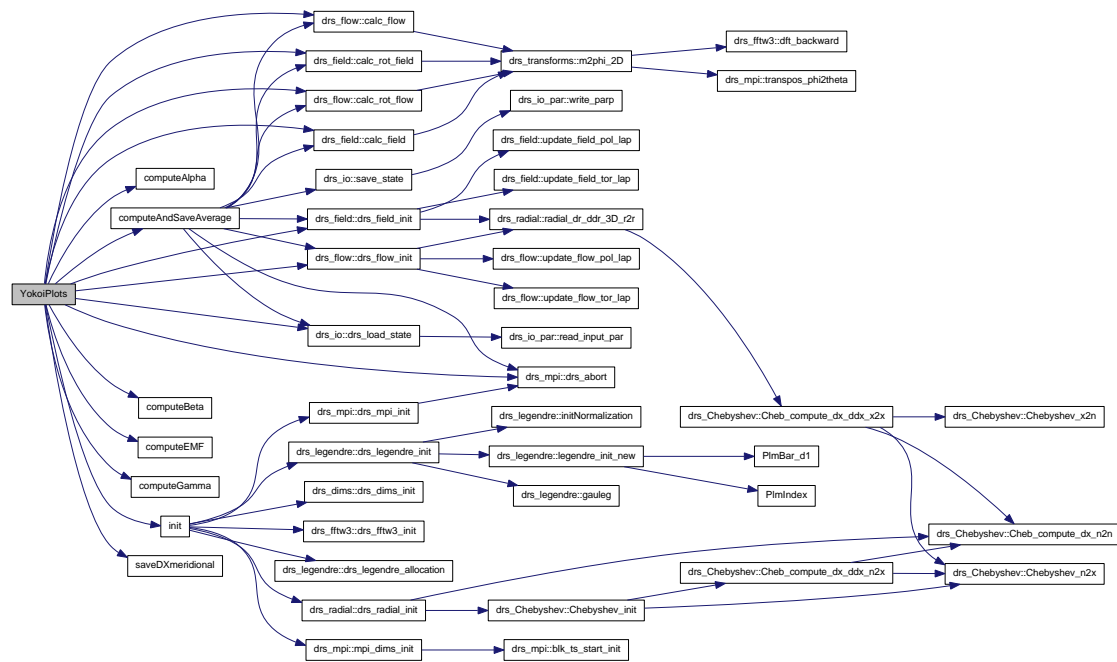
References `drs_legendre::costheta`, `drs_probes::dOmega`, `drs_dims::m0`, `drs_dims::Nt`, `drs_legendre::pi`, and `drs_radial::rcoll`.

7.47.1.8 program YokoiPlots ()

Computes the plots require for the Yokoi paper.

References `drs_field::calc_field()`, `drs_flow::calc_flow()`, `drs_field::calc_rot_field()`, `drs_flow::calc_rot_flow()`, `computeAlpha()`, `computeAndSaveAverage()`, `computeBeta()`, `computeEMF()`, `computeGamma()`, `drs_legendre::costheta`, `drs_io::deflate`, `drs_mpi::drs_abort()`, `drs_field::drs_field_init()`, `drs_flow::drs_flow_init()`, `drs_io::drs_load_state()`, `drs_field::field_pol`, `drs_field::field_pol_avg`, `drs_field::field_pol_ddr`, `drs_field::field_pol_dr`, `drs_field::field_tor`, `drs_field::field_tor_avg`, `drs_field::field_tor_ddr`, `drs_field::field_tor_dr`, `drs_flow::flow_pol`, `drs_flow::flow_pol_avg`, `drs_flow::flow_pol_ddr`, `drs_flow::flow_pol_dr`, `drs_flow::flow_tor`, `drs_flow::flow_tor_avg`, `drs_flow::flow_tor_ddr`, `drs_flow::flow_tor_dr`, `drs_io::inflate`, `init()`, `drs_io::io_calc_file_in`, `drs_dims::Np`, `drs_dims::Nr`, `drs_dims::Nt`, and `saveDXmeridional()`.

Here is the call graph for this function:



Index

- add_lock
 - drs_lock, 65
- adv_avg
 - drs_probes, 89
- apply_comp_BC
 - drs_comp, 18
- apply_field_pol_BC
 - drs_field, 30
- apply_field_tor_BC
 - drs_field, 30
- apply_flow_pol_BC
 - drs_flow, 37
- apply_flow_tor_BC
 - drs_flow, 37
- apply_temp_BC_RHS
 - drs_temp, 109
- applyGreen
 - drs.f90, 133
- average_unnormalised_field_1_spectrum
 - drs_probes, 83
- average_unnormalised_flow_1_spectrum
 - drs_probes, 83
- average_unnormalised_scalar_1_spectrum
 - drs_probes, 83
- b
 - drs_radial, 96
- Bdrkea
 - drs_probes, 89
- Bdrkes
 - drs_probes, 90
- Benchmark-gen.f90
 - Benchmarkv1, 192
 - init, 192
- Benchmark-v2.f90
 - Benchmarkv2, 194
 - cacheTemperatureProfile, 194
 - init, 195
 - selectEquatorMidShell, 196
 - volume, 196
- Benchmark.f90
 - Benchmarkv1, 197
 - cacheTemperatureProfile, 197
 - init, 198
 - selectEquatorMidShell, 199
 - volume, 199
- Benchmarkv1
 - Benchmark-gen.f90, 192
 - Benchmark.f90, 197
- Benchmarkv2
 - Benchmark-v2.f90, 194
- Betora
 - drs_probes, 90
- Betors
 - drs_probes, 90
- blk_ps_max_size
 - drs_mpi, 72
- blk_ps_size
 - drs_mpi, 72
- blk_ps_start
 - drs_mpi, 72
- blk_t_max_size
 - drs_mpi, 73
- blk_t_size
 - drs_mpi, 73
- blk_t_start
 - drs_mpi, 73
- blk_ts_start
 - drs_mpi, 73
- blk_ts_start_init
 - drs_mpi, 69
- Bmckea
 - drs_probes, 90
- Bmckes
 - drs_probes, 90
- Bnkea
 - drs_probes, 90
- Bnkes
 - drs_probes, 90
- Brspec_avg
 - drs_probes, 90
- c_integrate_r
 - drs_probes, 83
- cacheTemperatureProfile
 - Benchmark-v2.f90, 194
 - Benchmark.f90, 197
- calc_B
 - drs_field, 31
- calc_comp

- drs_comp, 18
- calc_field
 - drs_field, 31
- calc_field_lspec
 - drs_field, 31
- calc_field_mspec
 - drs_field, 32
- calc_field_nspec
 - drs_field, 32
- calc_flow
 - drs_flow, 37
- calc_flow_lspec
 - drs_flow, 37
- calc_flow_mspec
 - drs_flow, 38
- calc_flow_nspec
 - drs_flow, 38
- calc_rot_field
 - drs_field, 32
- calc_rot_flow
 - drs_flow, 38
- calc_temp
 - drs_temp, 109
- calc_u
 - drs_flow, 38
- cfl
 - drs_nonlinear, 78
- Cheb_compute_dx_ddx_n2x
 - drs_Chebyshev, 13
- Cheb_compute_dx_ddx_x2x
 - drs_Chebyshev, 13
- Cheb_compute_dx_n2n
 - drs_Chebyshev, 14
- Cheb_x
 - drs_Chebyshev, 15
- Chebyshev
 - drs_Chebyshev, 15
- Chebyshev_cleanup
 - drs_Chebyshev, 14
- Chebyshev_ddx
 - drs_Chebyshev, 15
- Chebyshev_dx
 - drs_Chebyshev, 15
- Chebyshev_init
 - drs_Chebyshev, 14
- Chebyshev_n2x
 - drs_Chebyshev, 14
- Chebyshev_x2n
 - drs_Chebyshev, 15
- check_dims
 - drs_dims, 22
- check_resolution_Hartman
 - drs_probes, 83
- commenti
 - drs_io_par, 54
- comp
 - drs_comp, 19
- comp_avg
 - drs_comp, 19
- comp_ddr
 - drs_comp, 19
- comp_dr
 - drs_comp, 19
- comp_dr_avg
 - drs_comp, 19
- comp_profile
 - drs_comp, 19
- comp_profile_dr
 - drs_comp, 20
- compProfName
 - drs_comp, 18
- compute_advection
 - drs_probes, 84
- compute_helicities
 - drs_probes, 84
- computeAlpha
 - Yokoi-plots-rs.f90, 216
 - Yokoi-plots.f90, 220
- computeAndSaveAverage
 - Yokoi-plots-rs.f90, 216
 - Yokoi-plots.f90, 220
- computeBeta
 - Yokoi-plots-rs.f90, 217
 - Yokoi-plots.f90, 221
- computeEMF
 - Yokoi-plots-rs.f90, 217
 - Yokoi-plots.f90, 221
- computeGamma
 - Yokoi-plots-rs.f90, 217
 - Yokoi-plots.f90, 221
- cos_r2r_1_n2r
 - drs_fftw3, 27
- cos_r2r_1_r2n
 - drs_fftw3, 27
- costheta
 - drs_legendre, 63
- cpu_max_time
 - drs_time, 114
- cpu_time_first_step
 - drs_time, 114
- cpu_time_now
 - drs_time, 114
- cpu_time_start
 - drs_time, 114
- CrankNicholson, 9
 - CrankNicholson_init, 10
 - field_lap_inv_pol, 10
 - field_lap_inv_tor, 10

- flow_lap_inv_pol, 10
 - flow_lap_inv_tor, 11
 - pinv, 11
 - temp_lap_inv, 11
 - updateCrankNicholson_matrices, 10
- CrankNicholson.f90, 131
- CrankNicholson_init
 - CrankNicholson, 10
- ct_buffer
 - drs_Chebyshev, 16
- cut_phi
 - drs_io_DX, 51
- cut_type
 - drs_io_DX, 51
- cut_z
 - drs_io_DX, 51
- deflate
 - drs_io, 47
- dft_backward
 - drs_fftw3, 27
- dft_forward
 - drs_fftw3, 27
- distribute_in_m
 - drs_mpi, 69
- dleg
 - drs_legendre, 63
- dOmega
 - drs_probes, 90
- drcoll
 - drs_radial, 96
- drift
 - drs_time, 114
- drifti
 - drs_io_par, 54
- drkea
 - drs_probes, 90
- drkes
 - drs_probes, 90
- drs
 - drs.f90, 134
- drs-spectra.f90
 - init, 200
 - StateAverage, 201
- drs-version.f90
 - drsVersion, 203
- drs.f90, 133
 - applyGreen, 133
 - drs, 134
 - drs_init, 136
 - kd_grothrate, 137
 - mk_green, 138
 - need_to_step, 138
 - update_field, 138
 - update_flow, 139
 - update_Green_functions, 140
 - update_temp, 140
- drs2dx
 - drs2dx.f90, 204
- drs2dx.f90
 - drs2dx, 204
 - init, 205
 - parse_drs2dx, 206
 - redefine_radial_coordinate, 206
- drs_abort
 - drs_mpi, 69
- drs_bcast_dble
 - drs_mpi, 69
 - drs_mpi::drs_bcast, 125
- drs_bcast_dble_scal
 - drs_mpi, 69
 - drs_mpi::drs_bcast, 125
- drs_bcast_int
 - drs_mpi, 70
 - drs_mpi::drs_bcast, 125
- drs_bcast_int_scal
 - drs_mpi, 70
 - drs_mpi::drs_bcast, 125
- drs_bcast_logical_scal
 - drs_mpi, 70
 - drs_mpi::drs_bcast, 125
- drs_calc_typei
 - drs_io_par, 54
- drs_Chebyshev, 12
 - Cheb_compute_dx_ddx_n2x, 13
 - Cheb_compute_dx_ddx_x2x, 13
 - Cheb_compute_dx_n2n, 14
 - Cheb_x, 15
 - Chebyshev, 15
 - Chebyshev_cleanup, 14
 - Chebyshev_ddx, 15
 - Chebyshev_dx, 15
 - Chebyshev_init, 14
 - Chebyshev_n2x, 14
 - Chebyshev_x2n, 15
 - ct_buffer, 16
 - Nx, 16
 - Nx_s, 16
 - pi, 16
 - plan_x, 16
- drs_Chebyshev.f90, 142
- drs_comp, 17
 - apply_comp_BC, 18
 - calc_comp, 18
 - comp, 19
 - comp_avg, 19
 - comp_ddr, 19
 - comp_dr, 19

- comp_dr_avg, 19
- comp_profile, 19
- comp_profile_dr, 20
- compProfName, 18
- drs_comp_allocation, 18
- drs_comp_init, 18
- drs_comp_randomize, 19
- drs_comp_reset, 19
- drs_comp.f90, 143
- drs_comp_allocation
 - drs_comp, 18
- drs_comp_init
 - drs_comp, 18
- drs_comp_randomize
 - drs_comp, 19
- drs_comp_reset
 - drs_comp, 19
- drs_debug, 21
 - save_lmr_quantity, 21
 - save_tpr_quantity, 21
- drs_debug.f90, 145
- drs_dims, 22
 - check_dims, 22
 - drs_dims_init, 23
 - lsymm, 23
 - m0, 23
 - Np, 23
 - Np_s, 23
 - Nr, 24
 - Nr_s, 24
 - Nt, 24
 - Nt_s, 24
 - usr_dims, 25
- drs_dims.f90, 146
- drs_dims_init
 - drs_dims, 23
- drs_fftw3, 26
 - cos_r2r_1_n2r, 27
 - cos_r2r_1_r2n, 27
 - dft_backward, 27
 - dft_forward, 27
 - drs_fftw3_cleanup, 27
 - drs_fftw3_init, 27
 - drs_fftw3_Np, 28
 - drs_fftw3_Nr, 28
 - drs_fftw3_Nt, 28
 - in_p, 28
 - inout_r, 28
 - plan_pb, 28
 - plan_pf, 28
 - plan_r, 29
 - remesh, 28
- drs_fftw3.f90, 148
- drs_fftw3_cleanup
 - drs_fftw3, 27
- drs_fftw3_init
 - drs_fftw3, 27
- drs_fftw3_Np
 - drs_fftw3, 28
- drs_fftw3_Nr
 - drs_fftw3, 28
- drs_fftw3_Nt
 - drs_fftw3, 28
- drs_field, 30
 - apply_field_pol_BC, 30
 - apply_field_tor_BC, 30
 - calc_B, 31
 - calc_field, 31
 - calc_field_lspec, 31
 - calc_field_mspec, 32
 - calc_field_nspec, 32
 - calc_rot_field, 32
 - drs_field_allocation, 32
 - drs_field_init, 32
 - drs_field_random_init, 33
 - field_pol, 33
 - field_pol_avg, 33
 - field_pol_ddr, 34
 - field_pol_dr, 34
 - field_pol_lap, 34
 - field_tor, 34
 - field_tor_avg, 34
 - field_tor_ddr, 34
 - field_tor_dr, 34
 - field_tor_lap, 34
 - update_field_pol_lap, 33
 - update_field_tor_lap, 33
- drs_field.f90, 149
- drs_field_allocation
 - drs_field, 32
- drs_field_init
 - drs_field, 32
- drs_field_random_init
 - drs_field, 33
- drs_flow, 36
 - apply_flow_pol_BC, 37
 - apply_flow_tor_BC, 37
 - calc_flow, 37
 - calc_flow_lspec, 37
 - calc_flow_mspec, 38
 - calc_flow_nspec, 38
 - calc_rot_flow, 38
 - calc_u, 38
 - drs_flow_allocation, 39
 - drs_flow_init, 39
 - flow_pol, 40
 - flow_pol_avg, 40
 - flow_pol_ddr, 40

- flow_pol_dr, 40
- flow_pol_lap, 40
- flow_tor, 40
- flow_tor_avg, 41
- flow_tor_ddr, 41
- flow_tor_dr, 41
- flow_tor_lap, 41
- update_flow_pol_lap, 39
- update_flow_tor_lap, 40
- drs_flow.f90, 151
- drs_flow_allocation
 - drs_flow, 39
- drs_flow_init
 - drs_flow, 39
- drs_gather_vars
 - drs_mpi, 70
- drs_hypDiff, 42
 - drs_hypDiff_init, 42
 - drs_want_hypDiff, 42
 - hypDiff, 42
- drs_hypDiff.f90, 153
- drs_hypDiff_init
 - drs_hypDiff, 42
- drs_init
 - drs.f90, 136
- drs_io, 43
 - deflate, 47
 - drs_load_state, 43
 - drs_open_output, 44
 - dump_state, 44
 - inflate, 47
 - io_calc_file_in, 47
 - io_calc_file_out, 47
 - save_l_spec, 45
 - save_m_spec, 45
 - save_n_spec, 46
 - save_state, 46
- drs_io.f90, 154
- drs_io_DX, 48
 - cut_phi, 51
 - cut_type, 51
 - cut_z, 51
 - save2DXscalar, 48
 - save2DXvector, 49
 - saveDXmeridional, 49
 - saveDXmeridional3DVec, 49
 - saveDXvolume, 50
 - saveDXvolume3DVec, 50
 - saveDXvolume_v2, 50
 - where_to_cut, 51
- drs_io_DX::save2DX, 129
 - save2DXscalar, 129
 - save2DXvector, 129
- drs_io_par, 52
 - commenti, 54
 - drifti, 54
 - drs_calc_typei, 54
 - drs_read_conf, 53
 - drs_read_conf_v2, 53
 - etai, 54
 - flowBCi, 54
 - hi, 54
 - lformi, 54
 - lsymmi, 54
 - m0i, 54
 - magBCi, 54
 - magic, 55
 - MAGICC1, 55
 - MAGICC2, 55
 - MAGICC3, 55
 - MAGICC4, 55
 - MAGICC5, 55
 - MAGICC6, 55
 - MAGICC7, 55
 - MAGICC9, 55
 - magici, 55
 - noise, 55
 - Npi, 55
 - Npi_s, 55
 - Nri, 55
 - Nri_s, 56
 - Nti, 56
 - Nti_s, 56
 - Pmi, 56
 - Pti, 56
 - Ra_ti, 56
 - read_input_par, 53
 - sampling_ratei, 56
 - stepmaxi, 56
 - Tai, 56
 - tempBCi, 56
 - transienti, 56
 - usr_dimsi, 57
 - write_parp, 53
- drs_io_par.f90, 155
- drs_io_units, 58
 - unit_adv, 58
 - unit_am, 58
 - unit_cfl, 58
 - unit_dissB, 58
 - unit_dissu, 58
 - unit_eb, 59
 - unit_ek, 59
 - unit_evp, 59
 - unit_evt, 59
 - unit_koeb, 59
 - unit_koeu, 59
 - unit_lspec, 59

- unit_mspec, 59
- unit_nspec, 59
- unit_nu, 59
- unit_t, 59
- unit_u_mid, 60
- unit_uaz, 60
- unit_ur, 60
- unit_uzon, 60
- drs_io_units.f90, 157
- drs_legendre, 61
 - costheta, 63
 - dleg, 63
 - drs_legendre_allocation, 62
 - drs_legendre_init, 62
 - gauleg, 62
 - initNormalization, 62
 - leg_neg, 63
 - leg_sin, 63
 - legendre, 64
 - legendre_init_new, 63
 - llp1, 64
 - pi, 64
 - plmfac, 64
 - sintheta, 64
 - w, 64
- drs_legendre.f90, 158
- drs_legendre::interface, 128
 - PLegendreA_d1, 128
 - PlmBar_d1, 128
 - PlmIndex, 128
- drs_legendre_allocation
 - drs_legendre, 62
- drs_legendre_init
 - drs_legendre, 62
- drs_load_state
 - drs_io, 43
- drs_lock, 65
 - add_lock, 65
 - drs_lock_init, 65
 - lockExists, 66
 - lockFileName, 66
 - lockFileUnit, 66
 - rm_lock, 66
- drs_lock.f90, 160
- drs_lock_init
 - drs_lock, 65
- drs_maximize_dble
 - drs_mpi, 70
 - drs_mpi::drs_maximize, 126
- drs_maximize_dble_scal
 - drs_mpi, 70
 - drs_mpi::drs_maximize, 126
- drs_minimize_dble
 - drs_mpi, 70
 - drs_mpi::drs_minimize, 127
- drs_mpi::drs_minimize, 127
- drs_minimize_dble_scal
 - drs_mpi, 70
 - drs_mpi::drs_minimize, 127
- drs_mpi, 67
 - blk_ps_max_size, 72
 - blk_ps_size, 72
 - blk_ps_start, 72
 - blk_t_max_size, 73
 - blk_t_size, 73
 - blk_t_start, 73
 - blk_ts_start, 73
 - blk_ts_start_init, 69
 - distribute_in_m, 69
 - drs_abort, 69
 - drs_bcast_dble, 69
 - drs_bcast_dble_scal, 69
 - drs_bcast_int, 70
 - drs_bcast_int_scal, 70
 - drs_bcast_logical_scal, 70
 - drs_gather_vars, 70
 - drs_maximize_dble, 70
 - drs_maximize_dble_scal, 70
 - drs_minimize_dble, 70
 - drs_minimize_dble_scal, 70
 - drs_mpi_init, 70
 - gather_from_m, 71
 - mm, 73
 - mpi_cleanup, 71
 - mpi_dims_init, 71
 - mpi_rank, 74
 - mpi_size, 74
 - sum_over_all_cpus_scal, 71
 - sum_over_all_cpus_vect, 71
 - transpos_phi2theta, 71
 - transpos_theta2phi, 72
 - wait_for_everyone, 72
- drs_mpi.f90, 161
- drs_mpi::drs_bcast, 125
 - drs_bcast_dble, 125
 - drs_bcast_dble_scal, 125
 - drs_bcast_int, 125
 - drs_bcast_int_scal, 125
 - drs_bcast_logical_scal, 125
- drs_mpi::drs_maximize, 126
 - drs_maximize_dble, 126
 - drs_maximize_dble_scal, 126
- drs_mpi::drs_minimize, 127
 - drs_minimize_dble, 127
 - drs_minimize_dble_scal, 127
- drs_mpi::sum_over_all_cpus, 130
 - sum_over_all_cpus_scal, 130
 - sum_over_all_cpus_vect, 130
- drs_mpi_init

- drs_mpi, 70
- drs_nonlinear, 75
 - cfl, 78
 - drs_nonlinear_init, 76
 - evaluate_real_space, 76
 - field_p_t, 78
 - field_r_t, 78
 - field_t_t, 78
 - flow_p_t, 78
 - flow_r_t, 78
 - flow_t_t, 79
 - ncfl, 79
 - rhs, 76
 - rhs_IE_pol, 79
 - rhs_IE_tor, 79
 - rhs_NS_pol, 79
 - rhs_NS_tor, 79
 - rhs_TE, 79
 - rot_field_p_t, 79
 - rot_field_r_t, 79
 - rot_field_t_t, 79
 - rot_flow_p_t, 79
 - rot_flow_r_t, 80
 - rot_flow_t_t, 80
 - save_stuff, 77
 - temp_t, 80
- drs_nonlinear.f90, 164
- drs_nonlinear_init
 - drs_nonlinear, 76
- drs_open_output
 - drs_io, 44
- drs_params.f90, 166
- drs_probes, 81
 - adv_avg, 89
 - average_unnormalised_field_l_spectrum, 83
 - average_unnormalised_flow_l_spectrum, 83
 - average_unnormalised_scalar_l_spectrum, 83
 - Bdrkea, 89
 - Bdrkes, 90
 - Betora, 90
 - Betors, 90
 - Bmckea, 90
 - Bmckes, 90
 - Bnkea, 90
 - Bnkes, 90
 - Brspec_avg, 90
 - c_integrate_r, 83
 - check_resolution_Hartman, 83
 - compute_advection, 84
 - compute_helicities, 84
 - dOmega, 90
 - drkea, 90
 - drkes, 90
 - drs_probes_allocation, 84
 - drs_probes_init, 84
 - EB, 91
 - Ekin, 91
 - energy, 85
 - etora, 91
 - etors, 91
 - groth, 91
 - integrate_power_surf, 85
 - integrate_r, 85
 - l_spec_of_scalar_field, 85
 - m_spec_of_scalar_field, 86
 - mckea, 91
 - mckes, 91
 - measure, 86
 - measure_lm, 86
 - n_spec_of_scalar_field, 87
 - nkea, 91
 - nkes, 91
 - nusselt, 87
 - Rm, 91
 - save_angular_momentum, 88
 - save_field_coeffs, 88
 - save_flow_coeffs, 88
 - save_flow_dissipation, 89
 - save_magnetic_dissipation, 89
 - t2_avg, 91
 - tspec_avg, 92
 - up2, 92
 - up_avg, 92
 - ur_avg, 92
 - urspec_avg, 92
 - ut2, 92
 - ut_avg, 92
- drs_probes.f90, 167
- drs_probes_allocation
 - drs_probes, 84
- drs_probes_init
 - drs_probes, 84
- drs_radial, 93
 - b, 96
 - drcoll, 96
 - drs_radial_init, 94
 - poly, 97
 - poly_ddr, 97
 - poly_dr, 97
 - radial_derivative_r2r, 94
 - radial_dr_ddr_1D_n2r, 94
 - radial_dr_ddr_1D_r2r, 95
 - radial_dr_ddr_3D_n2r, 95
 - radial_dr_ddr_3D_r2r, 96
 - rcoll, 97
 - rcoll2, 97
- drs_radial.f90, 170
- drs_radial_init

- drs_radial, 94
- drs_read_conf
 - drs_io_par, 53
- drs_read_conf_v2
 - drs_io_par, 53
- drs_renderers, 98
 - drs_renderers_allocation, 99
 - render, 99
 - render_B, 100
 - render_B_outside, 100
 - render_Bp, 101
 - render_Br, 101
 - render_Bt, 101
 - render_Bz, 102
 - render_helicity, 102
 - render_out, 107
 - render_poloidal_streamlines, 102
 - render_radial_streamfunction, 103
 - render_rotu, 103
 - render_rotu_p, 103
 - render_rotu_r, 103
 - render_rotu_t, 104
 - render_rotu_z, 104
 - render_streamlines_t, 104
 - render_temperature, 105
 - render_temperature_perturbation, 105
 - render_temprature_grad_r, 105
 - render_u, 105
 - render_up, 106
 - render_ur, 106
 - render_ut, 106
 - render_uz, 106
 - XX, 107
 - YY, 107
 - ZZ, 107
- drs_renderers.f90
 - ERR_UNSUPPORTED_OPTION, 210
- drs_renderers_allocation
 - drs_renderers, 99
- drs_temp, 108
 - apply_temp_BC_RHS, 109
 - calc_temp, 109
 - drs_temp_allocation, 109
 - drs_temp_init, 109
 - drs_temp_randomize, 110
 - drs_temp_reset, 110
 - temp, 110
 - temp_avg, 110
 - temp_ddd, 111
 - temp_dr, 111
 - temp_dr_avg, 111
 - temp_lap, 111
 - temp_profile, 111
 - temp_profile_dr, 111
 - tempProfName, 110
 - update_temp_lap, 110
- drs_temp.f90, 172
- drs_temp_allocation
 - drs_temp, 109
- drs_temp_init
 - drs_temp, 109
- drs_temp_randomize
 - drs_temp, 110
- drs_temp_reset
 - drs_temp, 110
- drs_time, 112
 - cpu_max_time, 114
 - cpu_time_first_step, 114
 - cpu_time_now, 114
 - cpu_time_start, 114
 - drift, 114
 - drs_time_init, 113
 - drs_time_update, 113
 - d timestep, 114
 - h, 115
 - h_old, 115
 - imeasure, 115
 - max_time, 115
 - nsample, 115
 - sampling_rate, 115
 - stepmax, 115
 - steps, 115
 - stepstart, 116
 - time, 116
 - time_last_sample, 116
 - time_since_last_sample, 116
 - time_start, 116
 - transient, 116
 - update_time_last_sample, 113
 - update_timestep, 114
 - variable_h, 116
- drs_time.f90, 174
- drs_time_init
 - drs_time, 113
- drs_time_update
 - drs_time, 113
- drs_transforms, 118
 - m2phi_2D, 119
 - my_div, 119
 - my_rot, 119
 - my_rotrot, 120
 - pi, 122
 - PolTor_common2PolTor_field, 120
 - PolTor_common2PolTor_flow, 120
 - vectorField2Divergence, 120
 - vectorField2PolTor_common, 121
 - ylmb, 121
 - ylmt, 121

- ylmt_3D, 122
- drs_transforms.f90, 176
- drs_want_hypDiff
 - drs_hypDiff, 42
- drsVersion
 - drs-version.f90, 203
- d timestep
 - drs_time, 114
- dump_state
 - drs_io, 44
- EB
 - drs_probes, 91
- Ekin
 - drs_probes, 91
- energy
 - drs_probes, 85
- ERR_UNSUPPORTED_OPTION
 - drs_renderers.f90, 210
- etai
 - drs_io_par, 54
- etora
 - drs_probes, 91
- etors
 - drs_probes, 91
- evaluate_real_space
 - drs_nonlinear, 76
- field_lap_inv_pol
 - CrankNicholson, 10
- field_lap_inv_tor
 - CrankNicholson, 10
- field_p_t
 - drs_nonlinear, 78
- field_pol
 - drs_field, 33
- field_pol_avg
 - drs_field, 33
- field_pol_ddr
 - drs_field, 34
- field_pol_dr
 - drs_field, 34
- field_pol_lap
 - drs_field, 34
- field_r_t
 - drs_nonlinear, 78
- field_t_t
 - drs_nonlinear, 78
- field_tor
 - drs_field, 34
- field_tor_avg
 - drs_field, 34
- field_tor_ddr
 - drs_field, 34
- field_tor_dr
 - drs_field, 34
- field_tor_lap
 - drs_field, 34
- flow_lap_inv_pol
 - CrankNicholson, 10
- flow_lap_inv_tor
 - CrankNicholson, 11
- flow_p_t
 - drs_nonlinear, 78
- flow_pol
 - drs_flow, 40
- flow_pol_avg
 - drs_flow, 40
- flow_pol_ddr
 - drs_flow, 40
- flow_pol_dr
 - drs_flow, 40
- flow_pol_lap
 - drs_flow, 40
- flow_r_t
 - drs_nonlinear, 78
- flow_t_t
 - drs_nonlinear, 79
- flow_tor
 - drs_flow, 40
- flow_tor_avg
 - drs_flow, 41
- flow_tor_ddr
 - drs_flow, 41
- flow_tor_dr
 - drs_flow, 41
- flow_tor_lap
 - drs_flow, 41
- flowBCi
 - drs_io_par, 54
- gather_from_m
 - drs_mpi, 71
- gauleg
 - drs_legendre, 62
- getProfile
 - getProfile.f90, 211
- getProfile.f90
 - getProfile, 211
 - init, 212
 - parseConfig, 213
 - setWhatName, 213
- groth
 - drs_probes, 91
- h
 - drs_time, 115
- h_old

- drs_time, 115
- hi
 - drs_io_par, 54
- hypDiff
 - drs_hypDiff, 42
- imeasure
 - drs_time, 115
- in_p
 - drs_fftw3, 28
- inflate
 - drs_io, 47
- init
 - Benchmark-gen.f90, 192
 - Benchmark-v2.f90, 195
 - Benchmark.f90, 198
 - drs-spectra.f90, 200
 - drs2dx.f90, 205
 - getProfile.f90, 212
 - state-average.f90, 214
 - test_vectorField2Divergence.f90, 190
 - Yokoi-plots-rs.f90, 217
 - Yokoi-plots.f90, 221
- initNormalization
 - drs_legendre, 62
- inout_r
 - drs_fftw3, 28
- integrate_power_surf
 - drs_probes, 85
- integrate_r
 - drs_probes, 85
- io_calc_file_in
 - drs_io, 47
- io_calc_file_out
 - drs_io, 47
- kd_grothrate
 - drs.f90, 137
- l_spec_of_scalar_field
 - drs_probes, 85
- leg_neg
 - drs_legendre, 63
- leg_sin
 - drs_legendre, 63
- legendre
 - drs_legendre, 64
- legendre_init_new
 - drs_legendre, 63
- lformi
 - drs_io_par, 54
- llp1
 - drs_legendre, 64
- lockExists
 - drs_lock, 66
- lockFileName
 - drs_lock, 66
- lockFileUnit
 - drs_lock, 66
- lsymm
 - drs_dims, 23
- lsymmi
 - drs_io_par, 54
- m0
 - drs_dims, 23
- m0i
 - drs_io_par, 54
- m2phi_2D
 - drs_transforms, 119
- m_spec_of_scalar_field
 - drs_probes, 86
- magBCi
 - drs_io_par, 54
- magic
 - drs_io_par, 55
- MAGICC1
 - drs_io_par, 55
- MAGICC2
 - drs_io_par, 55
- MAGICC3
 - drs_io_par, 55
- MAGICC4
 - drs_io_par, 55
- MAGICC5
 - drs_io_par, 55
- MAGICC6
 - drs_io_par, 55
- MAGICC7
 - drs_io_par, 55
- MAGICC9
 - drs_io_par, 55
- magici
 - drs_io_par, 55
- max_time
 - drs_time, 115
- mckea
 - drs_probes, 91
- mckes
 - drs_probes, 91
- measure
 - drs_probes, 86
- measure_lm
 - drs_probes, 86
- mk_green
 - drs.f90, 138
- mm
 - drs_mpi, 73

- mpi_cleanup
 - drs_mpi, 71
- mpi_dims_init
 - drs_mpi, 71
- mpi_rank
 - drs_mpi, 74
- mpi_size
 - drs_mpi, 74
- my_div
 - drs_transforms, 119
- my_rot
 - drs_transforms, 119
- my_rotrot
 - drs_transforms, 120
- n_spec_of_scalar_field
 - drs_probes, 87
- ncfl
 - drs_nonlinear, 79
- need_to_step
 - drs.f90, 138
- nkea
 - drs_probes, 91
- nkes
 - drs_probes, 91
- noise
 - drs_io_par, 55
- Np
 - drs_dims, 23
- Np_s
 - drs_dims, 23
- Npi
 - drs_io_par, 55
- Npi_s
 - drs_io_par, 55
- Nr
 - drs_dims, 24
- Nr_s
 - drs_dims, 24
- Nri
 - drs_io_par, 55
- Nri_s
 - drs_io_par, 56
- nsample
 - drs_time, 115
- Nt
 - drs_dims, 24
- Nt_s
 - drs_dims, 24
- Nti
 - drs_io_par, 56
- Nti_s
 - drs_io_par, 56
- nusselt
 - drs_probes, 87
- Nx
 - drs_Chebyshev, 16
- Nx_s
 - drs_Chebyshev, 16
- parse
 - parser, 123
- parse_drs2dx
 - drs2dx.f90, 206
- parseConfig
 - getProfile.f90, 213
- parser, 123
 - parse, 123
- parser/parser.f90, 178
- pi
 - drs_Chebyshev, 16
 - drs_legendre, 64
 - drs_transforms, 122
- pinv
 - CrankNicholson, 11
- plan_pb
 - drs_fftw3, 28
- plan_pf
 - drs_fftw3, 28
- plan_r
 - drs_fftw3, 29
- plan_x
 - drs_Chebyshev, 16
- PLegendreA_d1
 - drs_legendre::interface, 128
 - PLegendreA_d1.f90, 179
- PLegendreA_d1.f90
 - PLegendreA_d1, 179
- PlmBar_d1
 - drs_legendre::interface, 128
 - PlmBar_d1.f90, 180
- PlmBar_d1.f90
 - PlmBar_d1, 180
- plmfac
 - drs_legendre, 64
- PlmIndex
 - drs_legendre::interface, 128
 - PlmIndex.f90, 181
- PlmIndex.f90
 - PlmIndex, 181
- Pmi
 - drs_io_par, 56
- PolTor_common2PolTor_field
 - drs_transforms, 120
- PolTor_common2PolTor_flow
 - drs_transforms, 120
- poly
 - drs_radial, 97

- poly_ddr
 - drs_radial, 97
- poly_dr
 - drs_radial, 97
- Pti
 - drs_io_par, 56
- Ra_ti
 - drs_io_par, 56
- radial_derivative_r2r
 - drs_radial, 94
- radial_dr_ddr_1D_n2r
 - drs_radial, 94
- radial_dr_ddr_1D_r2r
 - drs_radial, 95
- radial_dr_ddr_3D_n2r
 - drs_radial, 95
- radial_dr_ddr_3D_r2r
 - drs_radial, 96
- rcoll
 - drs_radial, 97
- rcoll2
 - drs_radial, 97
- read_input_par
 - drs_io_par, 53
- redefine_radial_coordinate
 - drs2dx.f90, 206
- remesh
 - drs_fftw3, 28
- render
 - drs_renderers, 99
- render_B
 - drs_renderers, 100
- render_B_outside
 - drs_renderers, 100
- render_Bp
 - drs_renderers, 101
- render_Br
 - drs_renderers, 101
- render_Bt
 - drs_renderers, 101
- render_Bz
 - drs_renderers, 102
- render_helicity
 - drs_renderers, 102
- render_out
 - drs_renderers, 107
- render_poloidal_streamlines
 - drs_renderers, 102
- render_radial_streamfunction
 - drs_renderers, 103
- render_rotu
 - drs_renderers, 103
- render_rotu_p
 - drs_renderers, 103
- render_rotu_r
 - drs_renderers, 103
- render_rotu_t
 - drs_renderers, 104
- render_rotu_z
 - drs_renderers, 104
- render_streamlines_t
 - drs_renderers, 104
- render_temperature
 - drs_renderers, 105
- render_temperature_perturbation
 - drs_renderers, 105
- render_temprature_grad_r
 - drs_renderers, 105
- render_u
 - drs_renderers, 105
- render_up
 - drs_renderers, 106
- render_ur
 - drs_renderers, 106
- render_ut
 - drs_renderers, 106
- render_uz
 - drs_renderers, 106
- rhs
 - drs_nonlinear, 76
- rhs_IE_pol
 - drs_nonlinear, 79
- rhs_IE_tor
 - drs_nonlinear, 79
- rhs_NS_pol
 - drs_nonlinear, 79
- rhs_NS_tor
 - drs_nonlinear, 79
- rhs_TE
 - drs_nonlinear, 79
- Rm
 - drs_probes, 91
- rm_lock
 - drs_lock, 66
- rot_field_p_t
 - drs_nonlinear, 79
- rot_field_r_t
 - drs_nonlinear, 79
- rot_field_t_t
 - drs_nonlinear, 79
- rot_flow_p_t
 - drs_nonlinear, 79
- rot_flow_r_t
 - drs_nonlinear, 80
- rot_flow_t_t
 - drs_nonlinear, 80

- sampling_rate
 - drs_time, 115
- sampling_ratei
 - drs_io_par, 56
- save2DXscalar
 - drs_io_DX, 48
 - drs_io_DX::save2DX, 129
- save2DXvector
 - drs_io_DX, 49
 - drs_io_DX::save2DX, 129
- save_angular_momentum
 - drs_probes, 88
- save_field_coeffs
 - drs_probes, 88
- save_flow_coeffs
 - drs_probes, 88
- save_flow_dissipation
 - drs_probes, 89
- save_l_spec
 - drs_io, 45
- save_lmr_quantity
 - drs_debug, 21
- save_m_spec
 - drs_io, 45
- save_magnetic_dissipation
 - drs_probes, 89
- save_n_spec
 - drs_io, 46
- save_state
 - drs_io, 46
- save_stuff
 - drs_nonlinear, 77
- save_tpr_quantity
 - drs_debug, 21
- saveDXmeridional
 - drs_io_DX, 49
 - test_saveDXmeridional.f90, 189
 - Yokoi-plots.f90, 222
- saveDXmeridional3DVec
 - drs_io_DX, 49
- saveDXvolume
 - drs_io_DX, 50
- saveDXvolume3DVec
 - drs_io_DX, 50
- saveDXvolume_v2
 - drs_io_DX, 50
- saveIDLmeridional
 - Yokoi-plots-rs.f90, 218
- selectEquatorMidShell
 - Benchmark-v2.f90, 196
 - Benchmark.f90, 199
- setWhatName
 - getProfile.f90, 213
- SHTOOLS/PLegendreA_d1.f90, 179
- SHTOOLS/PlmBar_d1.f90, 180
- SHTOOLS/PlmIndex.f90, 181
- sintheta
 - drs_legendre, 64
- state-average.f90
 - init, 214
 - StateAverage, 215
- StateAverage
 - drs-spectra.f90, 201
 - state-average.f90, 215
 - Yokoi-plots-rs.f90, 218
- stepmax
 - drs_time, 115
- stepmaxi
 - drs_io_par, 56
- steps
 - drs_time, 115
- stepstart
 - drs_time, 116
- sum_over_all_cpus_scal
 - drs_mpi, 71
 - drs_mpi::sum_over_all_cpus, 130
- sum_over_all_cpus_vect
 - drs_mpi, 71
 - drs_mpi::sum_over_all_cpus, 130
- t2_avg
 - drs_probes, 91
- Tai
 - drs_io_par, 56
- temp
 - drs_temp, 110
- temp_avg
 - drs_temp, 110
- temp_ddr
 - drs_temp, 111
- temp_dr
 - drs_temp, 111
- temp_dr_avg
 - drs_temp, 111
- temp_lap
 - drs_temp, 111
- temp_lap_inv
 - CrankNicholson, 11
- temp_profile
 - drs_temp, 111
- temp_profile_dr
 - drs_temp, 111
- temp_t
 - drs_nonlinear, 80
- tempBCi
 - drs_io_par, 56
- tempProfName
 - drs_temp, 110

- test_drs_Chebyshev.f90
 - test_drs_radial, 182
- test_drs_fftw
 - test_drs_fftw-r2r.f90, 183
 - test_drs_fftw.f90, 184
 - test_remesh-r2r.f90, 188
- test_drs_fftw-r2r.f90
 - test_drs_fftw, 183
- test_drs_fftw.f90
 - test_drs_fftw, 184
- test_drs_radial
 - test_drs_Chebyshev.f90, 182
 - test_drs_radial.f90, 185
- test_drs_radial.f90
 - test_drs_radial, 185
 - test_radial_colocation_points, 185
 - test_radial_derivative_r2r, 185
 - test_radial_dr_ddr_1D_r2r, 186
- test_logFeature.f90
 - testLogFeature, 187
- test_radial_colocation_points
 - test_drs_radial.f90, 185
- test_radial_derivative_r2r
 - test_drs_radial.f90, 185
- test_radial_dr_ddr_1D_r2r
 - test_drs_radial.f90, 186
- test_remesh-r2r.f90
 - test_drs_fftw, 188
- test_saveDXMer
 - test_saveDXmeridional.f90, 189
- test_saveDXmeridional.f90
 - saveDXmeridional, 189
 - test_saveDXMer, 189
- test_vectorField2Divergence
 - test_vectorField2Divergence.f90, 190
- test_vectorField2Divergence.f90
 - init, 190
 - test_vectorField2Divergence, 190
- testLogFeature
 - test_logFeature.f90, 187
- tests/test_drs_Chebyshev.f90, 182
- tests/test_drs_fftw-r2r.f90, 183
- tests/test_drs_fftw.f90, 184
- tests/test_drs_radial.f90, 185
- tests/test_logFeature.f90, 187
- tests/test_remesh-r2r.f90, 188
- tests/test_saveDXmeridional.f90, 189
- tests/test_vectorField2Divergence.f90, 190
- time
 - drs_time, 116
- time_last_sample
 - drs_time, 116
- time_since_last_sample
 - drs_time, 116
- time_start
 - drs_time, 116
- transient
 - drs_time, 116
- transienti
 - drs_io_par, 56
- transpos_phi2theta
 - drs_mpi, 71
- transpos_theta2phi
 - drs_mpi, 72
- tspec_avg
 - drs_probes, 92
- unit_adv
 - drs_io_units, 58
- unit_am
 - drs_io_units, 58
- unit_cfl
 - drs_io_units, 58
- unit_dissB
 - drs_io_units, 58
- unit_dissu
 - drs_io_units, 58
- unit_eb
 - drs_io_units, 59
- unit_ek
 - drs_io_units, 59
- unit_evp
 - drs_io_units, 59
- unit_evt
 - drs_io_units, 59
- unit_koeb
 - drs_io_units, 59
- unit_koeu
 - drs_io_units, 59
- unit_kspec
 - drs_io_units, 59
- unit_mspect
 - drs_io_units, 59
- unit_nspec
 - drs_io_units, 59
- unit_nu
 - drs_io_units, 59
- unit_t
 - drs_io_units, 59
- unit_u_mid
 - drs_io_units, 60
- unit_uaz
 - drs_io_units, 60
- unit_ur
 - drs_io_units, 60
- unit_uzon
 - drs_io_units, 60
- up2

- drs_probes, 92
- up_avg
 - drs_probes, 92
- update_field
 - drs.f90, 138
- update_field_pol_lap
 - drs_field, 33
- update_field_tor_lap
 - drs_field, 33
- update_flow
 - drs.f90, 139
- update_flow_pol_lap
 - drs_flow, 39
- update_flow_tor_lap
 - drs_flow, 40
- update_Green_functions
 - drs.f90, 140
- update_temp
 - drs.f90, 140
- update_temp_lap
 - drs_temp, 110
- update_time_last_sample
 - drs_time, 113
- update_timestep
 - drs_time, 114
- updateCrankNicholson_matrices
 - CrankNicholson, 10
- ur_avg
 - drs_probes, 92
- urspec_avg
 - drs_probes, 92
- usr_dims
 - drs_dims, 25
- usr_dimsi
 - drs_io_par, 57
- ut2
 - drs_probes, 92
- ut_avg
 - drs_probes, 92
- utilities/Benchmark-gen.f90, 192
- utilities/Benchmark-v2.f90, 194
- utilities/Benchmark.f90, 197
- utilities/drs-spectra.f90, 200
- utilities/drs-version.f90, 203
- utilities/drs2dx.f90, 204
- utilities/drs_io_DX.f90, 207
- utilities/drs_renderers.f90, 209
- utilities/getProfile.f90, 211
- utilities/state-average.f90, 214
- utilities/Yokoi-plots-rs.f90, 216
- utilities/Yokoi-plots.f90, 220
- variable_h
 - drs_time, 116
- vectorField2Divergence
 - drs_transforms, 120
- vectorField2PolTor_common
 - drs_transforms, 121
- volume
 - Benchmark-v2.f90, 196
 - Benchmark.f90, 199
- w
 - drs_legendre, 64
- wait_for_everyone
 - drs_mpi, 72
- where_to_cut
 - drs_io_DX, 51
- write_parp
 - drs_io_par, 53
- XX
 - drs_renderers, 107
- ylmb
 - drs_transforms, 121
- ylmt
 - drs_transforms, 121
- ylmt_3D
 - drs_transforms, 122
- Yokoi-plots-rs.f90
 - computeAlpha, 216
 - computeAndSaveAverage, 216
 - computeBeta, 217
 - computeEMF, 217
 - computeGamma, 217
 - init, 217
 - saveIDLmeridional, 218
 - StateAverage, 218
- Yokoi-plots.f90
 - computeAlpha, 220
 - computeAndSaveAverage, 220
 - computeBeta, 221
 - computeEMF, 221
 - computeGamma, 221
 - init, 221
 - saveDXmeridional, 222
 - YokoiPlots, 222
- YokoiPlots
 - Yokoi-plots.f90, 222
- YY
 - drs_renderers, 107
- ZZ
 - drs_renderers, 107