DRS

1.7.0

Generated by Doxygen 1.6.1

Sat Feb 28 16:13:49 2015

Contents

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**) Ito 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:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

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.

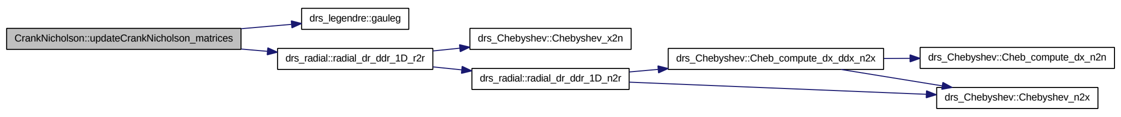
Pm the magnetic Prandtl number.

Pc the compositionsl 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:


CrankNicholson::updateCrankNicholson_matrices
drs_legendre::gauleg
drs_radial::radial_dr_ddr_1D_r2r
drs_Chebyshev::Chebyshev_x2n
drs_radial::radial_dr_ddr_1D_n2r
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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 Chebychev 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 Chebychev space, derivatives are returned in direct space.

Since:

1.6.5

Parameters:

f Chebyshev coefficients of the input function

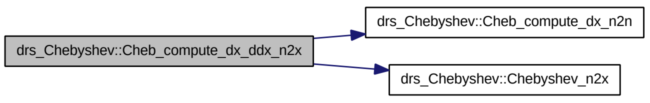
dfdx First derivative of f at points 1..Nx

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:


drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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

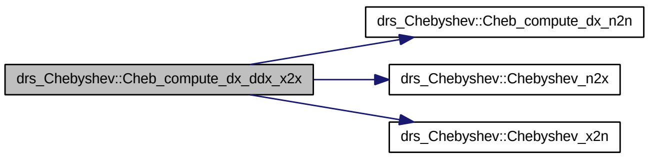
dfdx First derivative of f at points 1..Nx

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:


drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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:

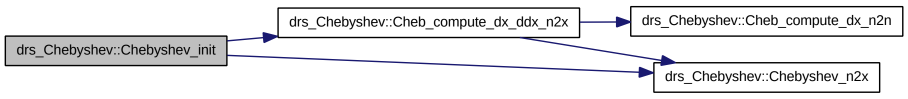
N Number of points the polynomials

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:


drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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 envolving 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 envolving 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().

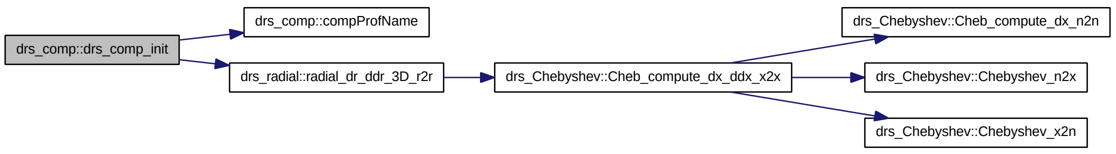
5.3.2.5 subroutine drs_comp::drs_comp_init ()

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:


drs_comp::drs_comp_init
drs_comp::compProfName
drs_radial::radial_dr_ddr_3D_r2r
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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

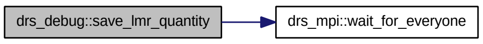
5.4.2.1 subroutine drs_debug::save_lmr_quantity (double precision,dimension(0:nt_s,blk_ps_-size(mpi_rank),intent(in) *t*, character(∗),intent(in) *tname*)

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().

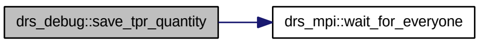
Here is the call graph for this function:



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(), saveI-DLmeridional(), 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(), Bench-markv1(), 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_temprature_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_temprature_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_mspec(), drs_flow::calc_flow_-mspec(), 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** ()

  Destroies 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.

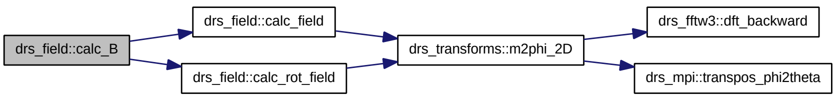
Referenced by drs(), and update_field().

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:

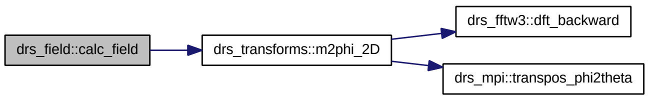


5.7.1.4 subroutine drs_field::calc_field (double precision,dimension(0:blk_t_size(mpi_-rank),intent(out) *Br*, double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *Bt*, double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *Bp*)

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_mspec (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.

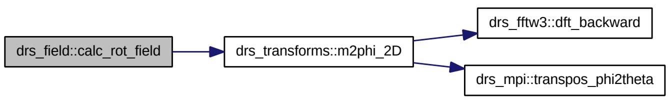
Referenced by drs_io::save_n_spec().

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_ddr, field_pol_dr, field_pol_lap, field_tor, field_tor_avg, field_tor_ddr, field_tor_dr, field_tor_lap, drs_mpi::mpi_rank, drs_dims::Nr, and drs_dims::Nt_s.

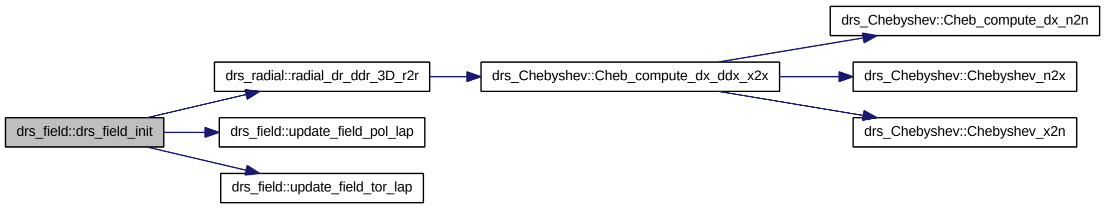
Referenced by drs_init(), and init().

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:


drs_field::drs_field_init
drs_radial::radial_dr_ddr_3D_r2r
drs_field::update_field_pol_lap
drs_field::update_field_tor_lap
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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_ddr

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_ddr

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_ddr**, **flow_pol_dr**, **flow_pol_ddr**)
- 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_ddr**
- double precision, dimension(:,:,:), allocatable **flow_tor_ddr**
- 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:.

$$\vec{u} = \vec{\nabla} \times (\vec{\nabla} \times (\vec{r} flow\_pol)) + \vec{\nabla} \times (r\vec{r} flow\_tor)$$

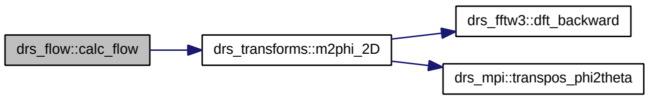
∼∼∼∼∼∼ the fields are defined as: field in program field in equation --------------- --------------- flow_pol = P flow_tor = T/r P id the poloidal scalar, and T the Toroidal scalar

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*)

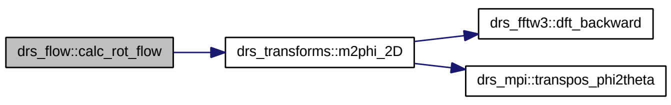
This routine computes:. rotu = rot(rotrot(rP) + rot(rT)) ∼∼∼∼∼∼ the fields are defined as: field in program field in equation --------------- --------------- flow_pol = P flow_tor = T/r P id the poloidal scalar, and T the Toroidal scalar

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,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:



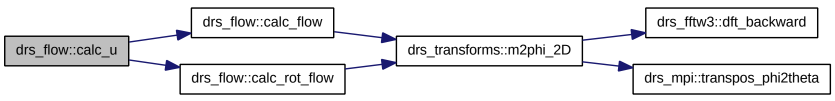
5.8.1.8 subroutine drs_flow::calc_u (double precision,dimension(0:blk_t_size(mpi_-rank),intent(out) *ur*, double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *ut*, double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *up*, double precision,dimension(0:blk_t_size(mpi_rank),intent(out) *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.

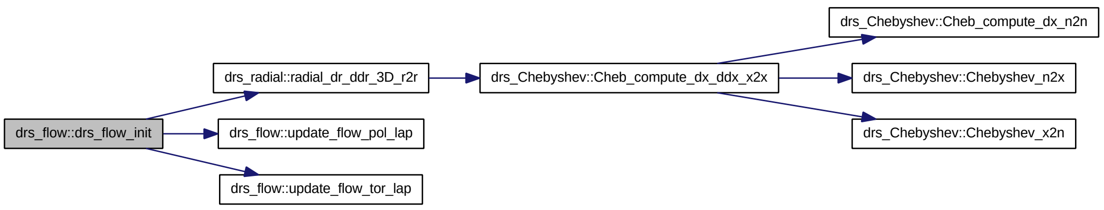
Referenced by drs_init(), and init().

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:


drs_flow::drs_flow_init
drs_radial::radial_dr_ddr_3D_r2r
drs_flow::update_flow_pol_lap
drs_flow::update_flow_tor_lap
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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(), StateAv-erage(), 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(), StateAv-erage(), 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

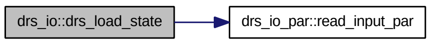
5.10.2.1 subroutine drs_io::drs_load_state (integer,intent(out) *error*)

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:



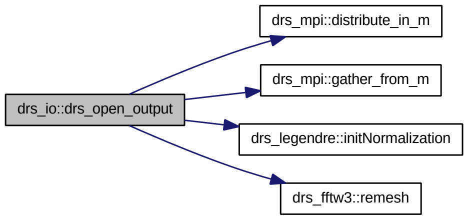
5.10.2.2 subroutine drs_io::drs_open_output ()

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:

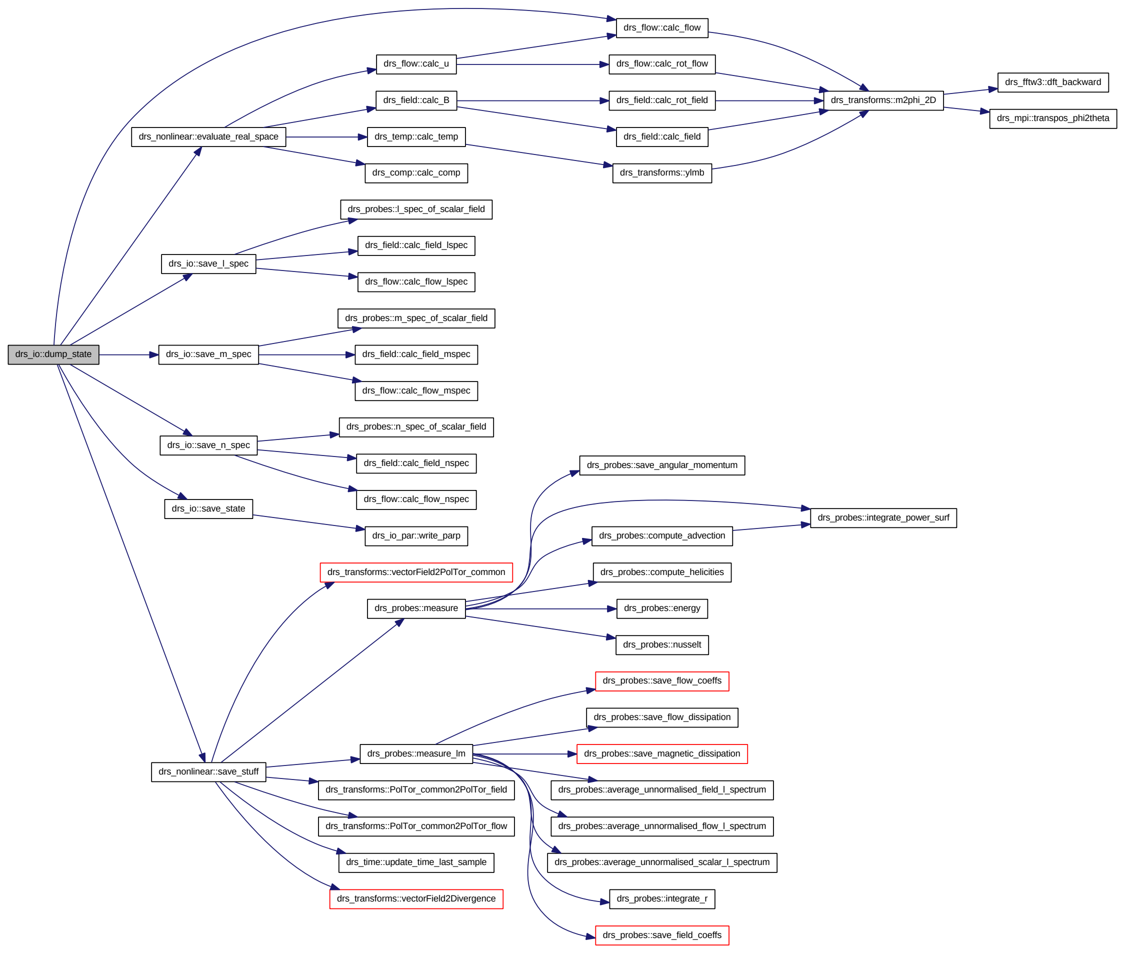


5.10.2.3 subroutine drs_io::dump_state ()

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:


drs_io::dump_state
drs_nonlinear::evaluate_real_space
drs_flow::calc_u
drs_field::calc_B
drs_temp::calc_temp
drs_comp::calc_comp
drs_flow::calc_flow
drs_flow::calc_rot_flow
drs_field::calc_rot_field
drs_field::calc_field
drs_transforms::ylmb
drs_transforms::m2phi_2D
drs_fftw3::dft_backward
drs_mpi::transpos_phi2theta
drs_io::save_l_spec
drs_probes::l_spec_of_scalar_field
drs_field::calc_field_lspec
drs_flow::calc_flow_lspec
drs_io::save_m_spec
drs_probes::m_spec_of_scalar_field
drs_field::calc_field_mspec
drs_flow::calc_flow_mspec
drs_io::save_n_spec
drs_probes::n_spec_of_scalar_field
drs_field::calc_field_nspec
drs_flow::calc_flow_nspec
drs_io::save_state
drs_io_par::write_parp
drs_transforms::vectorField2PolTor_common
drs_probes::measure
drs_probes::save_angular_momentum
drs_probes::integrate_power_surf
drs_probes::compute_advection
drs_probes::compute_helicities
drs_probes::energy
drs_probes::nusselt
drs_nonlinear::save_stuff
drs_probes::measure_lm
drs_probes::save_flow_coeffs
drs_probes::save_flow_dissipation
drs_probes::save_magnetic_dissipation
drs_probes::average_unnormalised_field_l_spectrum
drs_probes::average_unnormalised_flow_l_spectrum
drs_probes::average_unnormalised_scalar_l_spectrum
drs_probes::integrate_r
drs_probes::save_field_coeffs
drs_transforms::PolTor_common2PolTor_field
drs_transforms::PolTor_common2PolTor_flow
drs_time::update_time_last_sample
drs_transforms::vectorField2Divergence

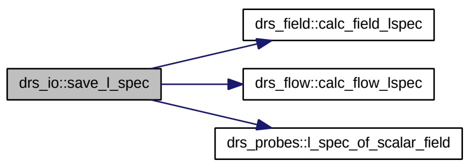
5.10.2.4 subroutine drs_io::save_l_spec ()

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:


drs_io::save_l_spec
drs_field::calc_field_lspec
drs_flow::calc_flow_lspec
drs_probes::l_spec_of_scalar_field

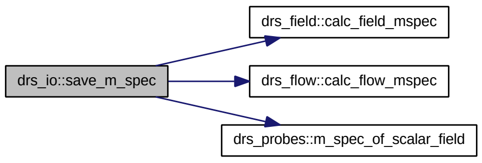
5.10.2.5 subroutine drs_io::save_m_spec ()

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:


drs_field::calc_field_mspec
drs_io::save_m_spec
drs_flow::calc_flow_mspec
drs_probes::m_spec_of_scalar_field

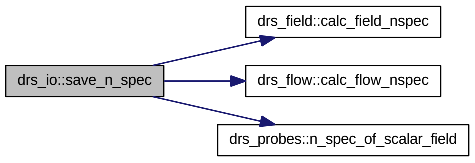
5.10.2.6 subroutine drs_io::save_n_spec ()

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:


drs_field::calc_field_nspec
drs_io::save_n_spec
drs_flow::calc_flow_nspec
drs_probes::n_spec_of_scalar_field

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*.

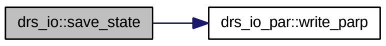
Todo

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 rendeer of the field.

- subroutine **saveDXvolume_v2** (field, filename)

  Writes a volume rendeer of the field.

- subroutine **saveDXvolume3DVec** (XX, YY, ZZ, filename)

  Writes a volume rendeer 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*)

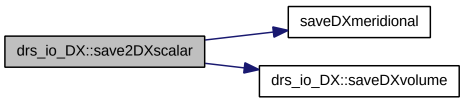
Saves the contents of a scalar field to file.

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*)

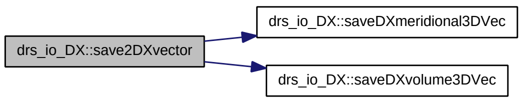
Saves the contents of a vector field to file given its three components.

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:


drs_io_DX::save2DXvector
drs_io_DX::saveDXmeridional3DVec
drs_io_DX::saveDXvolume3DVec

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 rendeer 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 rendeer 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 rendeer 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 parallell 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

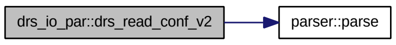
References drs_time::cpu_max_time, drs_time::h, drs_dims::lsymm, drs_dims::m0, magic, MAGICC5, MAGICC9, noise, drs_dims::Np, drs_dims::Np_s, drs_dims::Nr, drs_dims::Nr_s, drs_dims::Nt, drs_dims::Nt_s, drs_time::sampling_rate, drs_time::stepmax, and drs_time::transient.

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_mspec** = 41
- integer, parameter **unit_lspec** = 42
- integer, parameter **unit_nspec** = 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 Guass-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 Plm(cos(theta))/d theta

- double precision, dimension(:,:,:), allocatable **leg_sin**

  Plm/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().

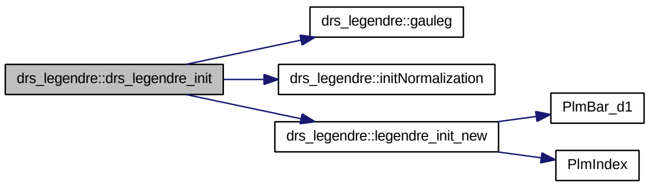
5.14.1.2 subroutine drs_legendre::drs_legendre_init ()

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:


drs_legendre::gauleg
drs_legendre::drs_legendre_init
drs_legendre::initNormalization
PlmBar_d1
drs_legendre::legendre_init_new
PlmIndex

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 Guass-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, \[(P_l^m)^2 = 2∗{(l+m)!}{(l-m)!(2l+1)} \].

lmax normalization types:

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

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

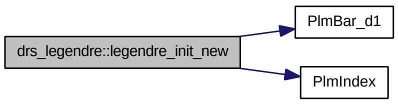
5.14.1.5 subroutine drs_legendre::legendre_init_new ()

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 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

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

sqrt((l+m)!/(l-m)!/(2l+1)) = sqrt((l-m+1)∗(l-m+2)∗...∗((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 openned 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.

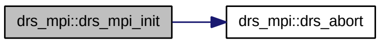
5.16.2.14 subroutine drs_mpi::drs_mpi_init ()

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().

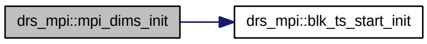
5.16.2.17 subroutine drs_mpi::mpi_dims_init (integer,intent(in) *Nt*, integer,intent(in) *Np_s*, integer,intent(in) *m0*, integer,intent(out) *error*)

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 dor 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 convinience 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_-mspec(), drs_flow::calc_flow_lspec(), drs_flow::calc_flow_mspec(), 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 contructing 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 contructing 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.

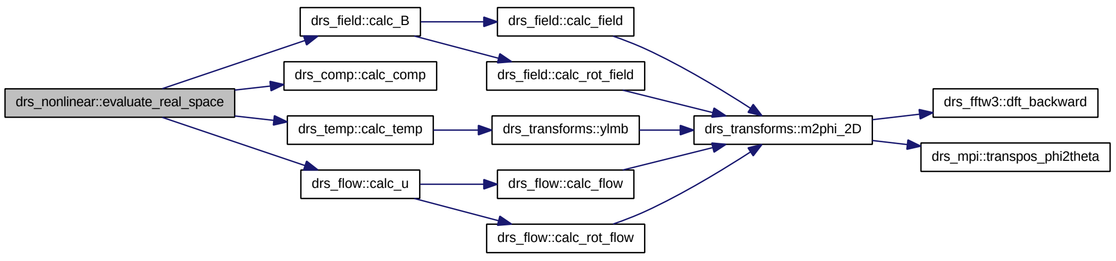
Referenced by drs_init().

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:

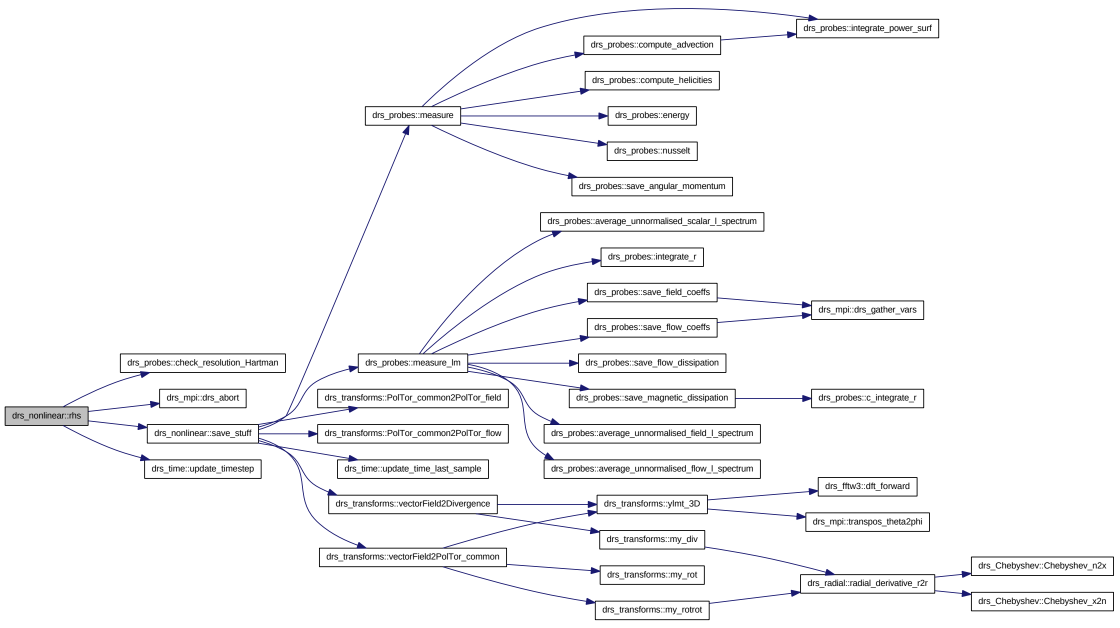


5.17.2.3 subroutine drs_nonlinear::rhs (double precision,intent(out) *h_old*, double precision,intent(inout) *h*)

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:


drs_nonlinear::rhs
drs_probes::check_resolution_Hartman
drs_mpi::drs_abort
drs_nonlinear::save_stuff
drs_time::update_timestep
drs_probes::measure
drs_probes::measure_lm
drs_transforms::PolTor_common2PolTor_field
drs_transforms::PolTor_common2PolTor_flow
drs_time::update_time_last_sample
drs_transforms::vectorField2Divergence
drs_transforms::vectorField2PolTor_common
drs_probes::integrate_power_surf
drs_probes::compute_advection
drs_probes::compute_helicities
drs_probes::energy
drs_probes::nusselt
drs_probes::save_angular_momentum
drs_probes::average_unnormalised_scalar_l_spectrum
drs_probes::integrate_r
drs_probes::save_field_coeffs
drs_probes::save_flow_coeffs
drs_mpi::drs_gather_vars
drs_probes::save_flow_dissipation
drs_probes::save_magnetic_dissipation
drs_probes::c_integrate_r
drs_probes::average_unnormalised_field_l_spectrum
drs_probes::average_unnormalised_flow_l_spectrum
drs_transforms::ylmt_3D
drs_fftw3::dft_forward
drs_mpi::transpos_theta2phi
drs_transforms::my_div
drs_transforms::my_rot
drs_transforms::my_rotrot
drs_radial::radial_derivative_r2r
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

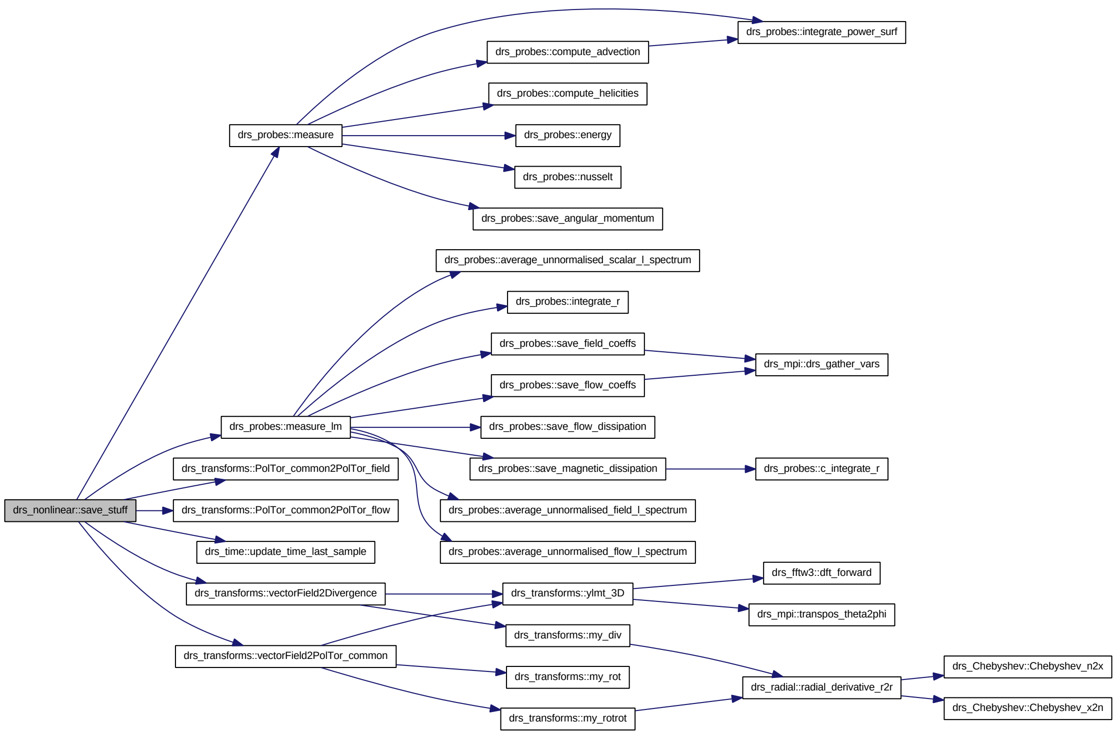
5.17.2.4 subroutine drs_nonlinear::save_stuff (integer,intent(inout) *nsample*)

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:


drs_nonlinear::save_stuff
drs_probes::measure
drs_probes::integrate_power_surf
drs_probes::compute_advection
drs_probes::compute_helicities
drs_probes::energy
drs_probes::nusselt
drs_probes::save_angular_momentum
drs_probes::measure_lm
drs_probes::average_unnormalised_scalar_l_spectrum
drs_probes::integrate_r
drs_probes::save_field_coeffs
drs_probes::save_flow_coeffs
drs_mpi::drs_gather_vars
drs_probes::save_flow_dissipation
drs_probes::save_magnetic_dissipation
drs_probes::c_integrate_r
drs_probes::average_unnormalised_field_l_spectrum
drs_probes::average_unnormalised_flow_l_spectrum
drs_transforms::PolTor_common2PolTor_field
drs_transforms::PolTor_common2PolTor_flow
drs_time::update_time_last_sample
drs_transforms::vectorField2Divergence
drs_transforms::ylmt_3D
drs_fftw3::dft_forward
drs_mpi::transpos_theta2phi
drs_transforms::my_div
drs_transforms::vectorField2PolTor_common
drs_transforms::my_rot
drs_transforms::my_rotrot
drs_radial::radial_derivative_r2r
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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

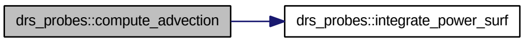
$$Q(r) = \int\int u_r(r,\theta,\phi) * (\Theta(r,\theta,\phi) + T_S(r)) \sin\theta d\theta d\phi$$

.

References integrate_power_surf(), and drs_temp::temp_profile.

Referenced by measure().

Here is the call graph for this function:


drs_probes::compute_advection
drs_probes::integrate_power_surf

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.

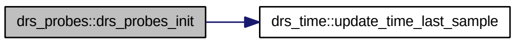
Referenced by drs_init(), and init().

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:


drs_probes::drs_probes_init
drs_time::update_time_last_sample

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 phy of a function f raised to the power n. as

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

.

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.

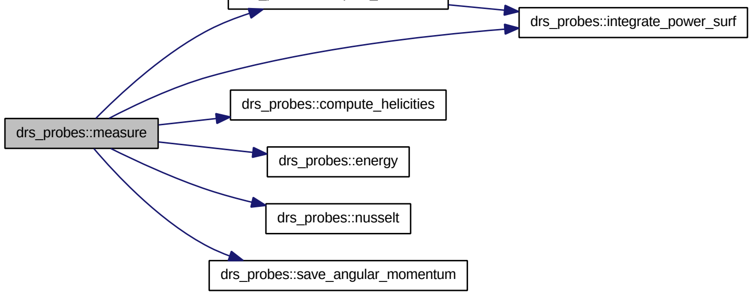
Referenced by drs_io::save_m_spec().

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:

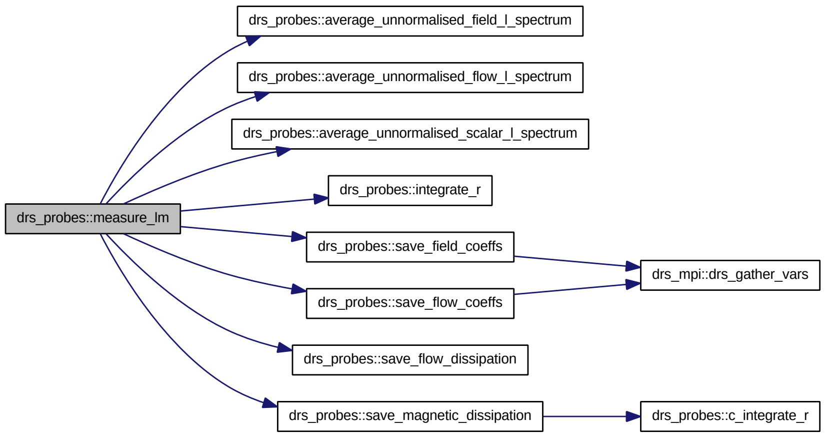

drs_probes::measure
drs_probes::compute_advection
drs_probes::integrate_power_surf
drs_probes::compute_helicities
drs_probes::energy
drs_probes::nusselt
drs_probes::save_angular_momentum

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:


drs_probes::measure_lm
drs_probes::average_unnormalised_field_l_spectrum
drs_probes::average_unnormalised_flow_l_spectrum
drs_probes::average_unnormalised_scalar_l_spectrum
drs_probes::integrate_r
drs_probes::save_field_coeffs
drs_probes::save_flow_coeffs
drs_probes::save_flow_dissipation
drs_probes::save_magnetic_dissipation
drs_mpi::drs_gather_vars
drs_probes::c_integrate_r

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 in 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 the ratio between the convective and the diffusive heat fluxes.

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

Todo

Ito 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.

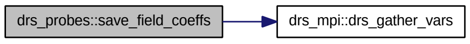
Todo

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.

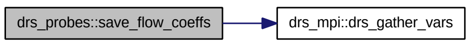
Todo

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*)

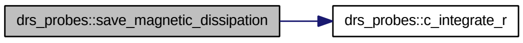
Computes the magnetic dissipation truncated up to degree.

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:


drs_probes::save_magnetic_dissipation
drs_probes::c_integrate_r

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

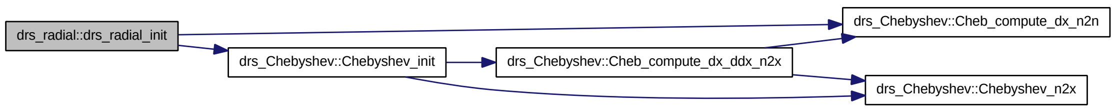
5.19.2.1 subroutine drs_radial::drs_radial_init (double precision,intent(in) *riro*)

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:


drs_radial::drs_radial_init
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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.

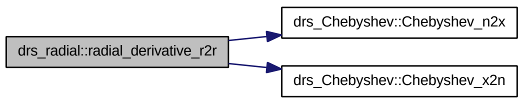
Since:

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:


drs_radial::radial_derivative_r2r
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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) *t1*, double precision,dimension(nr),intent(out) *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.

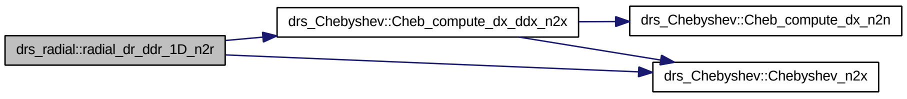
Since:

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:


drs_radial::radial_dr_ddr_1D_n2r
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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.

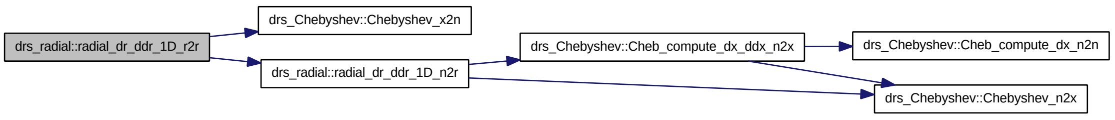
Since:

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:


drs_radial::radial_dr_ddr_1D_r2r
drs_Chebyshev::Chebyshev_x2n
drs_radial::radial_dr_ddr_1D_n2r
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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.

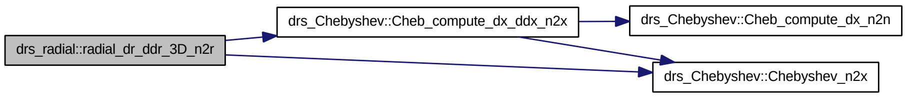
Since:

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:


drs_radial::radial_dr_ddr_3D_n2r
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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.

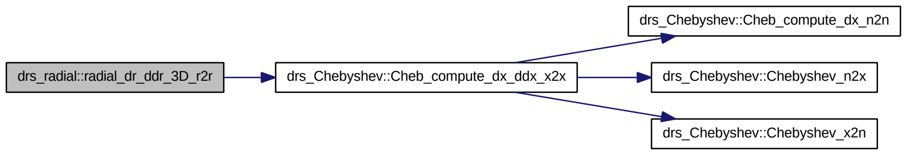
Since:

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:


drs_radial::radial_dr_ddr_3D_r2r
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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_ddr

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(), saveI-DLmeridional(), selectEquatorMidShell(), test_radial_colocation_points(), test_radial_derivative_r2r(), test_radial_dr_ddr_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_ddr_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 ploted | |> 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 ploted | |> 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 fiels 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

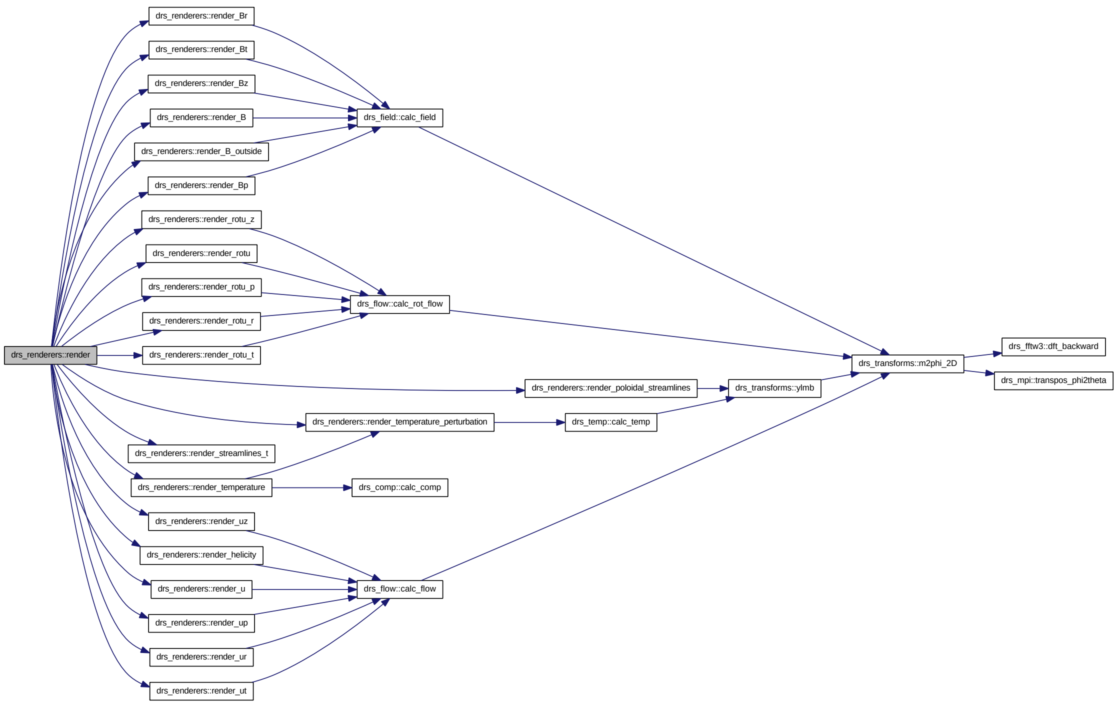
Parameters:

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:



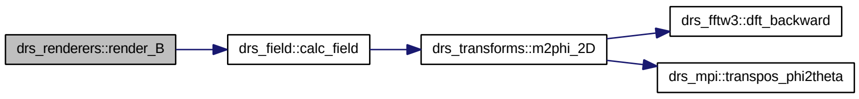
5.20.1.3 subroutine drs_renderers::render_B ()

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:



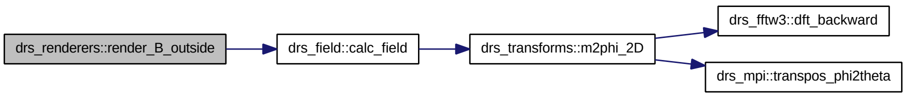
5.20.1.4 subroutine drs_renderers::render_B_outside ()

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:



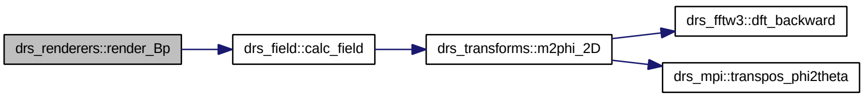
5.20.1.5 subroutine drs_renderers::render_Bp ()

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:

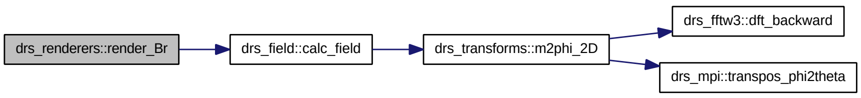


5.20.1.6 subroutine drs_renderers::render_Br ()

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:

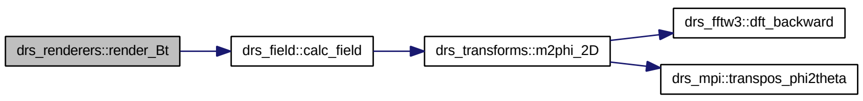


5.20.1.7 subroutine drs_renderers::render_Bt ()

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:



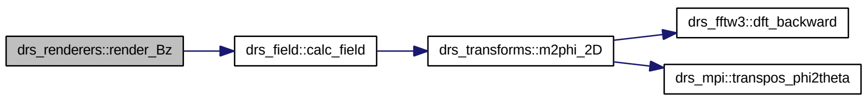
5.20.1.8 subroutine drs_renderers::render_Bz ()

Renders the z component of the magnetic field.

References drs_field::calc_field(), drs_legendre::costheta, drs_field::field_pol, 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_-legendre::pi, and render_out.

Referenced by render().

Here is the call graph for this function:



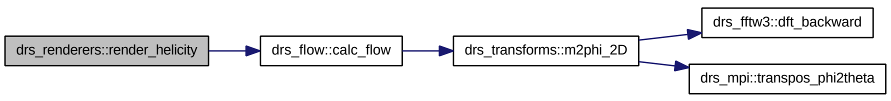
5.20.1.9 subroutine drs_renderers::render_helicity ()

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:



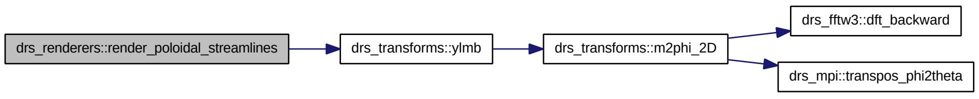
5.20.1.10 subroutine drs_renderers::render_poloidal_streamlines ()

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:

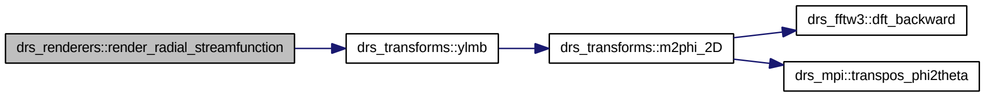

drs_renderers::render_poloidal_streamlines
drs_transforms::ylmb
drs_transforms::m2phi_2D
drs_fftw3::dft_backward
drs_mpi::transpos_phi2theta

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:


drs_renderers::render_radial_streamfunction
drs_transforms::ylmb
drs_transforms::m2phi_2D
drs_fftw3::dft_backward
drs_mpi::transpos_phi2theta

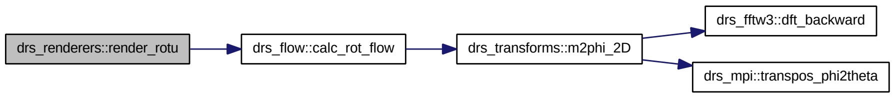
5.20.1.12 subroutine drs_renderers::render_rotu ()

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:



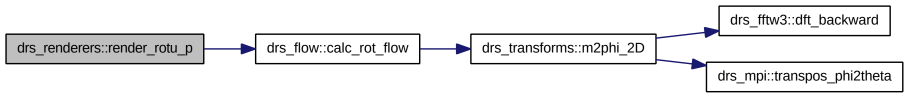
5.20.1.13 subroutine drs_renderers::render_rotu_p ()

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:



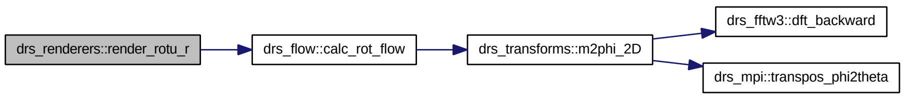
5.20.1.14 subroutine drs_renderers::render_rotu_r ()

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:



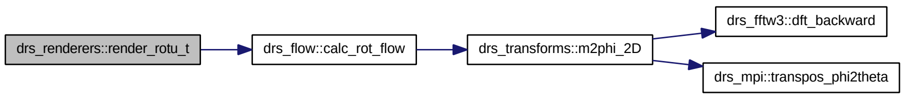
5.20.1.15 subroutine drs_renderers::render_rotu_t ()

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:



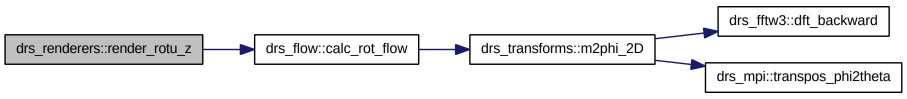
5.20.1.16 subroutine drs_renderers::render_rotu_z ()

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().

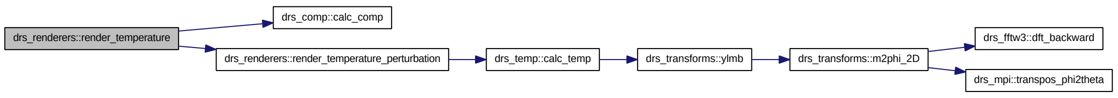
5.20.1.18 subroutine drs_renderers::render_temperature ()

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:


drs_renderers::render_temperature
drs_comp::calc_comp
drs_renderers::render_temperature_perturbation
drs_temp::calc_temp
drs_transforms::ylmb
drs_transforms::m2phi_2D
drs_fftw3::dft_backward
drs_mpi::transpos_phi2theta

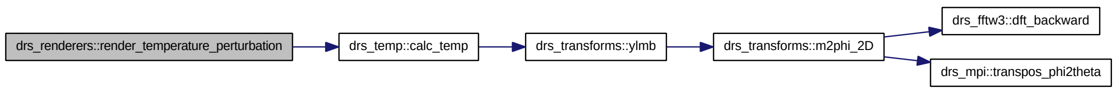
5.20.1.19 subroutine drs_renderers::render_temperature_perturbation ()

Renders the temperature perturbation.

References drs_temp::calc_temp(), and render_out.

Referenced by render(), and render_temperature().

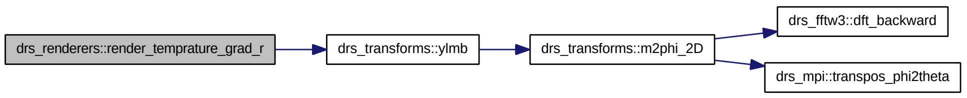
Here is the call graph for this function:



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:

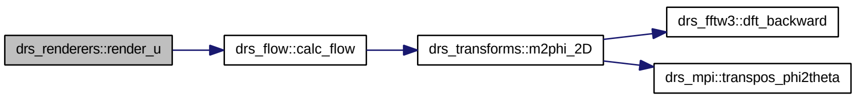


5.20.1.21 subroutine drs_renderers::render_u ()

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:



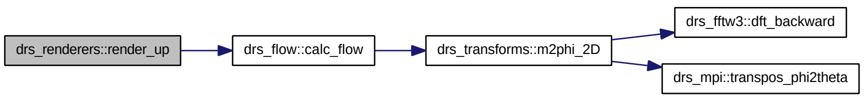
5.20.1.22 subroutine drs_renderers::render_up ()

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:

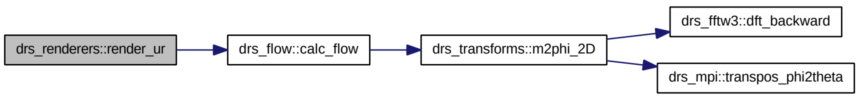


5.20.1.23 subroutine drs_renderers::render_ur ()

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:



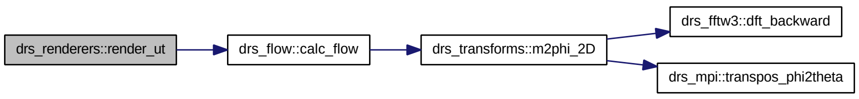
5.20.1.24 subroutine drs_renderers::render_ut ()

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:



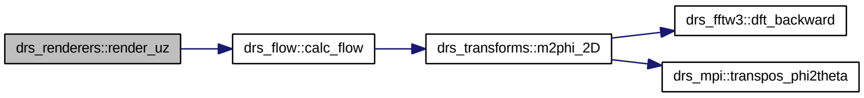
5.20.1.25 subroutine drs_renderers::render_uz ()

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_ddr**
- 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().

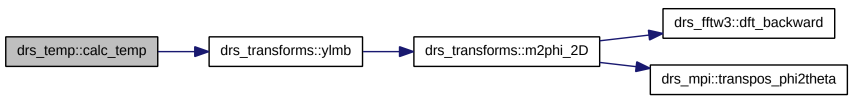
5.21.2.2 subroutine drs_temp::calc_temp (double precision,dimension(0:blk_t_size(mpi_-rank),intent(out) *temp_t*)

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().

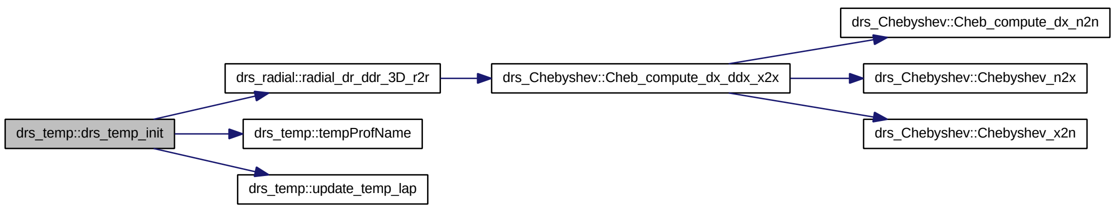
5.21.2.4 subroutine drs_temp::drs_temp_init ()

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:


drs_temp::drs_temp_init
drs_radial::radial_dr_ddr_3D_r2r
drs_temp::tempProfName
drs_temp::update_temp_lap
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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_temprature_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 wit 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 wit 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_par().

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_par(), 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) --> tt_t(theta,m) transposed. 2. redistribution dist(theta) --> dist(m) 3. transformation tt(theta,m) --> t(l,m) distrib. in m.

- subroutine **ylmt_3D** (input, 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.

- 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 apropriate 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 apropriate 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.

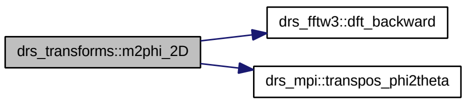
5.23.2 Function Documentation

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.

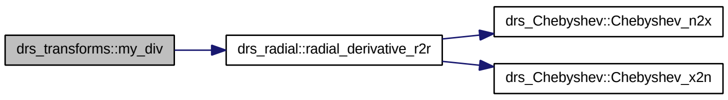
vec_p is $\frac{u_\phi}{r\sin\theta}$ in (l,m,r) space. On output:

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:


drs_transforms::my_div
drs_radial::radial_derivative_r2r
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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.

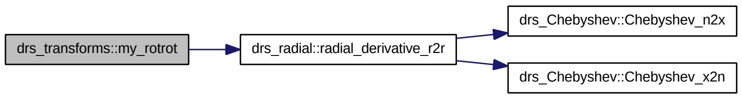
Referenced by vectorField2PolTor_common().

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:


drs_transforms::my_rotrot
drs_radial::radial_derivative_r2r
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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 apropriate 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 apropriate 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().

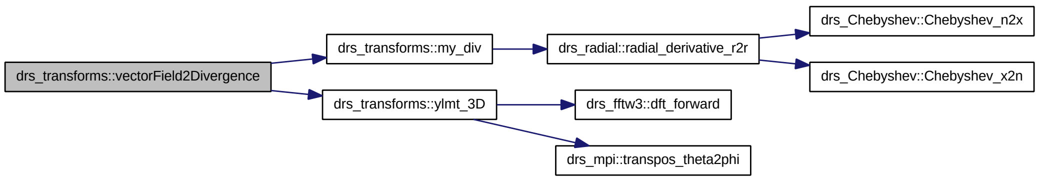
5.23.2.7 subroutine drs_transforms::vectorField2Divergence (double precision,dimension(0:blk_t_size(mpi_rank),intent(inout) *ur*, double precision,dimension(0:blk_t_size(mpi_rank),intent(inout) *ut*, double precision,dimension(0:blk_t_size(mpi_rank),intent(inout) *up*, double precision,dimension(0:nt_s,blk_ps_size(mpi_rank),intent(out) *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:


drs_transforms::vectorField2Divergence
drs_transforms::my_div
drs_transforms::ylmt_3D
drs_radial::radial_derivative_r2r
drs_fftw3::dft_forward
drs_mpi::transpos_theta2phi
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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*)

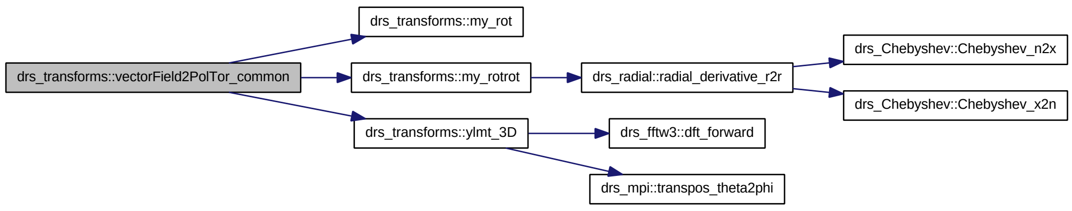
Computes the poloidal and toroidal scalar coefficients of a 3D vector field.

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:

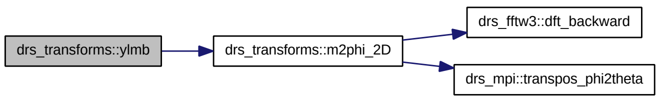

drs_transforms::vectorField2PolTor_common
drs_transforms::my_rot
drs_transforms::my_rotrot
drs_transforms::ylmt_3D
drs_radial::radial_derivative_r2r
drs_fftw3::dft_forward
drs_mpi::transpos_theta2phi
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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_temprature_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*)

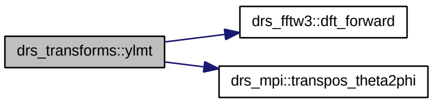
This routine performs three steps: ~~~~~~ 1. transformation tt_t(theta,phi) --> tt_t(theta,m) transposed. 2. redistribution dist(theta) --> dist(m) 3. transformation tt(theta,m) --> t(l,m) distrib. in m. input: tt_t(0:(blk_t_max_size-1),Np) (transposed) has to be in direct space, in phi-direction t(:,1:Np) only one period is stored: tt_t(.,j) means position phi=(2pi/m0)∗(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:


drs_transforms::ylmt
drs_fftw3::dft_forward
drs_mpi::transpos_theta2phi

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=(2pi/m0)∗(j-1)/Np.

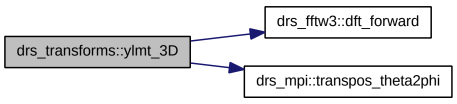
output: t(0:drs_Nt,drs_Np) dims: (0:Nt_s,Np_s) dealiased in (lm)! in m-dir.: real(m=0),real(m=m0),im(m=m0),real(m=2m0),im(m=2∗m0),... in l-dir.: includes normalization factors!

~~~~~~

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**

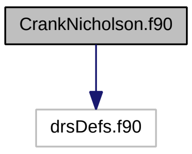
Chapter 7

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.

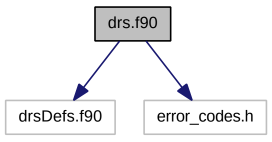
- double precision, dimension(:,:,:), allocatable, target **CrankNicholson::pinv**

  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 stoping 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.

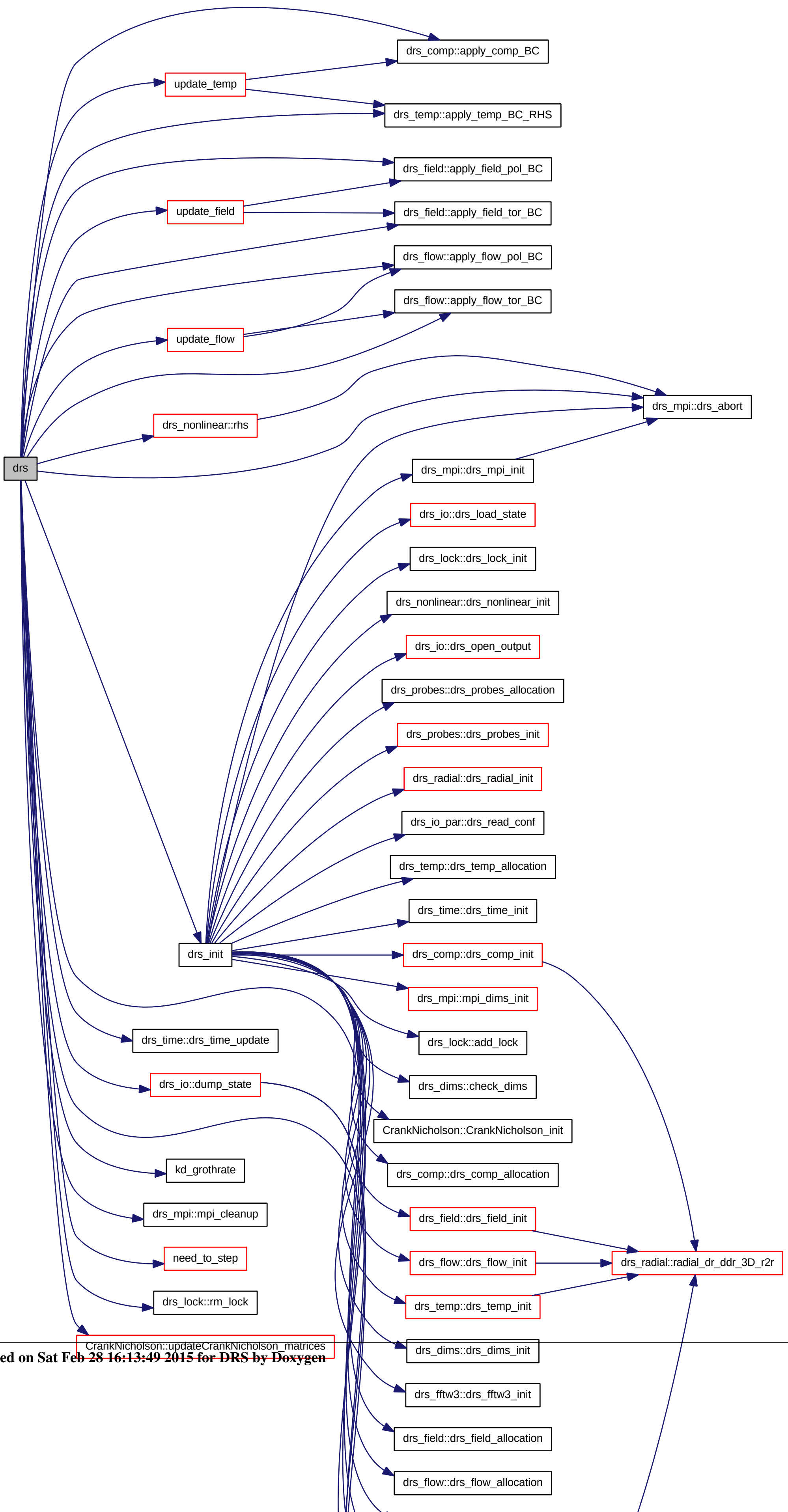
References drs_flow::flow_pol, drs_flow::flow_pol_ddr, drs_flow::flow_pol_dr, and drs_dims::Nr.

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:


drs
drs_comp::apply_comp_BC
update_temp
drs_temp::apply_temp_BC_RHS
drs_field::apply_field_pol_BC
update_field
drs_field::apply_field_tor_BC
drs_flow::apply_flow_pol_BC
drs_flow::apply_flow_tor_BC
update_flow
drs_mpi::drs_abort
drs_nonlinear::rhs
drs_mpi::drs_mpi_init
drs_io::drs_load_state
drs_lock::drs_lock_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_time::drs_time_init
drs_init
drs_comp::drs_comp_init
drs_mpi::mpi_dims_init
drs_lock::add_lock
drs_time::drs_time_update
drs_dims::check_dims
drs_io::dump_state
CrankNicholson::CrankNicholson_init
drs_comp::drs_comp_allocation
kd_grothrate
drs_field::drs_field_init
drs_mpi::mpi_cleanup
need_to_step
drs_flow::drs_flow_init
drs_radial::radial_dr_ddr_3D_r2r
drs_lock::rm_lock
drs_temp::drs_temp_init
CrankNicholson::updateCrankNicholson_matrices
drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_field::drs_field_allocation
drs_flow::drs_flow_allocation

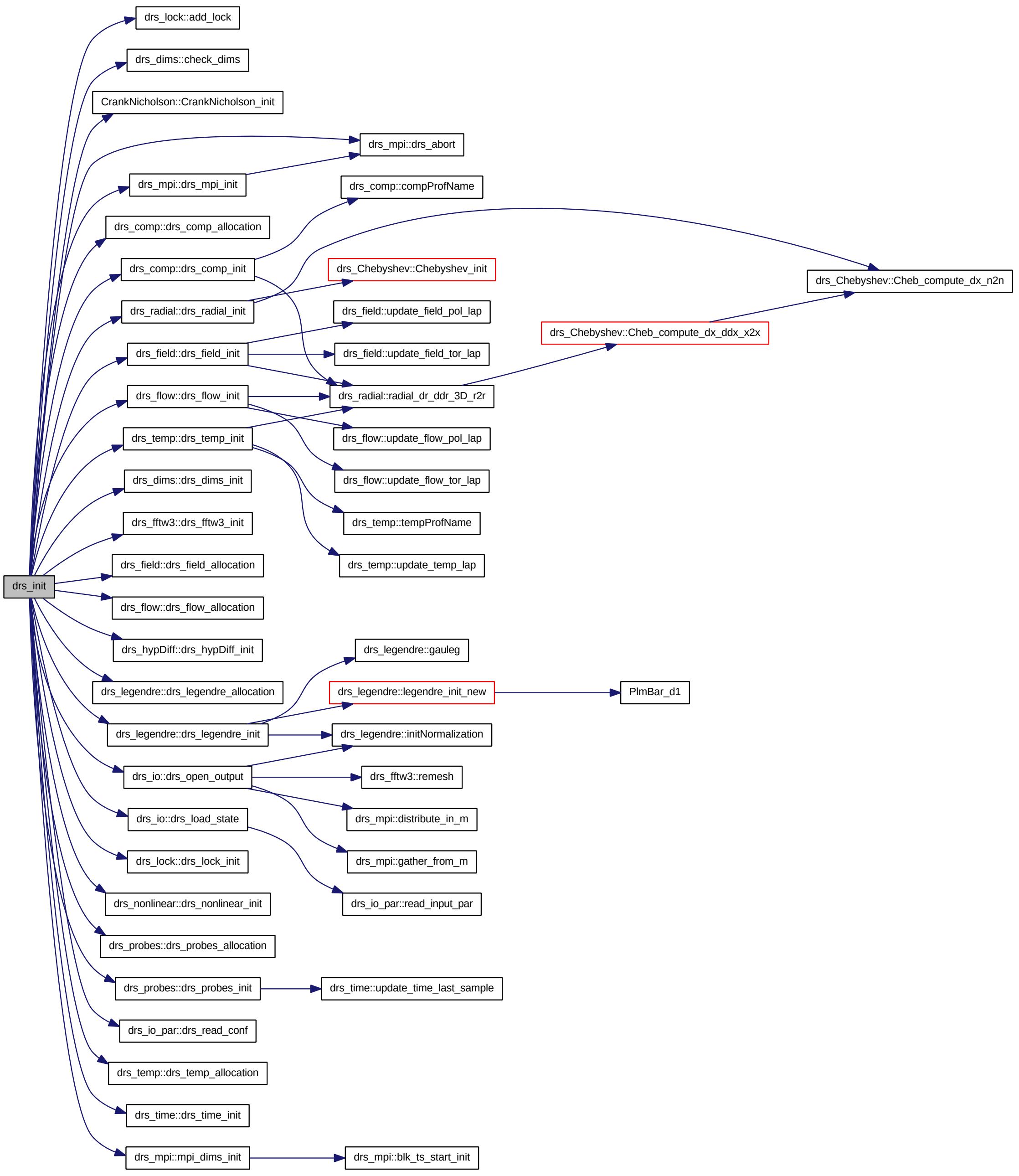
7.2.1.3 subroutine drs::drs_init (integer,intent(inout) *error*)

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:


drs_init
drs_lock::add_lock
drs_dims::check_dims
CrankNicholson::CrankNicholson_init
drs_mpi::drs_abort
drs_mpi::drs_mpi_init
drs_comp::compProfName
drs_comp::drs_comp_allocation
drs_comp::drs_comp_init
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_n2n
drs_radial::drs_radial_init
drs_field::update_field_pol_lap
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_field::drs_field_init
drs_field::update_field_tor_lap
drs_flow::drs_flow_init
drs_radial::radial_dr_ddr_3D_r2r
drs_temp::drs_temp_init
drs_flow::update_flow_pol_lap
drs_dims::drs_dims_init
drs_flow::update_flow_tor_lap
drs_fftw3::drs_fftw3_init
drs_temp::tempProfName
drs_field::drs_field_allocation
drs_temp::update_temp_lap
drs_flow::drs_flow_allocation
drs_hypDiff::drs_hypDiff_init
drs_legendre::gauleg
drs_legendre::drs_legendre_allocation
drs_legendre::legendre_init_new
PlmBar_d1
drs_legendre::drs_legendre_init
drs_legendre::initNormalization
drs_io::drs_open_output
drs_fftw3::remesh
drs_io::drs_load_state
drs_mpi::distribute_in_m
drs_lock::drs_lock_init
drs_mpi::gather_from_m
drs_nonlinear::drs_nonlinear_init
drs_io_par::read_input_par
drs_probes::drs_probes_allocation
drs_probes::drs_probes_init
drs_time::update_time_last_sample
drs_io_par::drs_read_conf
drs_temp::drs_temp_allocation
drs_time::drs_time_init
drs_mpi::mpi_dims_init
drs_mpi::blk_ts_start_init

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, 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().

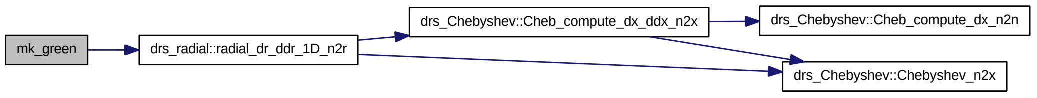
7.2.1.5 subroutine drs::mk_green (double precision,intent(in) *ib*, double precision,intent(in) *ob*, double precision,dimension(nr, 0:nt_s),intent(out) *green*, double precision,dimension(nr, 0:nt_s),intent(out) *greenD*, double precision,dimension(nr, 0:nt_s),intent(out) *greenS*)

- a stopping condition on number of steps;
- a stopping condition on cpu time; - a stoping condition on the existence of a lock file; - a stoping 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:


mk_green
drs_radial::radial_dr_ddr_1D_n2r
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

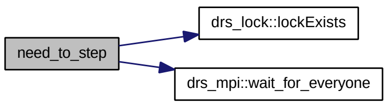
7.2.1.6 logical drs::need_to_step ()

This function will determine whether we need to take another time-step or not. There are several stoping 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

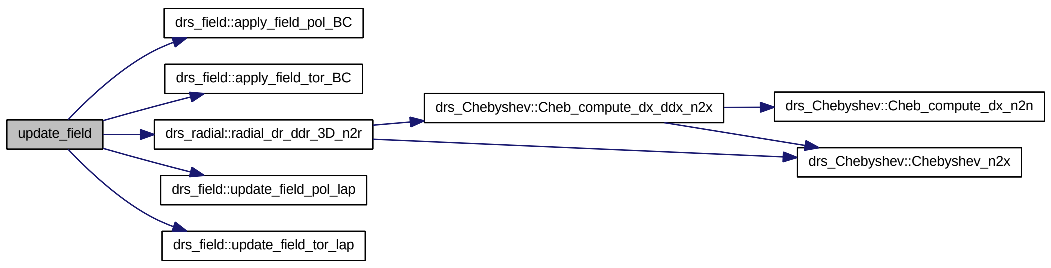
h2 are the Addams-Bashford weights;

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:


update_field
drs_field::apply_field_pol_BC
drs_field::apply_field_tor_BC
drs_radial::radial_dr_ddr_3D_n2r
drs_field::update_field_pol_lap
drs_field::update_field_tor_lap
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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;

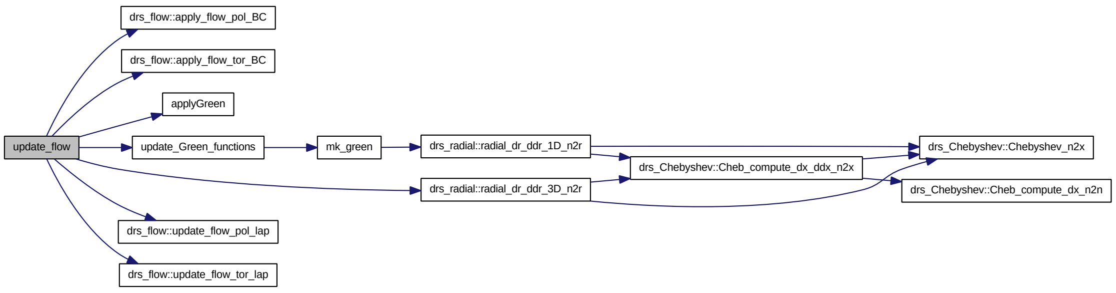
h1 and

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:


drs_flow::apply_flow_pol_BC
drs_flow::apply_flow_tor_BC
applyGreen
update_flow
update_Green_functions
mk_green
drs_radial::radial_dr_ddr_1D_n2r
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x
drs_radial::radial_dr_ddr_3D_n2r
drs_Chebyshev::Cheb_compute_dx_n2n
drs_flow::update_flow_pol_lap
drs_flow::update_flow_tor_lap

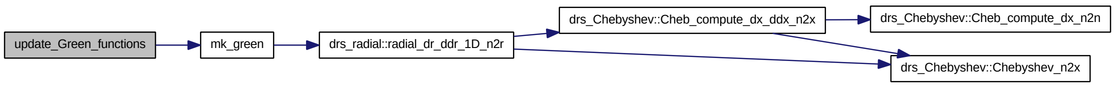
7.2.1.9 subroutine drs::update_Green_functions ()

Encapsulates dealing with the Green's functions.

References mk_green().

Referenced by update_flow().

Here is the call graph for this function:


update_Green_functions
mk_green
drs_radial::radial_dr_ddr_1D_n2r
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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

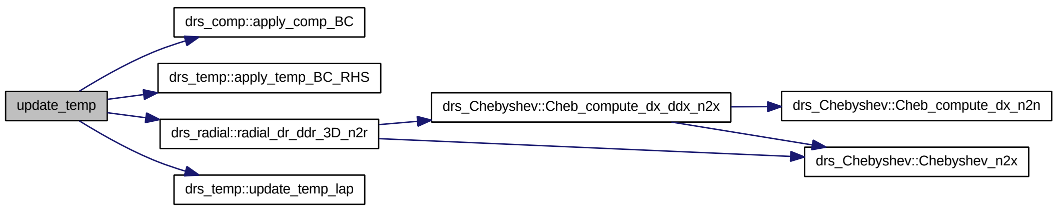
h2 are the Addams-Bashford weights;

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:


drs_comp::apply_comp_BC
drs_temp::apply_temp_BC_RHS
update_temp
drs_radial::radial_dr_ddr_3D_n2r
drs_temp::update_temp_lap
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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 Chebychev 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.

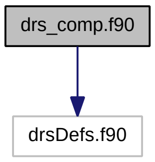
- integer $*8$ **drs_Chebyshev::plan_x**
- integer **drs_Chebyshev::Nx**
- integer **drs_Chebyshev::Nx_s**
- double precision, dimension(:), allocatable **drs_Chebyshev::ct_buffer**
- double precision, allocatable **drs_Chebyshev::Cheb_x**
- double precision, allocatable, target **drs_Chebyshev::Chebyshev**
- 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 envolving 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

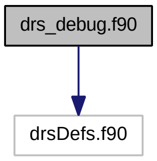
- double precision, dimension(:,:,:), allocatable **drs_comp::comp**
- double precision, dimension(:,:,:), allocatable **drs_comp::comp_dr**

- 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.

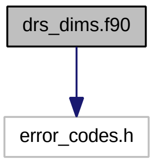
Functions

- 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** ()

  Destroies 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**

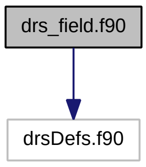
  It makes use of fftw3.

- 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

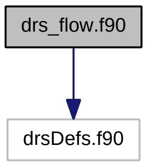
- double precision, dimension(:,:,:), allocatable **drs_field::field_pol**
- double precision, dimension(:,:,:), allocatable **drs_field::field_tor**
- double precision, dimension(:,:,:), allocatable **drs_field::field_pol_dr**
- double precision, dimension(:,:,:), allocatable **drs_field::field_tor_dr**
- 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)

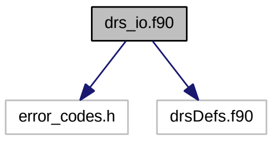
Variables

- 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.

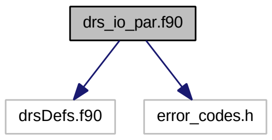
Variables

- 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)
- 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.

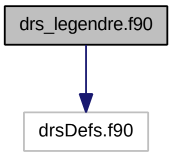
Variables

- 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_mspec** = 41
- integer, parameter **drs_io_units::unit_lspec** = 42
- integer, parameter **drs_io_units::unit_nspec** = 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 Guass-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**

Plm/sin(theta).

- double precision, dimension(:,:), allocatable **drs_legendre::plmfac**

  sqrt((l+m)!/(l-m)!/(2l+1)) = sqrt((l-m+1)∗(l-m+2)∗...∗((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**

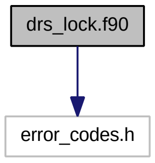
  Table of $l(l+1)$.

- 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.

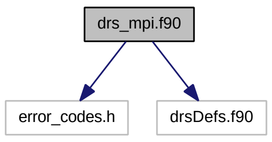
Variables

- 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 distrib(phi) --> tt_t distrib(theta):

- subroutine **drs_mpi::transpos_theta2phi** (input, Np_s, output, Nt)

  transposition: tt_t distrib(theta) --> t 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.

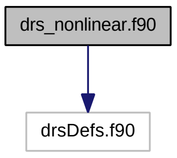
- integer **drs_mpi::blk_ps_max_size**

  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 contructing 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.

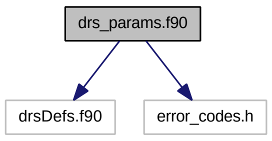
- double precision, dimension(:,:,:), allocatable **drs_nonlinear::flow_t_t**
- double precision, dimension(:,:,:), allocatable **drs_nonlinear::flow_p_t**
- double precision, dimension(:,:,:), allocatable **drs_nonlinear::field_r_t**
- double precision, dimension(:,:,:), allocatable **drs_nonlinear::field_t_t**
- 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:

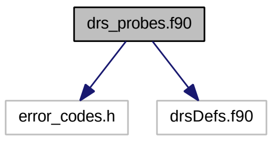


7.19 drs_probes.f90 File Reference

`#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**

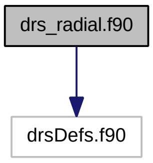
  Weights for volume integration.

- 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.

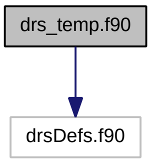
- double precision, dimension(:,:), allocatable **drs_radial::poly**
- double precision, dimension(:,:), allocatable **drs_radial::poly_dr**
- double precision, dimension(:,:), allocatable **drs_radial::poly_ddr**
- integer, dimension(2) **drs_radial::b**

  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

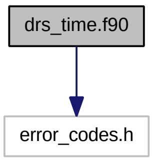
- double precision, dimension(:,:,:), allocatable **drs_temp::temp**
- double precision, dimension(:,:,:), allocatable **drs_temp::temp_dr**
- double precision, dimension(:,:,:), allocatable **drs_temp::temp_ddr**

- 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**

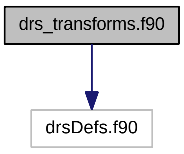
  The step count we start the run with.

- 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) --> tt_t(theta,m) transposed. 2. redistribution dist(theta) --> dist(m) 3. transformation tt(theta,m) --> t(l,m) distrib. in m.

- subroutine **drs_transforms::ylmt_3D** (input, 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.

- 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 apropriate 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 apropriate 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.

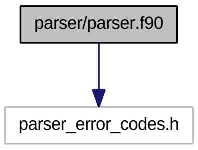
Variables

- 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

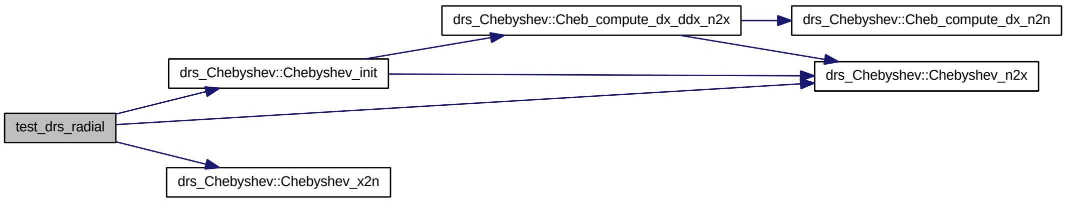
- program **test_drs_radial**

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:


drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Chebyshev_n2x
test_drs_radial
drs_Chebyshev::Chebyshev_x2n

7.29 tests/test_drs_fftw-r2r.f90 File Reference

Functions

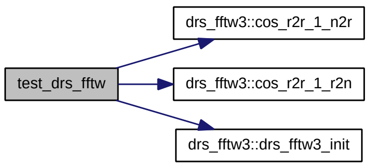
- program **test_drs_fftw**

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

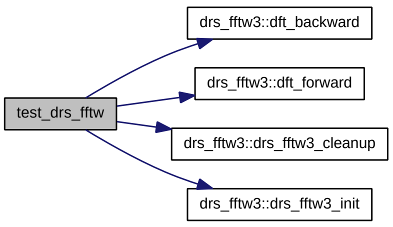
- program **test_drs_fftw**

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** ()

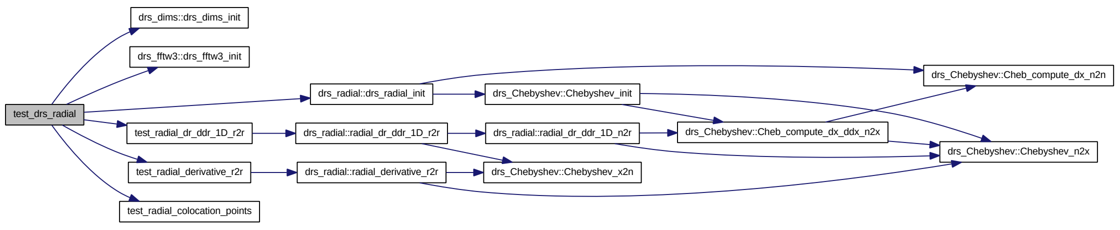
  Tests if the first derivative of r∗∗2 is 2∗r and the second is 2.

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:


drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
test_drs_radial
drs_radial::drs_radial_init
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_n2n
test_radial_dr_ddr_1D_r2r
drs_radial::radial_dr_ddr_1D_r2r
drs_radial::radial_dr_ddr_1D_n2r
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x
test_radial_derivative_r2r
drs_radial::radial_derivative_r2r
drs_Chebyshev::Chebyshev_x2n
test_radial_colocation_points

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().

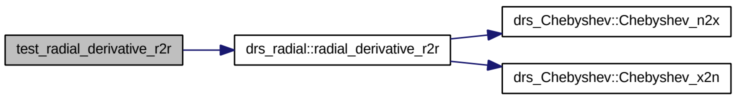
7.31.1.3 logical test_drs_radial::test_radial_derivative_r2r ()

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:


test_radial_derivative_r2r
drs_radial::radial_derivative_r2r
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

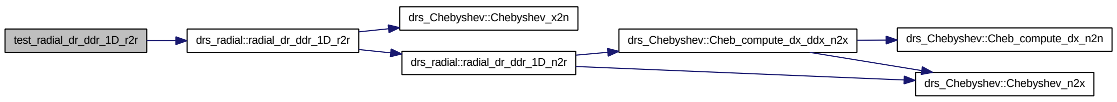
7.31.1.4 logical test_drs_radial::test_radial_dr_ddr_1D_r2r ()

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:


test_radial_dr_ddr_1D_r2r
drs_radial::radial_dr_ddr_1D_r2r
drs_Chebyshev::Chebyshev_x2n
drs_radial::radial_dr_ddr_1D_n2r
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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

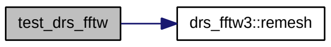
- program **test_drs_fftw**

7.33.1 Function Documentation

7.33.1.1 program test_drs_fftw ()

References drs_fftw3::remesh().

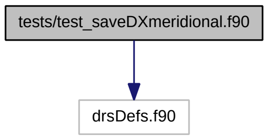
Here is the call graph for this function:



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*)

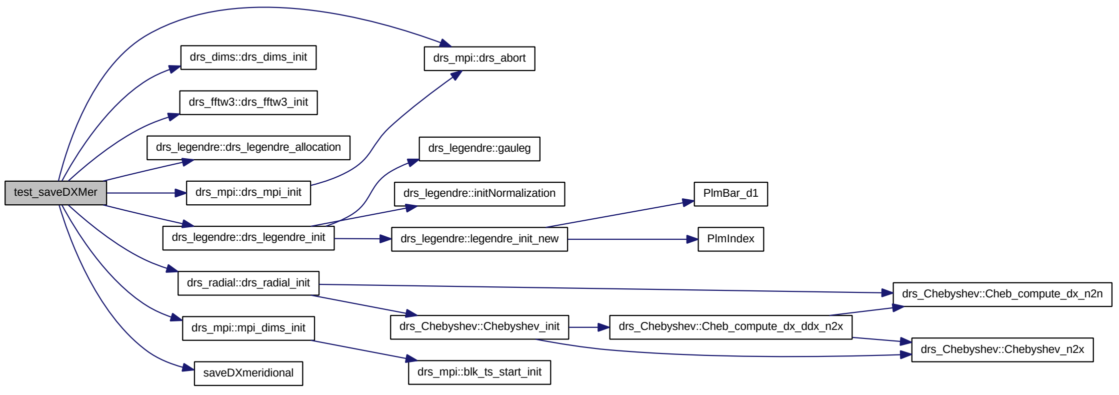
References drs_legendre::costheta, drs_dims::m0, drs_dims::Nt, drs_legendre::pi, and drs_radial::rcoll.

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().

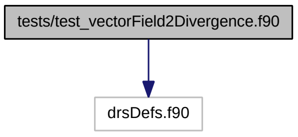
Here is the call graph for this function:


test_saveDXMer
drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_legendre::drs_legendre_allocation
drs_mpi::drs_mpi_init
drs_legendre::drs_legendre_init
drs_radial::drs_radial_init
drs_mpi::mpi_dims_init
saveDXmeridional
drs_mpi::drs_abort
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
drs_Chebyshev::Chebyshev_init
drs_mpi::blk_ts_start_init
PlmBar_d1
PlmIndex
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

7.35 tests/test_vectorField2Divergence.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for test_vectorField2Divergence.f90:



Functions

- program **test_vectorField2Divergence**
- subroutine **init** ()

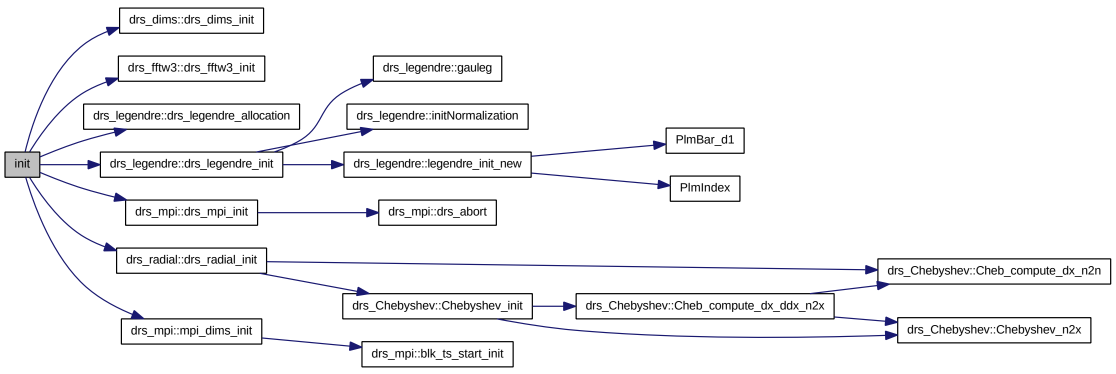
7.35.1 Function Documentation

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().

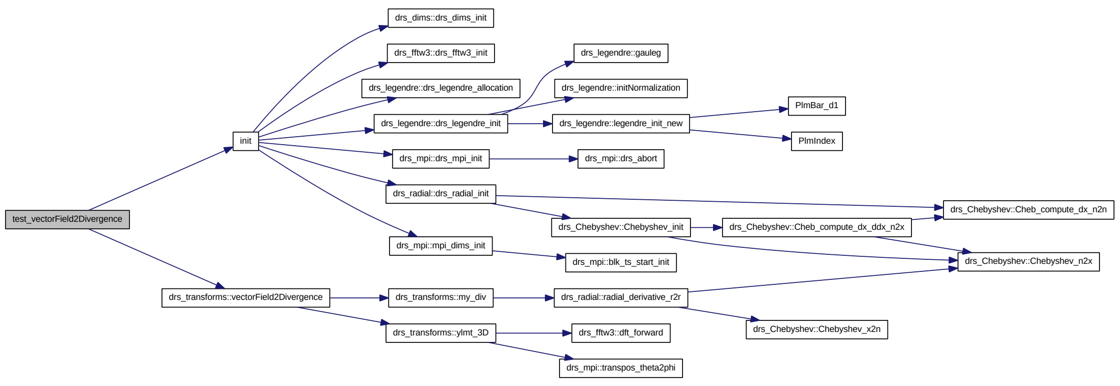
Here is the call graph for this function:


init
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_mpi::mpi_dims_init
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
drs_mpi::drs_abort
drs_Chebyshev::Chebyshev_init
drs_mpi::blk_ts_start_init
PlmBar_d1
PlmIndex
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

7.35.1.2 program test_vectorField2Divergence ()

References init(), drs_dims::Nr, drs_radial::rcoll, and drs_transforms::vectorField2Divergence().

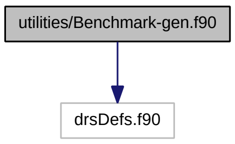
Here is the call graph for this function:


test_vectorField2Divergence
init
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_mpi::mpi_dims_init
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
PlmBar_d1
PlmIndex
drs_mpi::drs_abort
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x
drs_mpi::blk_ts_start_init
drs_transforms::vectorField2Divergence
drs_transforms::my_div
drs_radial::radial_derivative_r2r
drs_Chebyshev::Chebyshev_x2n
drs_transforms::ylmt_3D
drs_fftw3::dft_forward
drs_mpi::transpos_theta2phi

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)

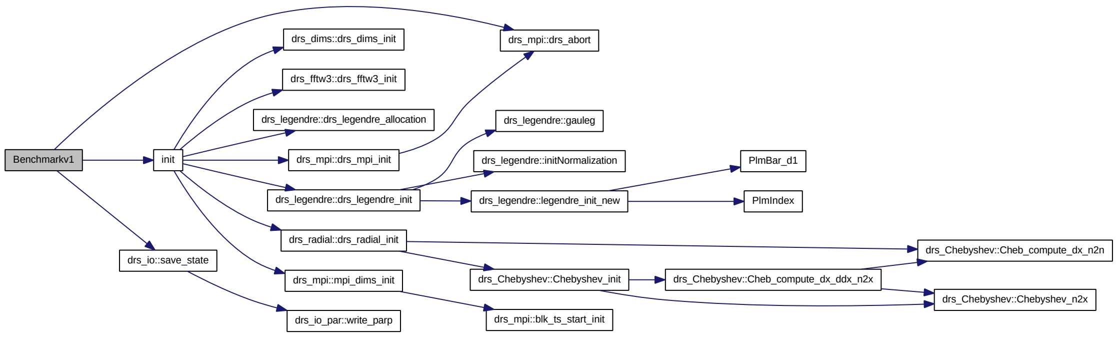
7.36.1 Function Documentation

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.

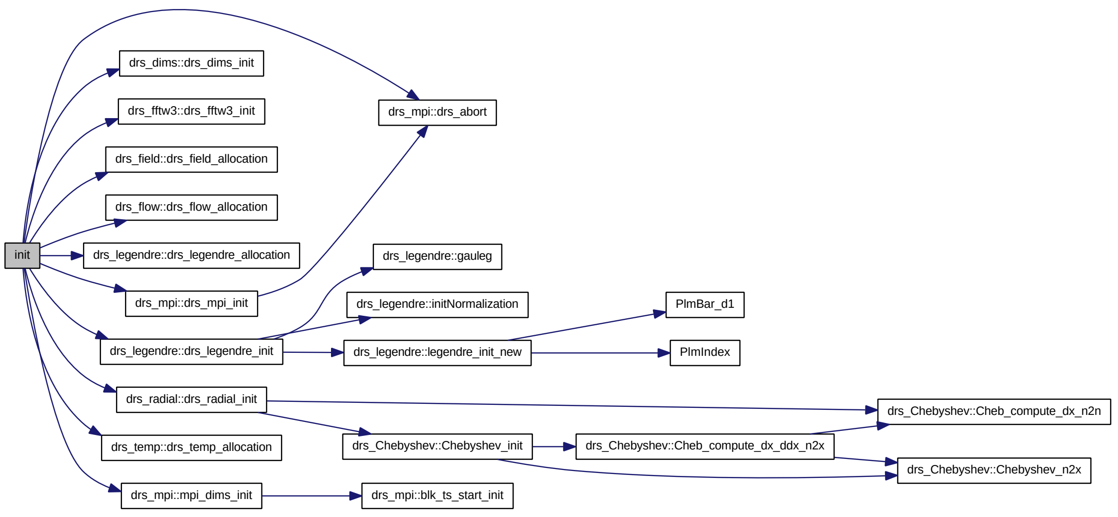
Here is the call graph for this function:


Benchmarkv1
init
drs_io::save_state
drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_legendre::drs_legendre_allocation
drs_mpi::drs_mpi_init
drs_legendre::drs_legendre_init
drs_radial::drs_radial_init
drs_mpi::mpi_dims_init
drs_io_par::write_parp
drs_mpi::drs_abort
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
drs_Chebyshev::Chebyshev_init
drs_mpi::blk_ts_start_init
PlmBar_d1
PlmIndex
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x

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_fftw3::drs_fftw3_-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.

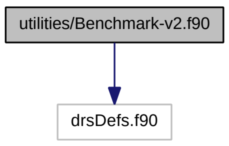
Here is the call graph for this function:


init
drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_field::drs_field_allocation
drs_flow::drs_flow_allocation
drs_legendre::drs_legendre_allocation
drs_mpi::drs_mpi_init
drs_legendre::drs_legendre_init
drs_radial::drs_radial_init
drs_temp::drs_temp_allocation
drs_mpi::mpi_dims_init
drs_mpi::drs_abort
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
drs_Chebyshev::Chebyshev_init
drs_mpi::blk_ts_start_init
PlmBar_d1
PlmIndex
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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)

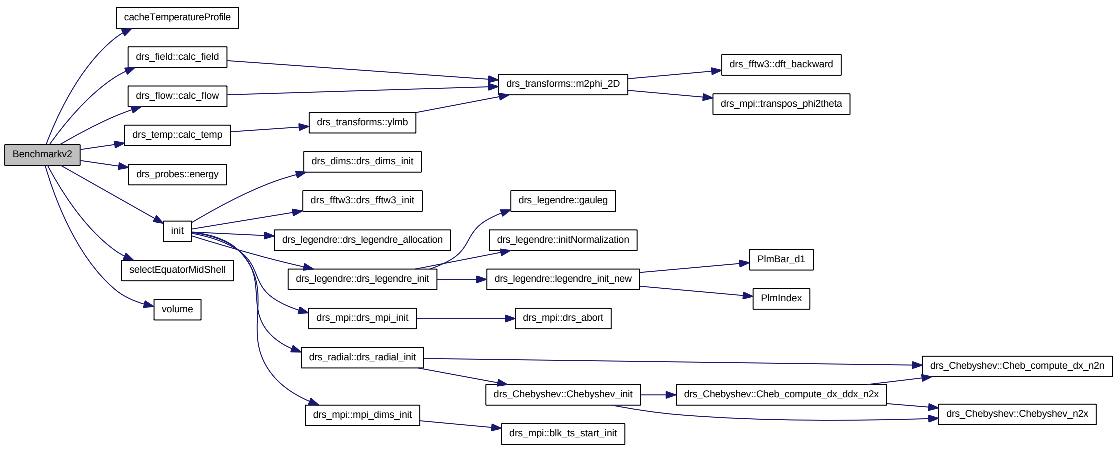
7.37.1 Function Documentation

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:


cacheTemperatureProfile
drs_field::calc_field
drs_flow::calc_flow
drs_temp::calc_temp
Benchmarkv2
drs_probes::energy
init
selectEquatorMidShell
volume
drs_transforms::m2phi_2D
drs_fftw3::dft_backward
drs_mpi::transpos_phi2theta
drs_transforms::ylmb
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_mpi::mpi_dims_init
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
drs_mpi::drs_abort
drs_Chebyshev::Chebyshev_init
drs_mpi::blk_ts_start_init
PlmBar_d1
PlmIndex
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

7.37.1.2 subroutine Benchmarkv2::cacheTemperatureProfile (integer,intent(in) *rcut*, double precision,intent(out) *T*)

Cache the temperature from the profile.

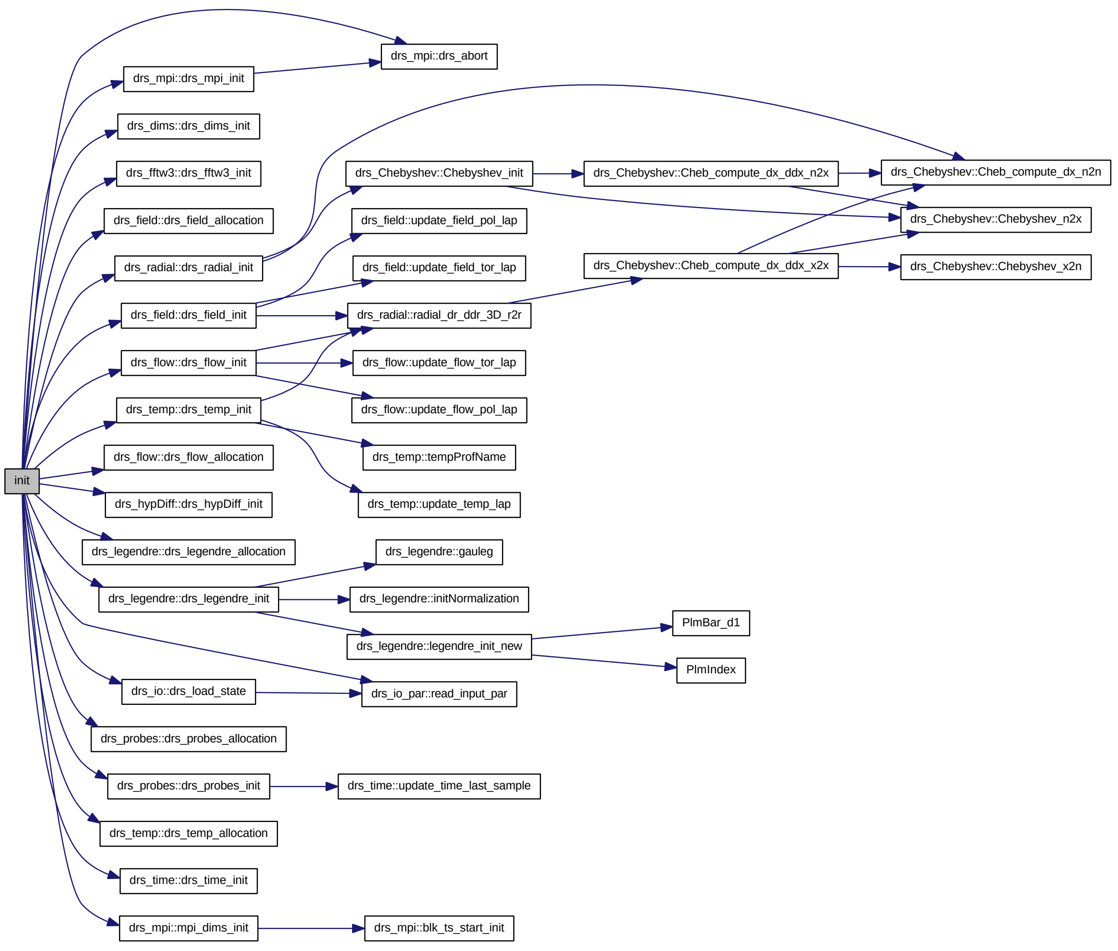
References drs_radial::drcoll, drs_radial::rcoll, and drs_temp::temp_profile.

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:


init
drs_mpi::drs_mpi_init
drs_mpi::drs_abort
drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_field::drs_field_allocation
drs_field::update_field_pol_lap
drs_Chebyshev::Chebyshev_n2x
drs_radial::drs_radial_init
drs_field::update_field_tor_lap
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Chebyshev_x2n
drs_field::drs_field_init
drs_radial::radial_dr_ddr_3D_r2r
drs_flow::drs_flow_init
drs_flow::update_flow_tor_lap
drs_temp::drs_temp_init
drs_flow::update_flow_pol_lap
drs_flow::drs_flow_allocation
drs_temp::tempProfName
drs_hypDiff::drs_hypDiff_init
drs_temp::update_temp_lap
drs_legendre::drs_legendre_allocation
drs_legendre::gauleg
drs_legendre::drs_legendre_init
drs_legendre::initNormalization
PlmBar_d1
drs_legendre::legendre_init_new
PlmIndex
drs_io::drs_load_state
drs_io_par::read_input_par
drs_probes::drs_probes_allocation
drs_probes::drs_probes_init
drs_time::update_time_last_sample
drs_temp::drs_temp_allocation
drs_time::drs_time_init
drs_mpi::mpi_dims_init
drs_mpi::blk_ts_start_init

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*)

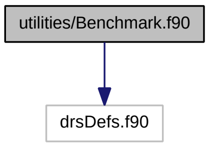
References drs_legendre::pi.

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)

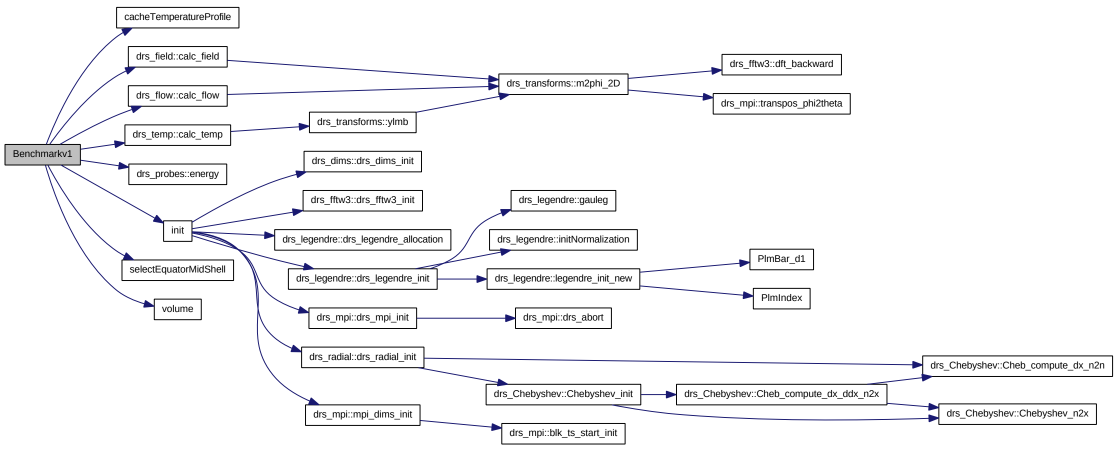
7.38.1 Function Documentation

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:


Benchmarkv1
cacheTemperatureProfile
drs_field::calc_field
drs_flow::calc_flow
drs_temp::calc_temp
drs_probes::energy
init
selectEquatorMidShell
volume
drs_transforms::m2phi_2D
drs_fftw3::dft_backward
drs_mpi::transpos_phi2theta
drs_transforms::ylmb
drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_legendre::drs_legendre_allocation
drs_legendre::drs_legendre_init
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
PlmBar_d1
PlmIndex
drs_mpi::drs_mpi_init
drs_mpi::drs_abort
drs_radial::drs_radial_init
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x
drs_mpi::mpi_dims_init
drs_mpi::blk_ts_start_init

7.38.1.2 subroutine Benchmarkv1::cacheTemperatureProfile (integer,intent(in) *rcut*, double precision,intent(out) *T*)

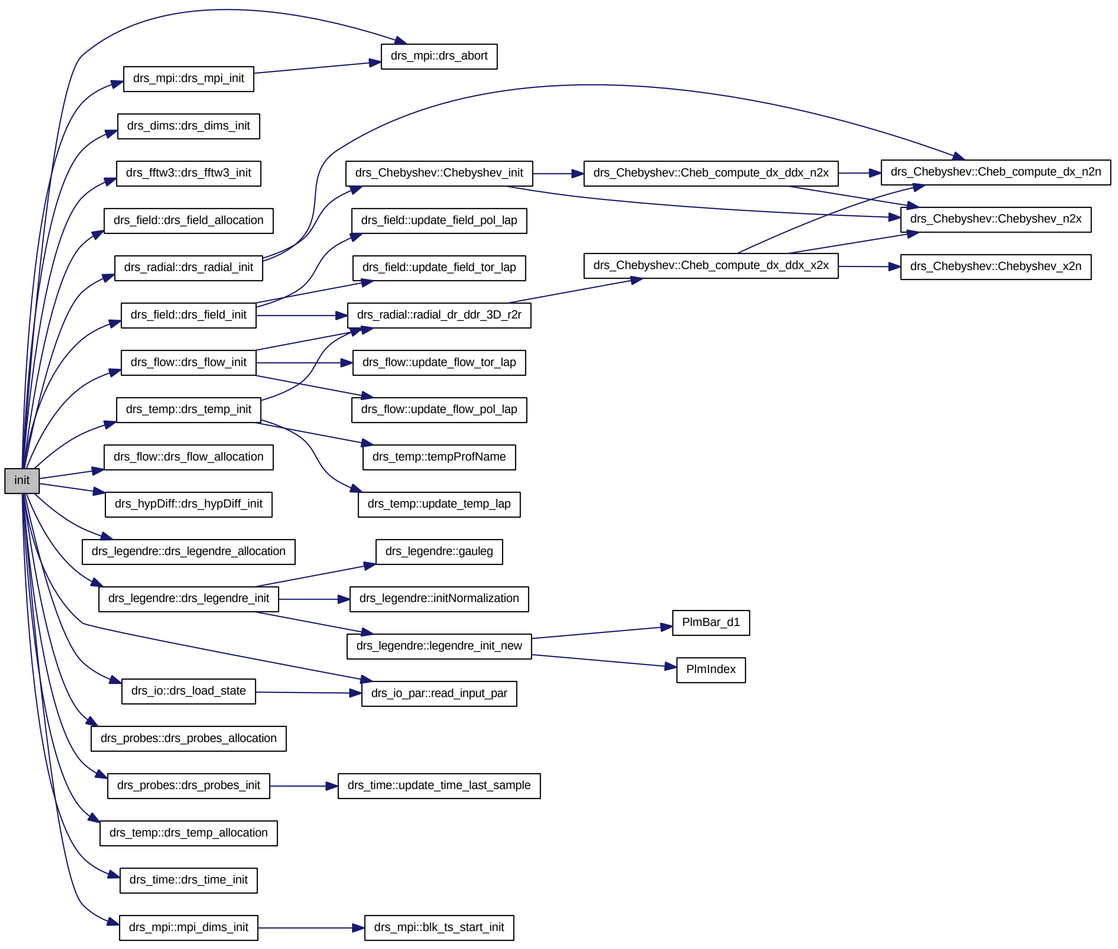
Cache the temperature from the profile.

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:


init
drs_mpi::drs_abort
drs_mpi::drs_mpi_init
drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_field::drs_field_allocation
drs_radial::drs_radial_init
drs_field::drs_field_init
drs_flow::drs_flow_init
drs_temp::drs_temp_init
drs_flow::drs_flow_allocation
drs_hypDiff::drs_hypDiff_init
drs_legendre::drs_legendre_allocation
drs_legendre::drs_legendre_init
drs_io::drs_load_state
drs_probes::drs_probes_allocation
drs_probes::drs_probes_init
drs_temp::drs_temp_allocation
drs_time::drs_time_init
drs_mpi::mpi_dims_init
drs_Chebyshev::Chebyshev_init
drs_field::update_field_pol_lap
drs_field::update_field_tor_lap
drs_radial::radial_dr_ddr_3D_r2r
drs_flow::update_flow_tor_lap
drs_flow::update_flow_pol_lap
drs_temp::tempProfName
drs_temp::update_temp_lap
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
drs_io_par::read_input_par
drs_time::update_time_last_sample
drs_mpi::blk_ts_start_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n
PlmBar_d1
PlmIndex

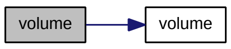
7.38.1.4 subroutine Benchmarkv1::selectEquatorMidShell (double precision,dimension(0:(blk_-t_size(mpi_rank),intent(in) *field*, double precision,dimension(np),intent(out) *line*, integer,intent(out) *rcut*)

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().

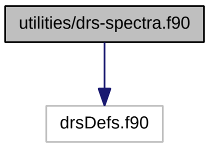
Here is the call graph for this function:



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 speciffic state.

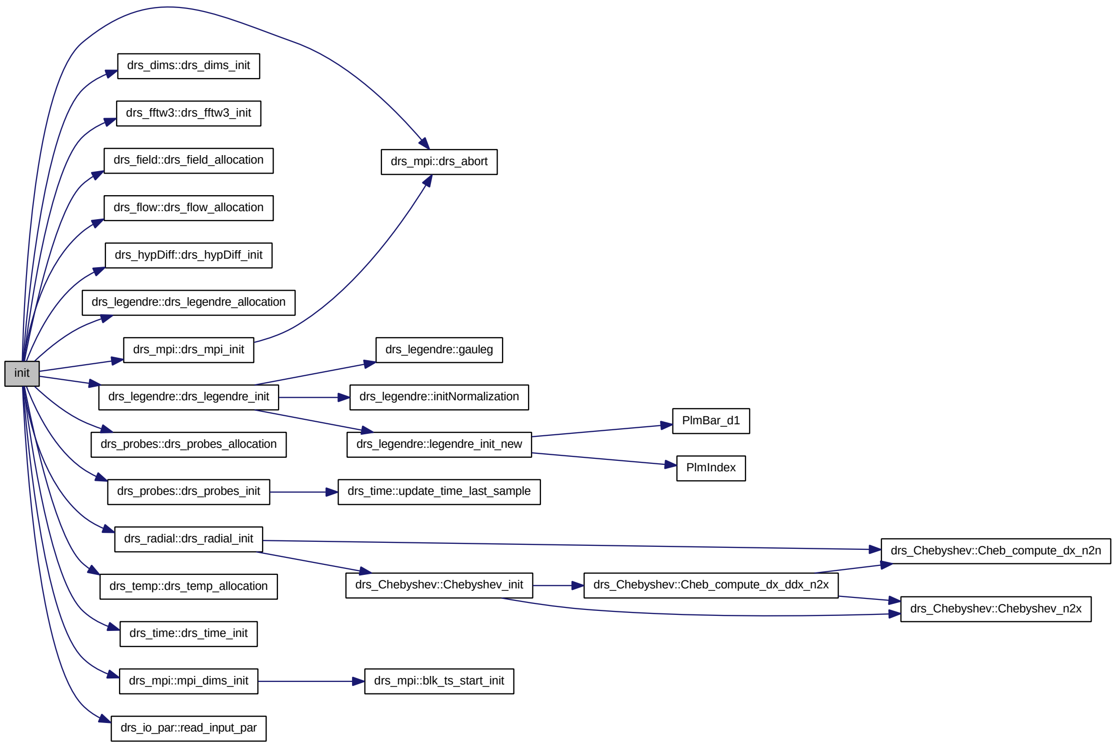
- subroutine **init** (error)

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:

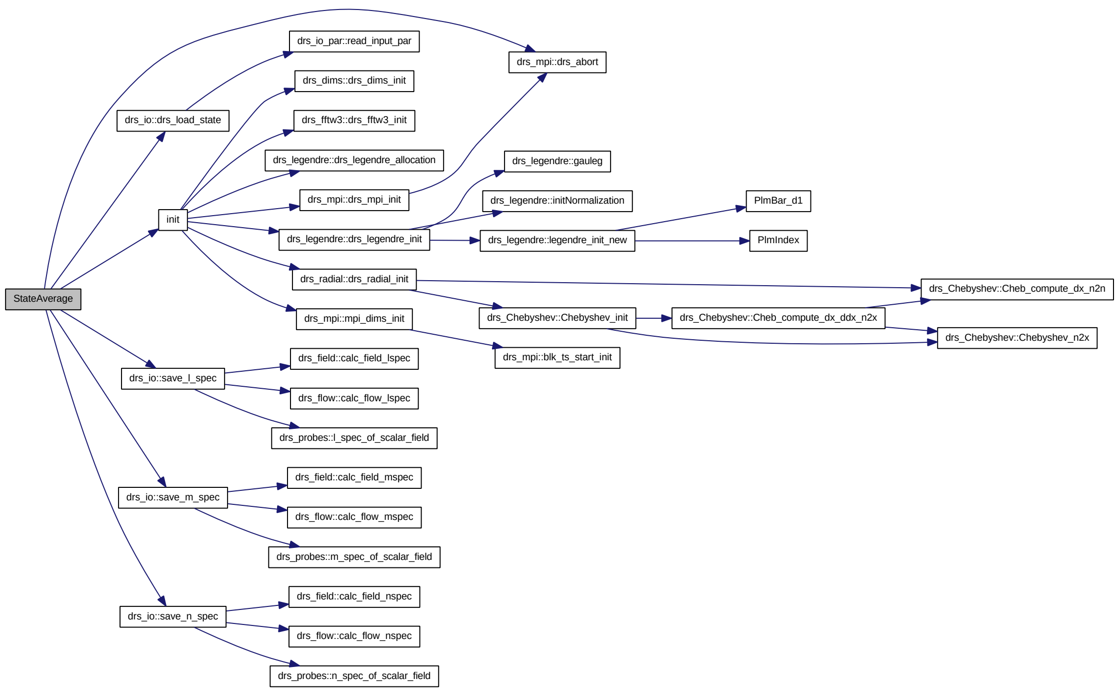

init
drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_field::drs_field_allocation
drs_flow::drs_flow_allocation
drs_hypDiff::drs_hypDiff_init
drs_legendre::drs_legendre_allocation
drs_mpi::drs_mpi_init
drs_legendre::drs_legendre_init
drs_probes::drs_probes_allocation
drs_probes::drs_probes_init
drs_radial::drs_radial_init
drs_temp::drs_temp_allocation
drs_time::drs_time_init
drs_mpi::mpi_dims_init
drs_io_par::read_input_par
drs_mpi::drs_abort
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
drs_time::update_time_last_sample
drs_Chebyshev::Chebyshev_init
drs_mpi::blk_ts_start_init
PlmBar_d1
PlmIndex
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

7.39.1.2 program StateAverage ()

Computes the spectra of all quantities for a speciffic 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().

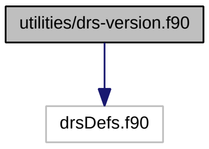
Here is the call graph for this function:


StateAverage
drs_io::drs_load_state
init
drs_io_par::read_input_par
drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_legendre::drs_legendre_allocation
drs_mpi::drs_mpi_init
drs_legendre::drs_legendre_init
drs_radial::drs_radial_init
drs_mpi::mpi_dims_init
drs_mpi::drs_abort
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
PlmBar_d1
PlmIndex
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x
drs_mpi::blk_ts_start_init
drs_io::save_l_spec
drs_field::calc_field_lspec
drs_flow::calc_flow_lspec
drs_probes::l_spec_of_scalar_field
drs_io::save_m_spec
drs_field::calc_field_mspec
drs_flow::calc_flow_mspec
drs_probes::m_spec_of_scalar_field
drs_io::save_n_spec
drs_field::calc_field_nspec
drs_flow::calc_flow_nspec
drs_probes::n_spec_of_scalar_field

7.40 utilities/drs-version.f90 File Reference

```
#include "drsDefs.f90"
```

Include dependency graph for drs-version.f90:



Functions

- program **drsVersion**

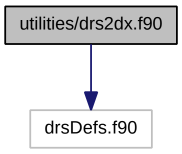
7.40.1 Function Documentation

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** ()

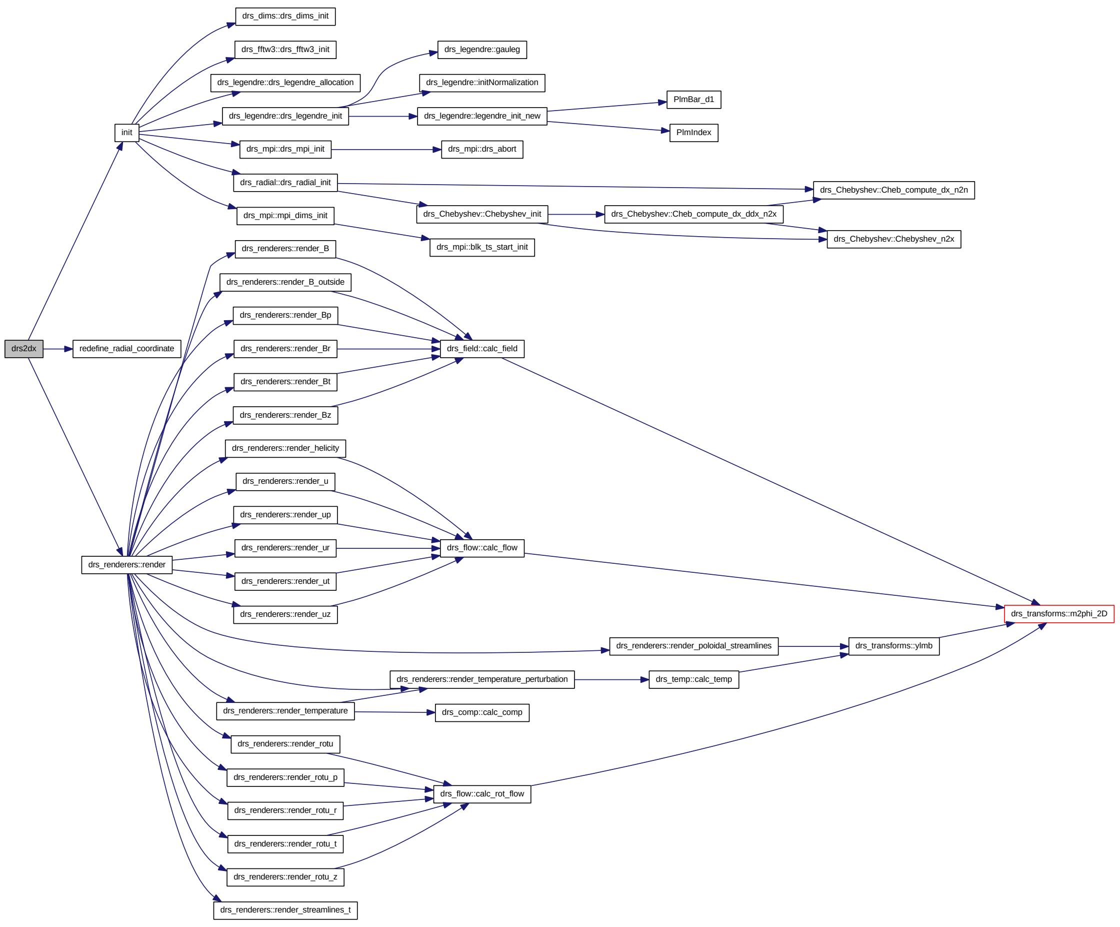
7.41.1 Function Documentation

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.

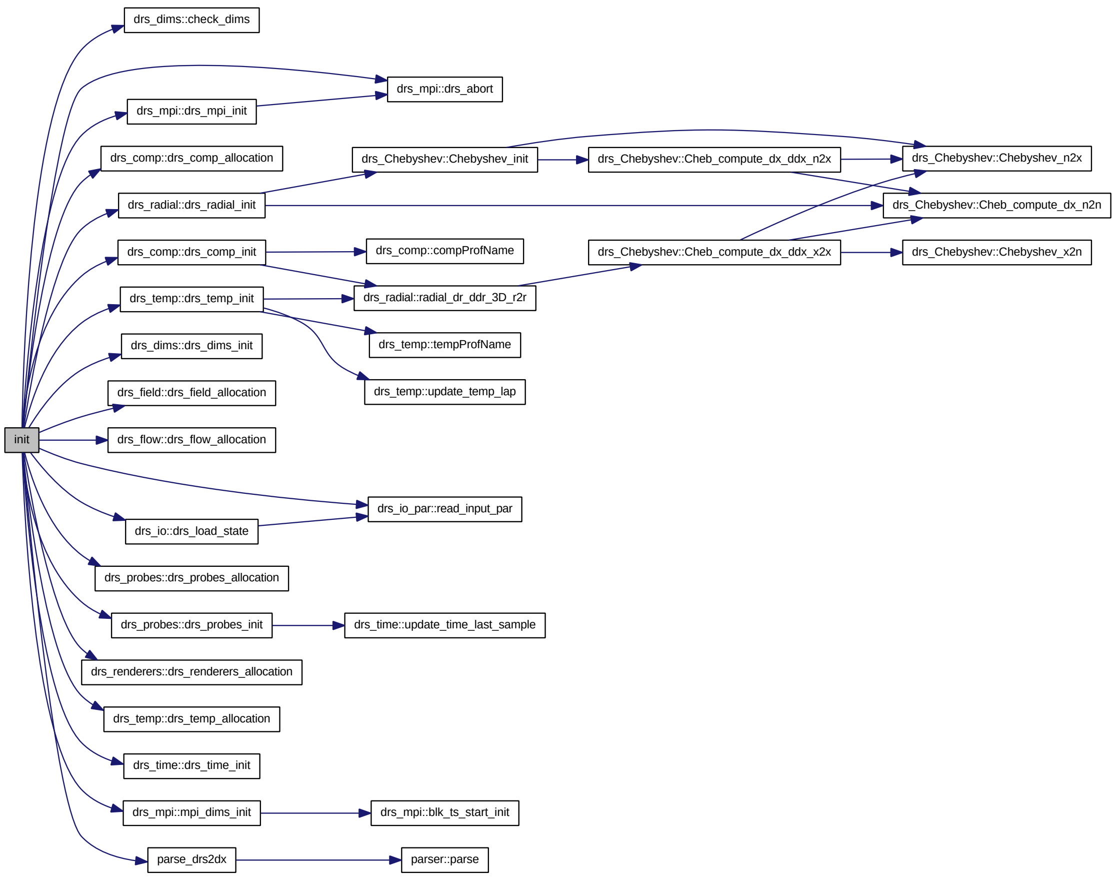
Here is the call graph for this function:


drs2dx
init
redefine_radial_coordinate
drs_renderers::render
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_mpi::mpi_dims_init
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
PlmBar_d1
PlmIndex
drs_mpi::drs_abort
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x
drs_mpi::blk_ts_start_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_field::calc_field
drs_renderers::render_helicity
drs_renderers::render_u
drs_renderers::render_up
drs_renderers::render_ur
drs_renderers::render_ut
drs_renderers::render_uz
drs_flow::calc_flow
drs_transforms::m2phi_2D
drs_renderers::render_poloidal_streamlines
drs_transforms::ylmb
drs_renderers::render_temperature_perturbation
drs_temp::calc_temp
drs_renderers::render_temperature
drs_comp::calc_comp
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_flow::calc_rot_flow
drs_renderers::render_streamlines_t

7.41.1.2 subroutine drs2dx::init (integer,intent(inout) *error*)

References drs_mpi::blk_t_size, drs_dims::check_dims(), drs_io::deflate, drs_time::drift, drs_io_-par::drifti, 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_drs2dx(), 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:

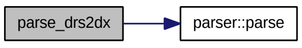

drs_dims::check_dims
drs_mpi::drs_abort
drs_mpi::drs_mpi_init
drs_comp::drs_comp_allocation
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x
drs_radial::drs_radial_init
drs_Chebyshev::Cheb_compute_dx_n2n
drs_comp::drs_comp_init
drs_comp::compProfName
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Chebyshev_x2n
drs_temp::drs_temp_init
drs_radial::radial_dr_ddr_3D_r2r
drs_dims::drs_dims_init
drs_temp::tempProfName
drs_field::drs_field_allocation
drs_temp::update_temp_lap
init
drs_flow::drs_flow_allocation
drs_io_par::read_input_par
drs_io::drs_load_state
drs_probes::drs_probes_allocation
drs_probes::drs_probes_init
drs_time::update_time_last_sample
drs_renderers::drs_renderers_allocation
drs_temp::drs_temp_allocation
drs_time::drs_time_init
drs_mpi::mpi_dims_init
drs_mpi::blk_ts_start_init
parse_drs2dx
parser::parse

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 ()

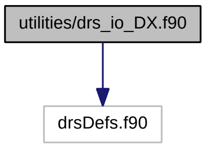
References drs_radial::drcoll, drs_dims::Nr, drs_radial::rcoll, and drs_radial::rcoll2.

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 rendeer of the field.
- subroutine **drs_io_DX::saveDXvolume_v2** (field, filename)
  Writes a volume rendeer of the field.
- subroutine **drs_io_DX::saveDXvolume3DVec** (XX, YY, ZZ, filename)
  Writes a volume rendeer 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

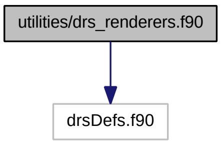
- double precision **drs_io_DX::where_to_cut** = 0.0d0
- integer **drs_io_DX::cut_type**

  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 ploted | |> 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_temprature_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**

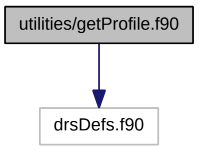
7.43.1 Define Documentation

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.

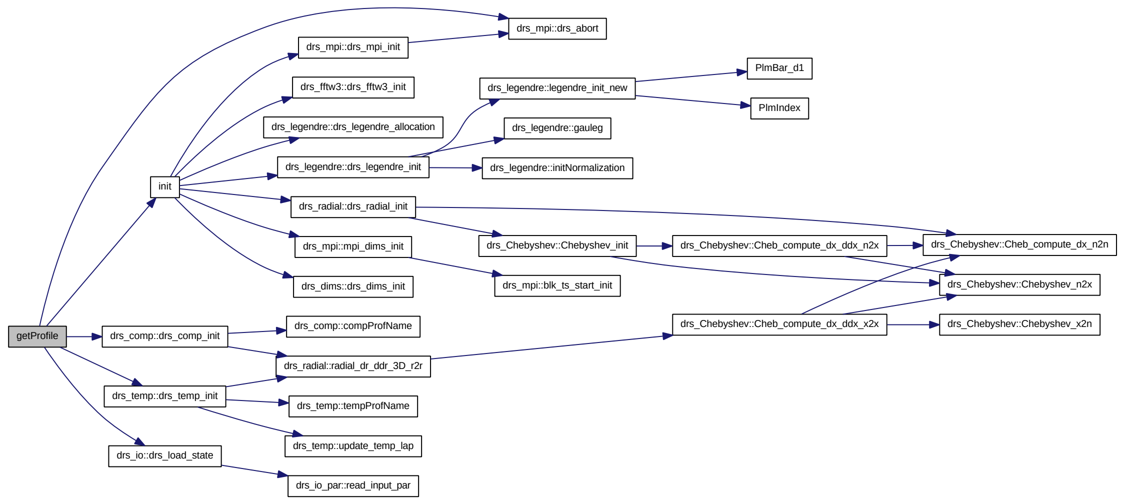
7.44.1 Function Documentation

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.

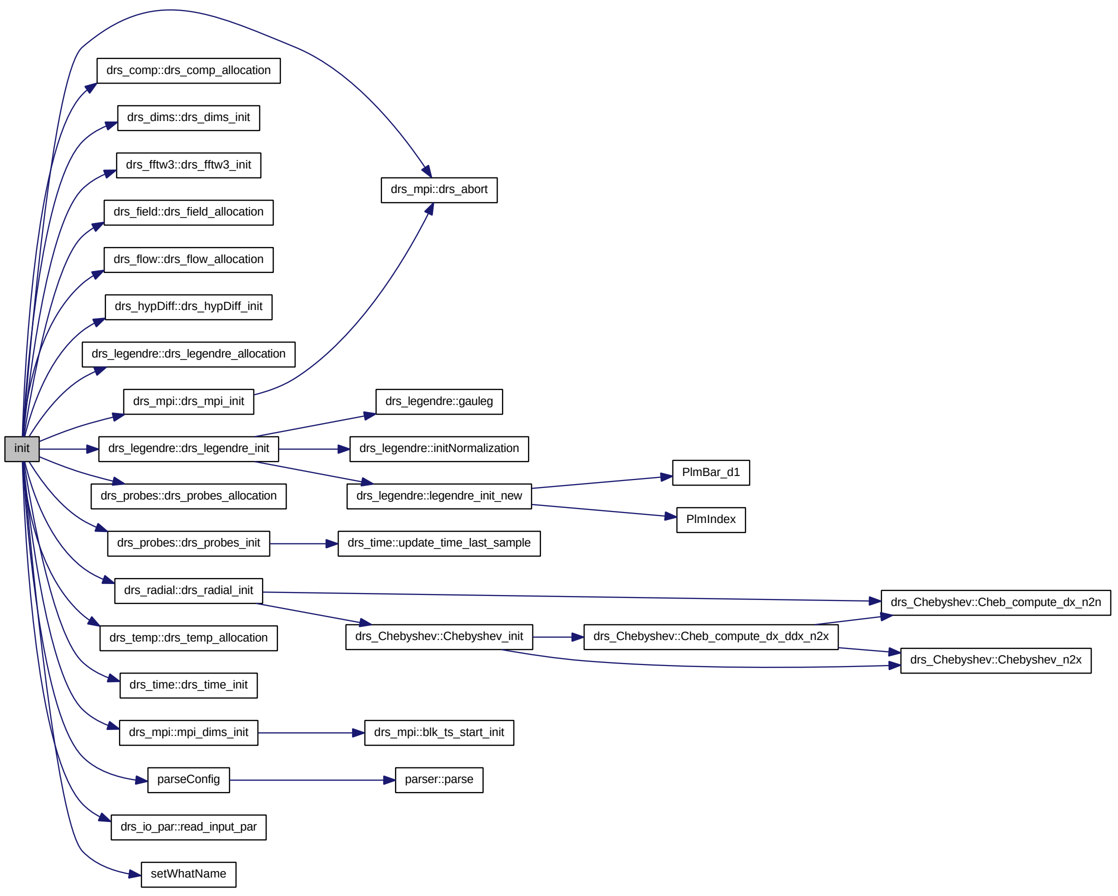
Here is the call graph for this function:


drs_mpi::drs_abort
drs_mpi::drs_mpi_init
drs_fftw3::drs_fftw3_init
drs_legendre::drs_legendre_allocation
drs_legendre::drs_legendre_init
init
drs_radial::drs_radial_init
drs_mpi::mpi_dims_init
drs_dims::drs_dims_init
getProfile
drs_comp::drs_comp_init
drs_comp::compProfName
drs_radial::radial_dr_ddr_3D_r2r
drs_temp::drs_temp_init
drs_temp::tempProfName
drs_temp::update_temp_lap
drs_io::drs_load_state
drs_io_par::read_input_par
drs_legendre::legendre_init_new
PlmBar_d1
PlmIndex
drs_legendre::gauleg
drs_legendre::initNormalization
drs_Chebyshev::Chebyshev_init
drs_mpi::blk_ts_start_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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:


init
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_mpi::drs_mpi_init
drs_legendre::drs_legendre_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_mpi::mpi_dims_init
parseConfig
drs_io_par::read_input_par
setWhatName
drs_mpi::drs_abort
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
PlmBar_d1
PlmIndex
drs_time::update_time_last_sample
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x
drs_mpi::blk_ts_start_init
parser::parse

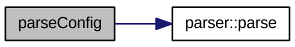
7.44.1.3 subroutine getProfile::parseConfig (integer *error*)

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 ()

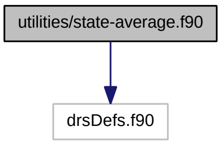
Sets a whuman readable name for what is being computed.

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.

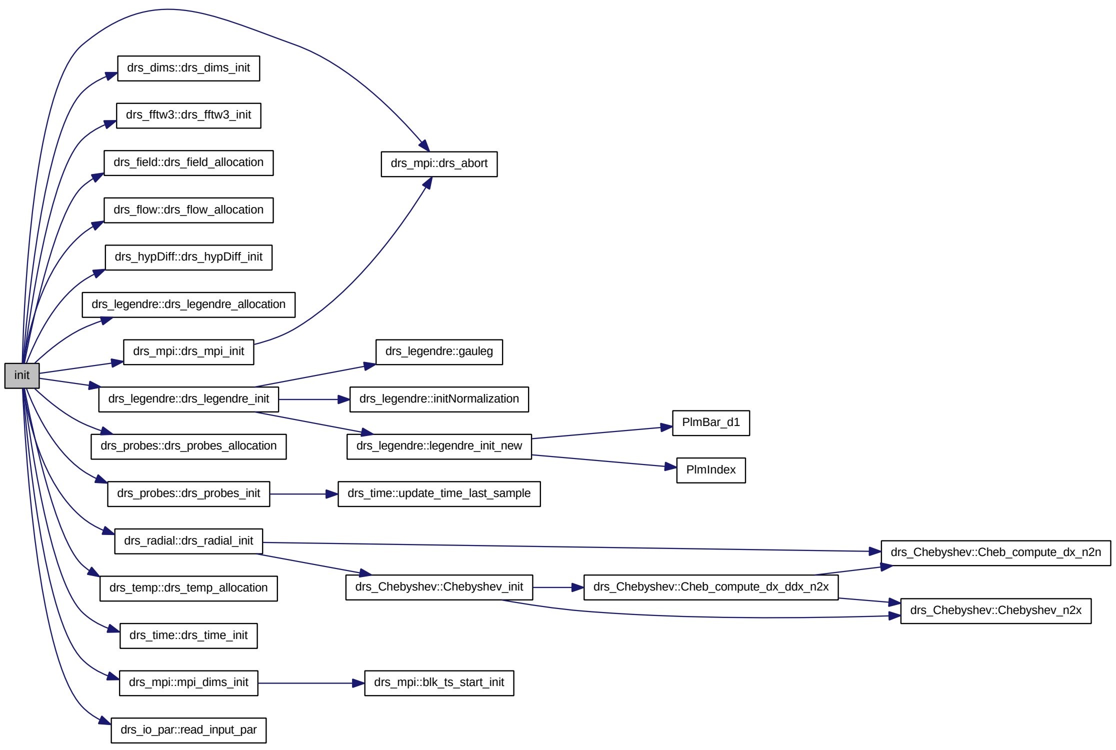
- subroutine **init** (error)

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:

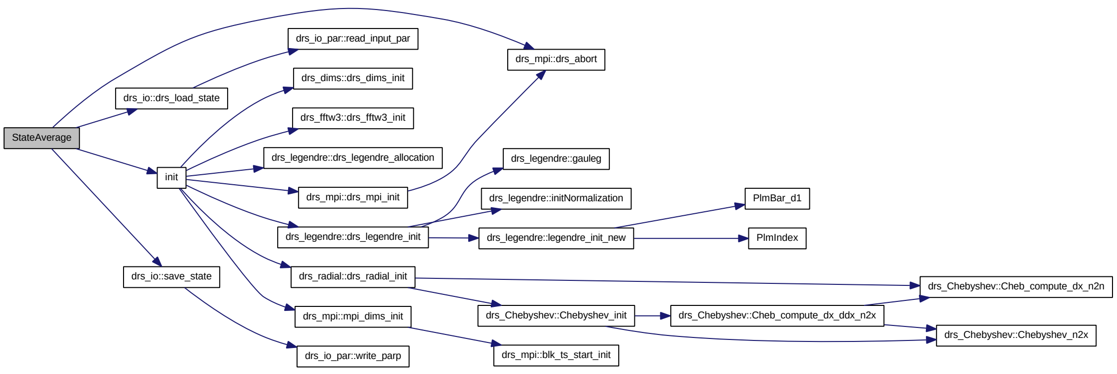

drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_field::drs_field_allocation
drs_mpi::drs_abort
drs_flow::drs_flow_allocation
drs_hypDiff::drs_hypDiff_init
drs_legendre::drs_legendre_allocation
drs_mpi::drs_mpi_init
drs_legendre::gauleg
init
drs_legendre::drs_legendre_init
drs_legendre::initNormalization
PlmBar_d1
drs_probes::drs_probes_allocation
drs_legendre::legendre_init_new
PlmIndex
drs_probes::drs_probes_init
drs_time::update_time_last_sample
drs_radial::drs_radial_init
drs_Chebyshev::Cheb_compute_dx_n2n
drs_temp::drs_temp_allocation
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x
drs_time::drs_time_init
drs_mpi::mpi_dims_init
drs_mpi::blk_ts_start_init
drs_io_par::read_input_par

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.

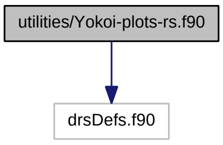
Here is the call graph for this function:


drs_io_par::read_input_par
drs_mpi::drs_abort
drs_dims::drs_dims_init
drs_io::drs_load_state
drs_fftw3::drs_fftw3_init
StateAverage
drs_legendre::drs_legendre_allocation
drs_legendre::gauleg
init
drs_mpi::drs_mpi_init
drs_legendre::initNormalization
PlmBar_d1
drs_legendre::drs_legendre_init
drs_legendre::legendre_init_new
PlmIndex
drs_io::save_state
drs_radial::drs_radial_init
drs_Chebyshev::Cheb_compute_dx_n2n
drs_mpi::mpi_dims_init
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x
drs_io_par::write_parp
drs_mpi::blk_ts_start_init

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.

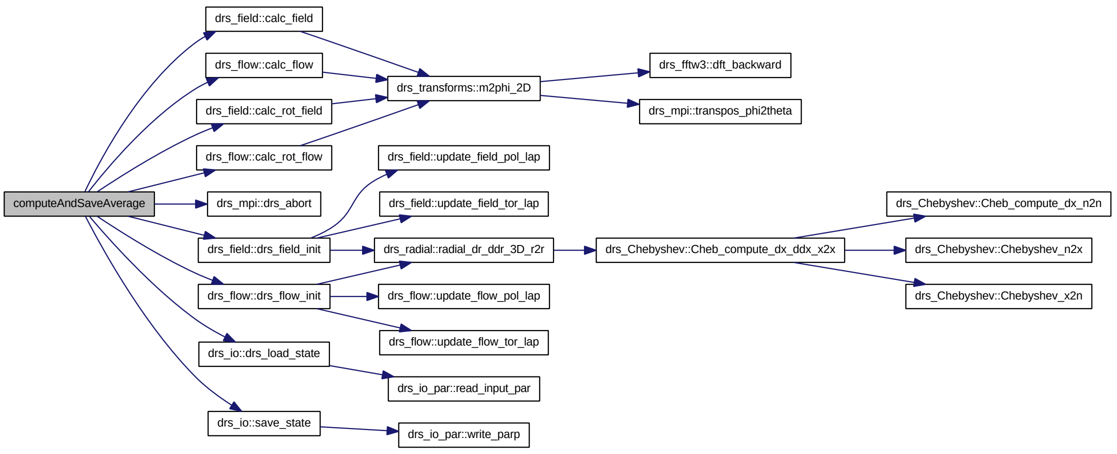
Referenced by StateAverage(), and YokoiPlots().

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:


drs_field::calc_field
drs_flow::calc_flow
drs_field::calc_rot_field
drs_flow::calc_rot_flow
computeAndSaveAverage
drs_mpi::drs_abort
drs_field::drs_field_init
drs_flow::drs_flow_init
drs_io::drs_load_state
drs_io::save_state
drs_transforms::m2phi_2D
drs_field::update_field_pol_lap
drs_field::update_field_tor_lap
drs_radial::radial_dr_ddr_3D_r2r
drs_flow::update_flow_pol_lap
drs_flow::update_flow_tor_lap
drs_io_par::read_input_par
drs_io_par::write_parp
drs_fftw3::dft_backward
drs_mpi::transpos_phi2theta
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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*)

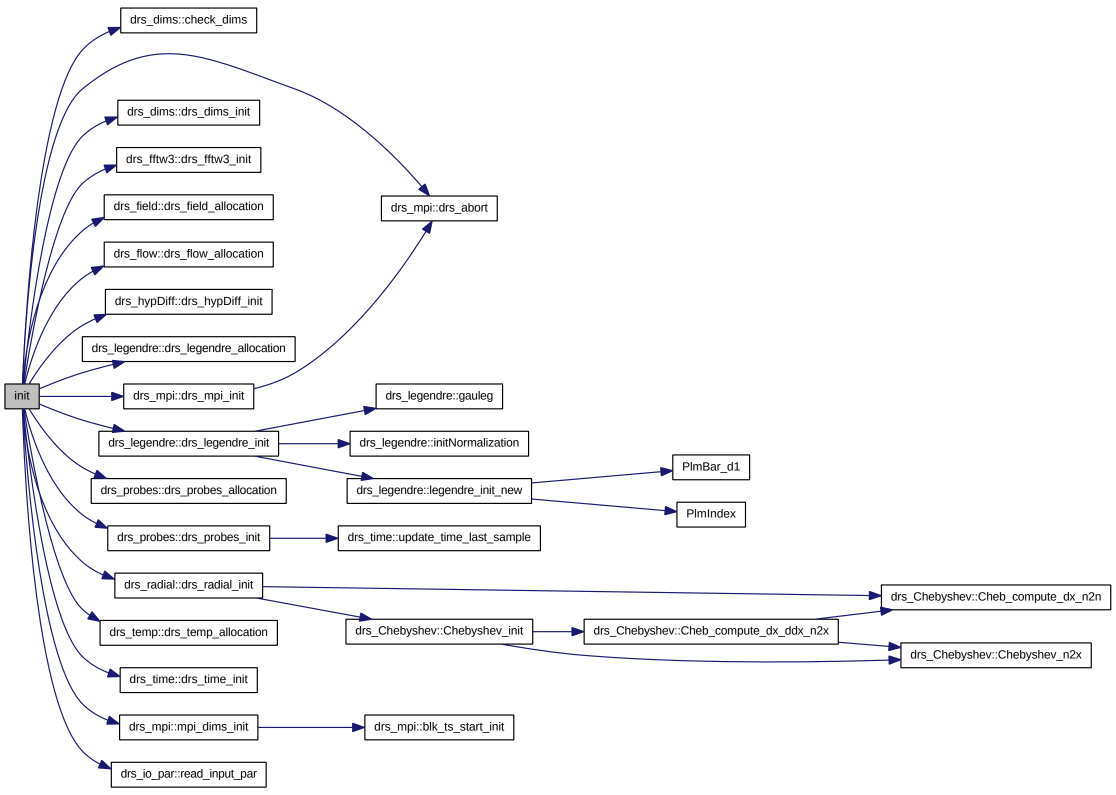
References drs_dims::Nt.

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:


init
drs_dims::check_dims
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_mpi::drs_mpi_init
drs_legendre::drs_legendre_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_mpi::mpi_dims_init
drs_io_par::read_input_par
drs_mpi::drs_abort
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
PlmBar_d1
PlmIndex
drs_time::update_time_last_sample
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_init
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x
drs_mpi::blk_ts_start_init

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.

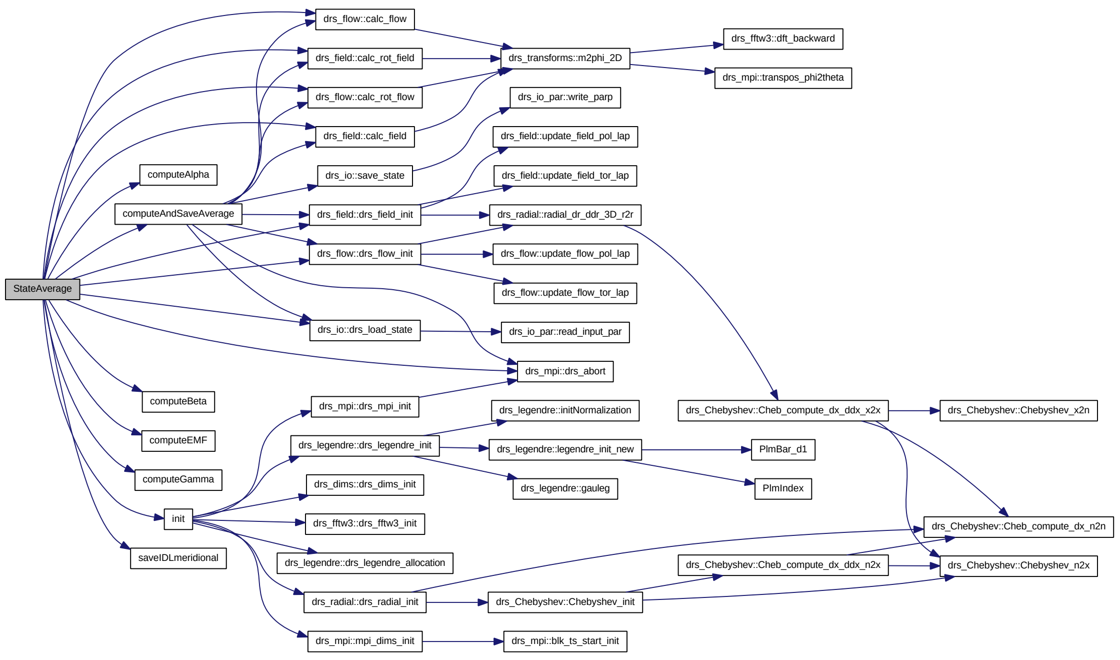
Referenced by StateAverage().

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 saveIDLmerid-ional().

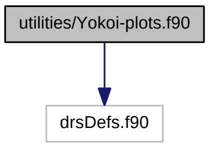
Here is the call graph for this function:


StateAverage
computeAlpha
computeAndSaveAverage
computeBeta
computeEMF
computeGamma
init
saveIDLmeridional
drs_flow::calc_flow
drs_field::calc_rot_field
drs_flow::calc_rot_flow
drs_field::calc_field
drs_io::save_state
drs_field::drs_field_init
drs_flow::drs_flow_init
drs_io::drs_load_state
drs_mpi::drs_mpi_init
drs_legendre::drs_legendre_init
drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_legendre::drs_legendre_allocation
drs_radial::drs_radial_init
drs_mpi::mpi_dims_init
drs_transforms::m2phi_2D
drs_io_par::write_parp
drs_field::update_field_pol_lap
drs_field::update_field_tor_lap
drs_radial::radial_dr_ddr_3D_r2r
drs_flow::update_flow_pol_lap
drs_flow::update_flow_tor_lap
drs_io_par::read_input_par
drs_mpi::drs_abort
drs_legendre::initNormalization
drs_legendre::legendre_init_new
drs_legendre::gauleg
drs_Chebyshev::Chebyshev_init
drs_mpi::blk_ts_start_init
drs_fftw3::dft_backward
drs_mpi::transpos_phi2theta
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
PlmBar_d1
PlmIndex
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_x2n
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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

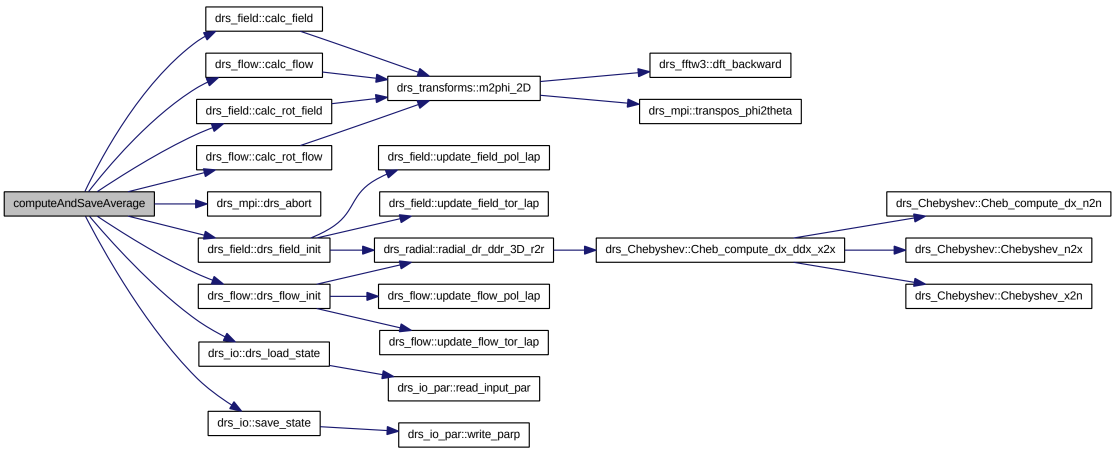
7.47.1.1 subroutine YokoiPlots::computeAlpha (double precision,dimension(0:(blk_t_size(mpi_-rank),intent(inout) *alpha*)

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:


computeAndSaveAverage
drs_field::calc_field
drs_flow::calc_flow
drs_field::calc_rot_field
drs_flow::calc_rot_flow
drs_mpi::drs_abort
drs_field::drs_field_init
drs_flow::drs_flow_init
drs_io::drs_load_state
drs_io::save_state
drs_transforms::m2phi_2D
drs_fftw3::dft_backward
drs_mpi::transpos_phi2theta
drs_field::update_field_pol_lap
drs_field::update_field_tor_lap
drs_radial::radial_dr_ddr_3D_r2r
drs_flow::update_flow_pol_lap
drs_flow::update_flow_tor_lap
drs_io_par::read_input_par
drs_io_par::write_parp
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x
drs_Chebyshev::Chebyshev_x2n

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.

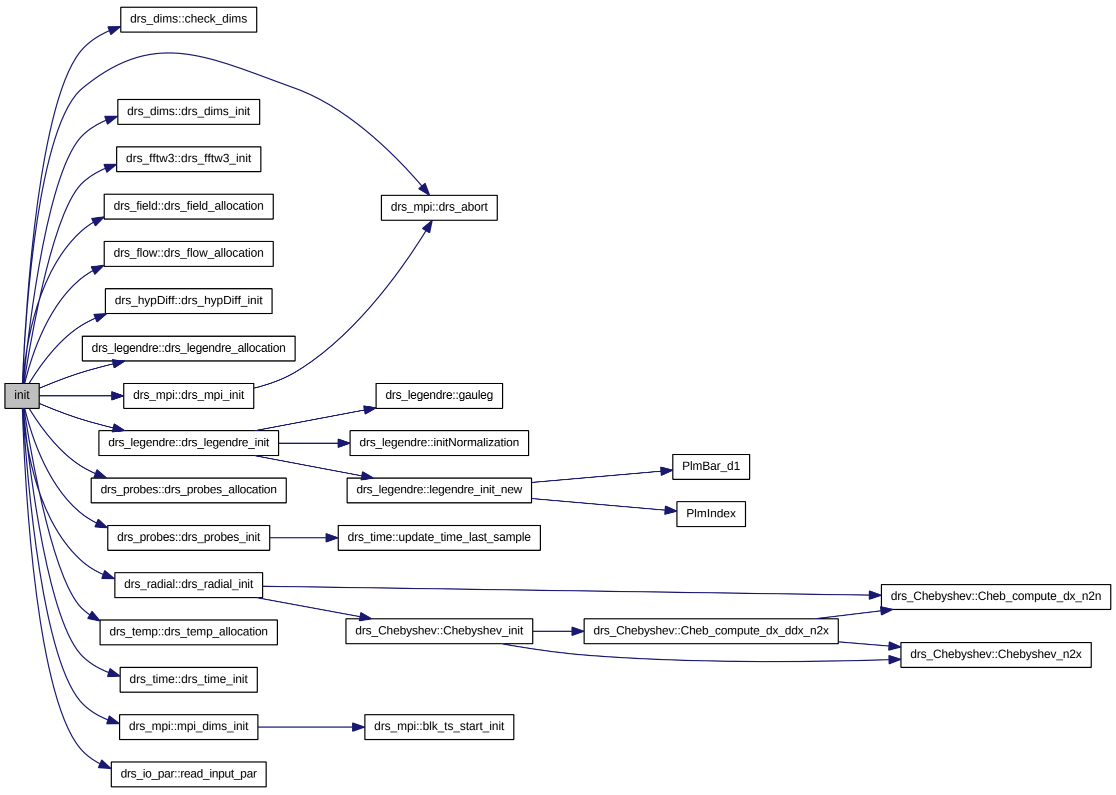
7.47.1.5 subroutine YokoiPlots::computeGamma (double precision,dimension(0:(blk_t_size(mpi_-rank),intent(inout) *gamma*)

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_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:


init
drs_dims::check_dims
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_mpi::drs_mpi_init
drs_legendre::drs_legendre_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_mpi::mpi_dims_init
drs_io_par::read_input_par
drs_mpi::drs_abort
drs_legendre::gauleg
drs_legendre::initNormalization
drs_legendre::legendre_init_new
drs_time::update_time_last_sample
drs_Chebyshev::Chebyshev_init
drs_mpi::blk_ts_start_init
PlmBar_d1
PlmIndex
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Chebyshev_n2x

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*)

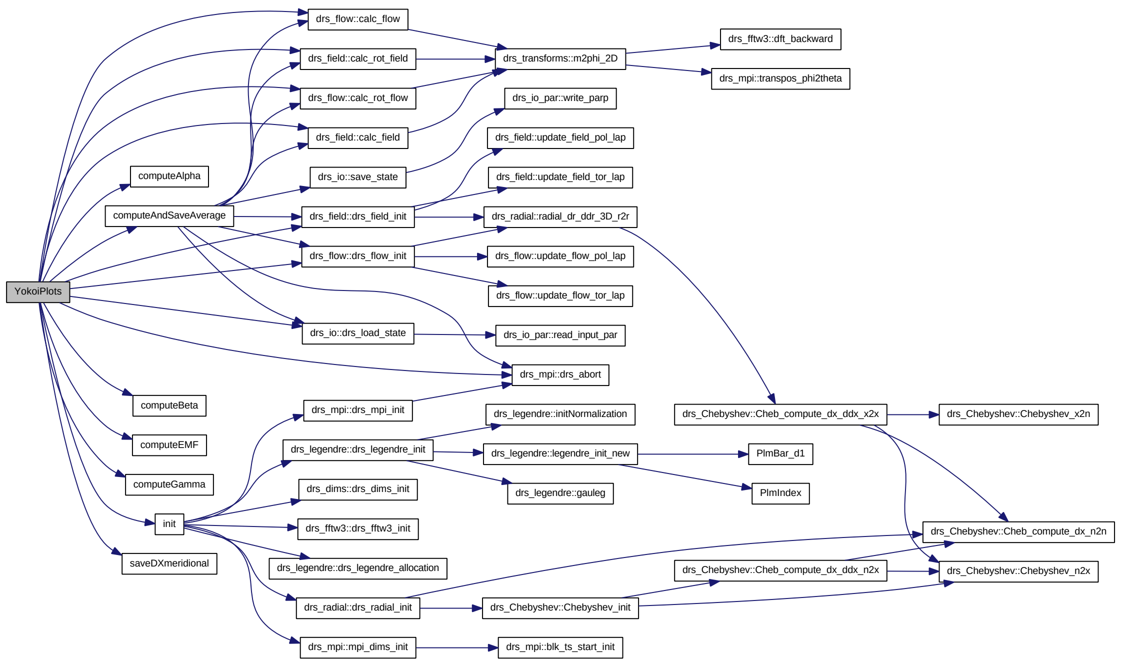
References drs_legendre::costheta, drs_probes::dOmega, drs_dims::m0, drs_dims::Nt, drs_legendre::pi, and drs_radial::rcoll.

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 saveDXmerid-ional().

Here is the call graph for this function:


YokoiPlots
computeAlpha
computeAndSaveAverage
computeBeta
computeEMF
computeGamma
init
saveDXmeridional
drs_flow::calc_flow
drs_field::calc_rot_field
drs_flow::calc_rot_flow
drs_field::calc_field
drs_io::save_state
drs_field::drs_field_init
drs_flow::drs_flow_init
drs_io::drs_load_state
drs_transforms::m2phi_2D
drs_fftw3::dft_backward
drs_mpi::transpos_phi2theta
drs_io_par::write_parp
drs_field::update_field_pol_lap
drs_field::update_field_tor_lap
drs_radial::radial_dr_ddr_3D_r2r
drs_flow::update_flow_pol_lap
drs_flow::update_flow_tor_lap
drs_io_par::read_input_par
drs_mpi::drs_abort
drs_mpi::drs_mpi_init
drs_legendre::drs_legendre_init
drs_legendre::initNormalization
drs_legendre::legendre_init_new
drs_legendre::gauleg
PlmBar_d1
PlmIndex
drs_dims::drs_dims_init
drs_fftw3::drs_fftw3_init
drs_legendre::drs_legendre_allocation
drs_radial::drs_radial_init
drs_mpi::mpi_dims_init
drs_Chebyshev::Chebyshev_init
drs_mpi::blk_ts_start_init
drs_Chebyshev::Cheb_compute_dx_ddx_x2x
drs_Chebyshev::Chebyshev_x2n
drs_Chebyshev::Cheb_compute_dx_n2n
drs_Chebyshev::Cheb_compute_dx_ddx_n2x
drs_Chebyshev::Chebyshev_n2x

Index