WIPPS

Generated by Doxygen 1.8.9.1

Tue Jun 6 2017 14:40:08

Contents

1 Main Page

1.1 Introduction

This program post-processes EPOCH PIC code output files to explore Whistler wave-particle interactions. It can create wave spectra, and calculate theory based growth rates and particle diffusion coefficients.

Function is divided across a set of utilities. `make list_utils` will list their names and each has an entry in Available programs. For example, generate_ffts processes input data and outputs trimmed FFTs, or calculate_↩ growth calculates theoretical growth rates of whistlers.

The Modules section shows the various parts of the code. Broadly, we have classes to store data, access data from

files, represent physical entities such as plasma, and non-thermal electrons, and perform required calculations, and collections of helpers for common actions.

1.1.1 Code Setup

Setup and use of the code is via a combination of config files and command line arguments. These specify the domain, working directory etc. For each program, help on the command line arguments is available with `name -h` Larger pieces of configuration are read from files. We use the deck.status file generated by EPOCH for user defined constants, a plasma.conf file for background plasma configuration and a non-thermal.conf file for non-thermal electron distributions (required for growth rate calculations). Examples of all the conf files are in the files subdirectory.

Some of the programs can use multiple cores, by parallelising over space.

1.1.2 Arrays

Data arrays are a specialised class containing data, axes and information. Get/set_element methods are provided for accessing specific parts. Spectrum and diffusion_coeff's are specialised data arrays representing physical entities.

1.1.3 SDF IO

SDF file reading uses the SDF C libraries. A copy of these is provided with the code, and an installer script install given to build them.

1.1.4 Fourier Transforms and Special Functions

FFTs are handled by the FFTW routines in suitable precision (float or double). Special functions are provided by Boost.

1.1.5 Spectrum and D generation

Some objects are deeply connected. In particular, spectrums are connected to plasmas, and diffusion_coeffs are deeply connected to spectrums. Keeping these connected is the job on the controller class, which creates and stores them.

1.2 Prerequisites

As well as the code, an install of boost is needed (exists on OSX and most Linux systems) as well as the correct version of FFTW libraries (float for float data, double for double). A copy of EPOCH's SDF file library is included with the code. To generate the docs Doxygen and pdftex are used. See Building the Code for information on how to build prereqs.

1.3 This Documentation

These docs describe all classes and methods under the Classes section. Helper functions, constants etc are grouped under Modules . Assumptions and caveats within the code are collected under Caveats and Assumptions.

Full and User docs are available. The former includes all private entities, function references, and a full source code listing. The latter does not. Change between these modes with `make DOCS=full` and `make DOCS=user`

1.4 Building the Code

An install script is provided to build SDF, install fftw etc. For details run `./install --help` Build using make. If input data is type double, use `make TYPE=double` to build with correct FFT etc libraries.

Once the code has been built, `make tar_built` to produce a tarball Runnable.tgz of utilities and necessary files that can be copied elsewhere and run.

1.5 IDL routines

Some IDL helpers are provided for reading the output files, reading the deck.status file etc. The simple way to use these is to copy the enclosed .idlstartup file to the directory you run analysis from and set this as a startup file like, for example, `pref_set, "IDL_STARTUP", "/path/to/.idlstartup",/commit` Then set a `WIPPS_PATH` environment variable giving the location of the wipps code. Start idl, and you should see a series of "% Compiled module:" lines. Try `IDL> v_to_kev(0.5) 79.051798621061963` to test

1.6 Modifying the Code

Any changes to code include files or addition of files will change dependencies. In this case run `make echo_deps` before a clean build to regenerate the makefile listing of included headers.

test, profile and debug modes are also available, and are invoked with `make MODE=[test,profile,debug]` after a clean. More details on testing are below.

See the Todo List and Extensions to Consider pages for tasks and extensions.

1.6.1 Versioning

Version numbers follow the usual major-minor numbering. So in general any change to file-io, data normalisation etc which may break compatibility should bump the major number, and any substantive change to how things are done bumps the minor. For example, adding a new field to data files, or changing the FFT normalisation by 2!pi is a major change. Tweaking spectrum extraction to be different but equally correct is a minor. Internal-only changes change neither number.

Version number is derived using a git tag, so to set one do something like `git tag -a vx.y -m "Messsage here"` before commit and make sure to push tags

Generally only some utilities will be broken by changes, so we don't necessarily quit on mismatch. It's probably easiest to write a small file-converter to update old files and solve errors that way.

1.7 Integrated Testing

All significant parts of the code should be covered by inbuilt tests. These are defined in tests.cpp, tests_basic_and_code.cpp and tests_data_and_calc.cpp and cover a mixture of unit testing, science testing and library integration tests. To run the tests, clean build with `make clean && make MODE=test` and run `./main`. Errors and outcomes are written to stderr, information to stdout. Thus `./main 1> /dev/null` will show only the former, etc. A logfile is produced, by default named tests.log. Temporary testing files are written into tests::tests_tmp_dir. Some tests are rather heavy so are only run if flags are set. See tests in the Test Docs for these. By default, all compiles are at optimisation O3. To compile with O0 for testing or debugging, pass NO_OPT = 1 to the make command.

Consider adding tests for any significant additions or changes, and running the existing ones in this case. To include tests in this documentation, run `make MODE=test docs`. Similarly, to omit them, make without test mode.

1.8 Minimally wrong description of C++ as used here

1.8.1 Types and typedefs

There's two ways to do a character string, old C-style array or characters, or a std::string (see below for meaning of std::). Some older functions expect the former, so the .c_str() conversion is used. I use C-strings for some things to match the libraries being used. The special type size_t is defined as an unsigned integer type large enough to count "anything". Unsigned so cannot be negative, but is used for counts, sizes etc. I have also added a typedef for the type in the data files and that to do the calculations in, my_type and calc_type respectively, defined in support.h. I decided to do all calculations as double, but as my data are float I use float versions of the FFTW libraries.

1.8.2 Classes

For this code, classes are basically structs containing data, with special methods (member functions). These always know the contents of the class and can access bits that might be hidden from the outside (private or protected). One class can extend another, as data_array does to my_array. The former holds just a chunk of data, the latter adds axes and additional functions. Once you have a class instance, i.e. a variable containing a thing of that class, you can call methods on it using the '.' Constructors are functions used to set up a new instance. For instance, if I want to make a 10x10 array, we set the parameters recording the dimension to define a 2-d array with sizes 10 and 10, and we grab some memory to store 10x10=100 data values etc. These special functions look like class_name(parameter list). We also have a destructor, which clean up when the variable is destroyed, and some special things which let us make copies, set one thing equal another and so on. These can be safely ignored. Quick example:

```
data_array dat = data_array(10, 10); //Make a new 10x10 array
dat.set_element(5, 5, 2.0); //Set element 5, 5 to 2.0
my_type element = dat.get_element(5, 5); //element = 2.0
```

1.8.3 Default arguments

Function definitions can set default values for arguments which will be used if nothing is explicitly given. For example,

```
 int add(int a, int b = 1){return a+b;}
cout<< add(1, 2)<<endl; //Prints 3
cout<< add(5)<<endl; //Prints 6
```

With two arguments this prints their sum, but if only one if given, b is set to a value of 1.

1.8.4 Pointers * and ->

Some classes get rather large, for instance if they hold a lot of data internally. In this case, you might want to pass them about not by value (copying everything) but just by getting a pointer to where they are. Pointer variables are defined like class * my_pointer with an asterix. This variable holds only the address. Conversions between pointer and instance are:

```
 class * my_pointer = new class();
class my_instance = class();
my_instance = *(my_pointer); 'dereference' pointer to get value it points to
my_pointer = &(my_instance); take 'address of' instance to get pointer.
```

The special operator '->' is used to apply a method to a pointer:

```
 my_instance.set_element(5, 5, 2.0);
my_pointer->set_element(5, 5, 2.0);
```

Some of the core code uses these, none of the stuff in use should. The special id 'this' inside a class method refers to the instance the method was called on and is a pointer, so uses this-> & in a function means that you may pass an instance, but a copy will not be made, the function will just be given the address. For instance the function to read data into an array dat is

```
my_reader.read_data(dat, time, space);
```

But a copy of dat is not made.

1.8.5 The double colons ::

The :: appears either with something like std:: or boost::math:: or with a class name, and means that this refers to the function X in that library, class etc (i.e. the function in the given namespace). So there might be a function abs() in the standard std library and in a math library and one must distinguish between them. Or both my_array and data_array have a function called is_good() and the definitions must state which they refer to.

2 Caveats and Assumptions

Member diffusion_coeff::calculate (D_type_spec type_of_D=D_type_spec::alpha_alpha, bool quiet=0)

We re-range the D calculation so what is actually calculated is D/p^2

We currently use plasma::get_high_dens_phi_mu_om which uses the high-density approximation to the plasma dispersion, as does the resonant frequency solver plasma::get_resonant_omega. This missed some solutions for smaller om_pe/om_ce. Above 3 or so seems to be alright

Module dist_util

This utility only works for 2-D distributions right now.

globalScope> Member get_G1 (spectrum *my_spect, calc_type omega)

We assume omega symmetry here and just take abs(omega). If this is changed, best to add a omega_symm flag to spectra and select based on that. Note also that this routine uses the given spectrums omega range and assumes that beyond this there is no wave power

globalScope> Member get_G2 (spectrum *my_spect, calc_type omega, calc_type x)

We assume omega symmetry here and just take abs(omega). See get_G1 for more

Class non_thermal

This is written for the input.deck files I used, so assumes, for example, that the term called "dens" in deck.status is the cold plasma density. The names of deck constants for additional species etc are set using the conf files.

Class plasma

Note that no spatial variations in density or ambient B field are included in the dispersion calculations here

Member plasma::get_dispersion (my_type k, int wave_type, bool reverse=0, bool deriv=0, my_type theta=0.0) const

For Whistler modes this is an approximation and intended to be perfectly reversible.

Member plasma::get_high_dens_phi_mu_om (calc_type w, calc_type psi, calc_type alpha, int n, calc_type gamma_particle, bool skip_phi=false, bool Righthand=true) const

Unsurprisingly this routine uses a high density approximation to the dispersion, which assumes $\omega_{pe} >> \omega_{ce}$

This routine assumes that electrons are the first species of the plasma, i.e. the first species in the plasma.conf file

Member plasma::get_resonant_omega_full (calc_type theta, calc_type v_par, calc_type gamma_particle, int n) const

I am assuming the solutions move but no more appear in the full equation. This may not be accurate, but we need a first guess for the solver

We set a hard minimum for resonant frequencies of interest, NR_min_om

Member plasma::plasma (std::string file_prefix, my_type Bx_local=-1)

The ion frequencies assume a single ion species right now

3 Extensions to Consider

Class bounce_av_data

Actually use this B_x rather than assuming a dipole

Class controller

Consider adding copy and move constructors etc

globalScope> Member get_Bx (std::string file_prefix, size_t space_in[2], size_t time_0)

Add 3-D space handling!

Member plasma::get_resonant_omega (calc_type theta, calc_type v_par, calc_type gamma_particle, int n) const

Extend this to use full solution rather than the high density approx

4 Todo List

Member controller::add_spectrum (std::string file)

Perhaps this should append the file_prefix to file??

Member data_array::write_to_file (std::fstream &file, bool close_file=true)

Test read/write with double, also IDL

Member diffusion_coeff::calculate (D_type_spec type_of_D=D_type_spec::alpha_alpha, bool quiet=0)

Check for double counting

Member diffusion_coeff::get_element_by_values (my_type p, my_type alpha)

Q? try interpolating?

globalScope> Member estimate_spectrum_noise (spectrum &spec_in)

Find better noise estimate...

globalScope> Member get_G2 (spectrum *my_spect, calc_type omega, calc_type x)

Currently interpolates angle only. Perhaps interpolate on omega too?

globalScope> Member main (int argc, char *argv[])

generate_spectrum goes wrong if spec_sz != k_sz Note we have to rebin if we change to work

We have to do the time bins better than this!

FFT normalisation -> V2.0. Create file converter if so

Member my_array::populate_complex_slice (my_type *dat_in, size_t n_dims, size_t *offsets, size_t *sizes)

Add testing of this

Member my_array::shift (size_t dim, long n_els)

Fix special case

globalScope> Member my_print (std::string text, int rank, int rank_to_write=0, bool noreturn=false)

Check what happens when multi-cores print....

Class non_thermal

Consider using .conf to set background params too

Member plasma::get_dispersion (my_type k, int wave_type, bool reverse=0, bool deriv=0, my_type theta=0.0) const

Complete Xmode?

globalScope> Member process_filelist (int argc, char *argv[])

Write sort rather than using map shortcut

Member resonance_poly::calculate_coeffs_full (double psi, double v_par, int n, double gamma)

Pre-calc and stash as much as possible, in particular consider psi in A, B, C

Member resonance_poly::calculate_coeffs_no_ion (double psi, double v_par, int n, double gamma)

Pre-calc and stash as much as possible, in particular consider psi in A, B, C

Module spect_util

Add per-util target to makefile

Member spectrum::generate_spectrum (data_array &parent, int om_fuzz, int angle_type, my_type std_dev, data_array *mask=nullptr)

2-d and 3-d extractions don't quite agree at k=0. factor ~10 and variations near 0

Module utils

Add install recipe to makefile

Since DOCS are now complex, consider just using two Doxyfiles...

5 Module Documentation

5.1 Diffusion calculation utility

Utility to calculate a particle diffusion coefficient.

Classes

- struct diff_cmd_line

  Command line arguments for diffusion calculation utility.

Functions

- diff_cmd_line special_command_line (int argc, char *argv[])
- bool is_filenumber (char *str)
- int main (int argc, char *argv[])

  Main program.

Variables

- const char PER_UTIL_HELP_ID = 'i'

5.1.1 Detailed Description

Utility to calculate a particle diffusion coefficient.

Calculates a particle diffusion coefficient from given data, in the form of sdf files, ffts or spectrum files. The latter can be created by the generate_ffts and FFT_to_spectrum utils. Note that cross-version compatibility is not guaranteed, but most normalisation changes etc will cancel. The resulting particle diffusion coefficient are calculated using [3] [4], [1] and such. Note that this makes no sense for E fields! Depends on the SDF file libraries, the FFTW library, and boost's math for special functions. A set of test arguments is supplied. Call using ./calculate_diffusion <test_pars to use these. Or try ./calculate_diffusion -h for argument help

```
Command line options:
-h Show help
-f <string> Prefix for all input files, including directory
-d <int int> Dimension of D to generate
-ref <int/string> EITHER a reference SDF file number (default 0) giving an sdf file containing Bx data or a WI
-om_lims <float float> Limits of omega/omega_ce to truncate spectrum before calculation
-ang_lims <float float> Limits of tan(theta) to truncate spectrum before calculation

Supply one and only one of:
-Finput <string ...> Start from [list of] FFT files
-Sinput <string ...> Start from [list of] spectrum files
If one filename is supplied, no bounce averaging is done. A list of files with names ending in [timebins]_[spa

If supplying FFTS with -Finput the following args can be given:
-wave <char> Identifies wave mode, *W*histler, *P*lasma, *O*rdinary EM (Default whistler)
-om <int> Percent "fuzz" around dispersion curve (default 10%)
-n_ang <int> Number of angles to use (default set in support.h)
-ang <int> Angular function to generate *D*elta, *G*auss, *I*so (default Delta) N-D data (N, space dims, >1) i
```

```
-ang_w <float> Width of angular function (stddev for Gaussian, width for iso etc)
-extr Extract angles from Data (N>1) only. Flatten data to 2 spatial dims and extract angular dependence. Over
-smooth <int> Smooth the intermediate B spectrum using this boxcar width

Sample usage:
make TYPE=double
mpiexec -n 4 ./calculate_diffusion `<test_pars`
or:
./calculate_diffusion -f ./Run1/ -ref 0 -d 100 100
```

Author

Heather Ratcliffe

Date

17/02/2017

5.1.2 Function Documentation

5.1.2.1 bool is_filenumber (char * *str*)

Check string is valid filenumber

Checks string contains only digits, as filenumbers can't be negative or floats

Parameters

str	String to check

Returns

Boolean true if valid, false else

5.1.2.2 int main (int *argc,* char * *argv[ ]*)

Main program.

Calculate particle diffusion from a spectrum

Parameters

argc	Command line argument count
argv	Command line arguments

Returns

System error code

Todo generate_spectrum goes wrong if spec_sz != k_sz Note we have to rebin if we change to work

Todo We have to do the time bins better than this!

5.1.2.3 gen_cmd_line special_command_line (int *argc,* char * *argv[ ]*)

Process commandline arguments

This handles all command line arguments to this utility, so expects no arguments not listed below or in spect_↩ process_command_line

5.1.3 Variable Documentation

5.1.3.1 const char PER_UTIL_HELP_ID = 'i'

ID to identify help file for this utility

5.2 Growth rate calculation utility

Utility to calculate growth rates.

Classes

- struct g_args

  Additional command line arguments for growth calculation utility.

Functions

- calc_type * make_momentum_axis (int n_mom, calc_type v_max)
- calc_type get_growth_rate (plasma *my_plas, non_thermal *my_elec, int n_momenta, calc_type *p_axis, calc_type omega_in)
- g_args g_command_line (int argc, char *argv[])
- bool write_growth_closer (std::string in_file, size_t n_momenta, calc_type min_v, calc_type max_v, std::fstream &file)
- std::vector< std::string > read_filelist (std::string infile)
- void dump_distrib (non_thermal *my_elec, std::string filename)
- calc_type estimate_spectrum_noise (spectrum &spec_in)
- int main (int argc, char *argv[])

  Main program.

Variables

- const char PER_UTIL_HELP_ID = 'w'
- const int n_trials = 2000

5.2.1 Detailed Description

Utility to calculate growth rates.

Calculates analytical growth rate using electron distribution specified in {path}nonthermal.conf and if requested also calculates linear growth from a series of spectrum files. These must be in time-order and all have the same axes. The analytics growth rates use the axes as in spectrum files, if supplied, or a wide coverage log axis with n_trials elements otherwise. We use plasma.conf and deck.status files for configuration, and a nonthermal.conf file to create the non-thermal electron distribution. See non_thermal for details. A set of test arguments is supplied as growth_test_pars

```
Command line options:
-h Show help
-f <string> Path for for all input files (full or relative) e.g. deck.status and plasma.conf
-s <string> File containing input spectra names (optional)
-out <string> Name of output file to create

Sample usage:
make utils TYPE=float
./calculate_growth -f ./Run1/ -out growth.dat
or:
./calculate_growth `<growth_test_pars`
```

Author

Heather Ratcliffe

Date

11/02/2016

5.2.2 Function Documentation

5.2.2.1 void dump_distrib (non_thermal * *my_elec,* std::string *filename*)

Write out non_thermal distribution

Write out a table of the current non_thermal distribution. Writes fixed number of elements between fixed bounds. Used for testing

Parameters

my_elec	Non-thermal electrons to dump
filename	Full filepath to dump to

5.2.2.2 calc_type estimate_spectrum_noise (spectrum & *spec_in*)

Estimates the "noise" in a given spectrum.

There are many ways we might do this, using varying amounts of additional information about the "spectrum". Averaging the end values works poorly. Averaging the largest two values that are greater than something? Fit a straight line to the 0.8 to 1 region?

Parameters

spec_in	Input spectrum

Returns

Value of estimated noise for this input spectrum

Todo Find better noise estimate...

5.2.2.3 g_args g_command_line (int *argc,* char * *argv[ ]*)

Process the extra command line args for calculate_growth

Check whether to handle real spectra. If -s we use the spectra (plural) listed in that file, if not we output analytic growth only. -out specifies the output file name. Use in conjunction with process_command_line for the rest

5.2.2.4 calc_type get_growth_rate (plasma * *my_plas,* non_thermal * *my_elec,* int *n_momenta,* calc_type * *p_axis,* calc_type *omega_in*)

Calculate growth rate

Calculates the analytical growth rate at omega_in by integrating on p_axis, using supplied plasma and non_thermal distributions.

Parameters

my_plas	Plasma object to use
my_elec	Non-thermal electron object to use
n_momenta	Number of elements in p_axis
p_axis	The momentum axis to use for integration
omega_in	Frequency to eval. growth rate at

Returns

Calculated growth rate

5.2.2.5 int main (int *argc,* char * *argv[ ]*)

Main program.

Calculate linear theory growth rates

Parameters

argc	Command line argument count
argv	Command line arguments

Returns

System error code

5.2.2.6 calc_type * make_momentum_axis (int *n_mom,* calc_type *v_max*)

Create a momentum axis

Creates linear axis for momentum starting at 0

Parameters

n_mom	number of elements in resulting axis
v_max	Max value of axis, divided by v0

Returns

Pointer to start of created axis

5.2.2.7 std::vector< std::string > read_filelist (std::string *infile*)

Read list from file

Reads a list of strings from given infile, omitting blanks

Parameters

infile	Complete filepath to read from

Returns

Vector of the filenames read

5.2.2.8 bool write_growth_closer (std::string *in_file,* size_t *n_momenta,* calc_type *min_v,* calc_type *max_v,* std::fstream & *file*)

Close growth rate file

Write the general parameters to specified file, including the file-location markers.

Parameters

in_file	Input filepath (to write)
n_momenta	Number of momenta used in calcs
min_v	Minimum particle velocty used in calcs
max_v	Maximum particle velocty used in calcs
file	Filestream to write to

Returns

0 if writing successful, 1 for error

5.2.3 Variable Documentation

5.2.3.1 const int n_trials = 2000

Number of data points for analytic growth (if no real data supplied)

5.2.3.2 const char PER_UTIL_HELP_ID = 'w'

ID to identify help file for this utility

5.3 Distribution extraction utility

Utility to extract distribution functions from data.

Classes

- struct dist_cmd_line

  Command line arguments for distribution utility.

Functions

- dist_cmd_line special_command_line (int argc, char ∗argv[ ])
- int main (int argc, char ∗argv[ ])

  Main program.

Variables

- const char PER_UTIL_HELP_ID = 'd'

5.3.1 Detailed Description

Utility to extract distribution functions from data.

Extracts distributions from SDF files and optionally compresses their (first) spatial dimension.

```
Command line options:
-h Show help
-f <string> filepath (prepended to in, out and deck.status files)
-dump <int> Dump number
-x_blocks <int> Divide into this many x blocks. If this does not evenly divide the x-size overlap will occur
-list Just list the present distributions (identified by presence of grid)
-extr Extract and redump, do no compression

Sample usage:
make utils TYPE=float
./compress_distributions -f ./Run1/ -dump 10 -x_blocks 8
```

Author

Heather Ratcliffe

Date

09/09/2016

Caveat This utility only works for 2-D distributions right now.

5.3.2 Function Documentation

5.3.2.1 int main (int *argc*, char ∗ *argv[ ]*)

Main program.

Extract distributions and space average

Parameters

argc	Command line argument count
argv	Command line arguments

Returns

System error code

5.3.2.2 dist_cmd_line special_command_line (int *argc,* char * *argv[ ]*)

Process commandline arguments

This handles all command line arguments to this utility, so expects no arguments not listed below

5.3.3 Variable Documentation

5.3.3.1 const char PER_UTIL_HELP_ID = 'd'

ID to identify help file for this utility

5.4 FFT cutout utility

Utility to trim FFTd data to specified axis limits.

Classes

- struct cutout_args

  Command line arguments for FFT cutout utility.

Functions

- cutout_args cutout_process_command_line (int argc, char ∗argv[ ])
- int main (int argc, char ∗argv[ ])

  Main program.

Variables

- const char PER_UTIL_HELP_ID = 'c'

5.4.1 Detailed Description

Utility to trim FFTd data to specified axis limits.

- Reads array from given file, cuts out to supplied limits and saves to given output file. deck.status file is read from file_prefix+deck.status and allows to specify frequency cuts in w_ce. Wavenumber cuts are in m^-1. If no output file is given, output will be in [inputfile]_trim. Output file gains current version number, so we enforce strict checking

```
Command line options:
-h Show help
-f <string> Filepath (prepended to in, out and deck.status files)
-in <string> Input file name
-out <string> Output file name. If absent [inputfile] is used with "trim" inserted before extension
-lims <float float ...> The limits to cutout. Should be one pair per dimension. Pairs except the last are

Sample usage:
make utils TYPE=float
./cutout_fft -f ./Run1/ -in FFT_dat -out FFT_trim -lims -0.01 0.01 -2 2
```

Author

Heather Ratcliffe

Date

11/08/2016

5.4.2 Function Documentation

5.4.2.1 cutout_args cutout_process_command_line (int *argc,* char ∗ *argv[ ]*)

Process commandline arguments

Expects full list and no more.

5.4.2.2 int main (int *argc,* char ∗ *argv[ ]*)

Main program.

Read an FFT, cutout to limits and print to new file

Parameters

argc	Command line argument count
argv	Command line arguments

Returns

System error code

5.4.3 Variable Documentation

5.4.3.1 const char PER_UTIL_HELP_ID = 'c'

ID to identify help file for this utility

5.5 Example main programs

Example main programs showing details of use.

Classes

- struct my_args

  Command line arguments for example utility.

Functions

- my_args example_process_command_line (int argc, char ∗argv[ ])
- int main (int argc, char ∗argv[ ])

  Main program.

Variables

- const char PER_UTIL_HELP_ID = 'm'
- const char PER_UTIL_HELP_ID = 's'

5.5.1 Detailed Description

Example main programs showing details of use.

- An example skeleton main program, showing how to use the various parts for a single-core or multicore program. For single-core, we initialise MPI but only do work on root (rank 0) processor.

  Author

  Heather Ratcliffe

  Date

  29/05/2017

5.5.2 Function Documentation

5.5.2.1 my_args example_process_command_line (int *argc,* char ∗ *argv[ ]*)

Process commandline arguments

Expects full list and no extra options, so will warn if it doesn't understand a key

5.5.2.2 int main (int *argc,* char ∗ *argv[ ]*)

Main program.

Do things

Parameters

argc	Command line argument count
argv	Command line arguments

Returns

System error code

5.5.3 Variable Documentation

5.5.3.1 const char PER_UTIL_HELP_ID = 'm'

ID to identify help file for this utility

5.5.3.2 const char PER_UTIL_HELP_ID = 's'

ID to identify help file for this utility

5.6 Field extractor utility

Utility to read sdf files and extract single fields (at first time if accumulated)

Classes

- struct extractor_args
  Command line arguments for field extractor utility.

Functions

- extractor_args extractor_process_command_line (int argc, char ∗argv[ ])
- int main (int argc, char ∗∗argv)
  Main program.

Variables

- const char PER_UTIL_HELP_ID = 'e'

5.6.1 Detailed Description

Utility to read sdf files and extract single fields (at first time if accumulated)

Opens files, extracts specified field. Can optionally flatten the data in space.

```
Command line options:
-h Show help
-f <string> Prefix for all input files, including directory

-start <int> Dump number to read
-block <string> Block id string
-flatten <int>  Flatten the data on specified dimension before output (0 to n_dims-1)
-out <string> Output file name (-f arg will be prepended)

Sample usage:
make TYPE=float
./extract_field -f ./Run1/ -start 0 -block bx -out Bx_ref.dat
```

Author

Heather Ratcliffe

Date

23/05/2017

5.6.2 Function Documentation

5.6.2.1 extractor_args extractor_process_command_line (int *argc,* char ∗ *argv[ ]*)

Process special arguments to extract_field

Part of argument handling is shared with calculate_diffusion and generate_ffts so we handle only the extras here and must pass the rest on. So we nullify those we handle here to enable warning for unknown arguments by setting the first character of them all to HANDLED_ARG

5.6.2.2 int main (int *argc,* char ∗∗ *argv*)

Main program.

Extract distributions and space average

Parameters

argc	Command line argument count
argv	Command line arguments

Returns

System error code

5.6.3 Variable Documentation

5.6.3.1 const char PER_UTIL_HELP_ID = 'e'

ID to identify help file for this utility

5.7 Spectrum generation utility

Utility to generate a spectrum from Fourier transformed data.

Classes

- struct fft_spect_args

  Command line arguments for spectrum generation utility.

Functions

- fft_spect_args fft_spect_process_command_line (int argc, char ∗argv[ ])
- int main (int argc, char ∗argv[ ])

  Main program.

Variables

- const char PER_UTIL_HELP_ID = 'f'

5.7.1 Detailed Description

Utility to generate a spectrum from Fourier transformed data.

Requires an input directory and an input fft'd data file. The output is either specified, or is the input file with _↩ spectrum appended before the extension. The "wave" option specifies the wave mode by single-character key (w, p, o) and defaults to Whistler. A "fuzz" parameter controlling how tight a band around the dispersion curve can be supplied as a percentage, default is 10%. Spectra contain both frequency and angle data, the n_ang, ang and extra flags control this. FFTs may not stay compatible cross-code version, so we do a version check first.

```
Command line options:
-h Show help
-f <string> Filepath (prepended to in, out and deck.status files)
-in <string> Input file name
-out <string> Output file name. If absent [inputfile] is used with spectrum inserted before extension
-wave <char> Identifies wave mode, *W*histler, *P*lasma, *O*rdinary EM (Default whistler)
-om <int> Percent "fuzz" around dispersion curve (default 10%)
-n_ang <int> Number of angles to use (default set in support.h)
-ang <int> Angular function to generate *D*elta, *G*auss, *I*so (default Delta) N-D data (N, space dims, >1) i
-ang_w <float> Width of angular function (stddev for Gaussian, width for iso etc)
-extr Extract angles from Data (N>1) only. Flatten data to 2 spatial dims and extract angular dependence. Over
-smooth <int> Smooth the output B spectrum using this boxcar width

-mask Also output a mask, filename has _mask inserted before extension

Sample usage:
./fft_to_spectrum -f ./Run1/ -in FFT_dat -out spect_dat -wave w
```

Author

Heather Ratcliffe

Date

12/08/2016

Todo Add per-util target to makefile

5.7.2 Function Documentation

5.7.2.1 fft_spect_args fft_spect_process_command_line (int *argc,* char * *argv[ ]*)

Process special command line args

Process the fft utility arguments. Expects full list and no more.

5.7.2.2 int main (int *argc,* char * *argv[ ]*)

Main program.

Convert FFT to spectrum and write to file

Parameters

argc	Command line argument count
argv	Command line arguments

Returns

System error code

Todo FFT normalisation -> V2.0. Create file converter if so

5.7.3 Variable Documentation

5.7.3.1 const char PER_UTIL_HELP_ID = 'f'

ID to identify help file for this utility

5.8 FFT generator utility

Utility to read files and perform Fourier transforms.

Classes

- struct gen_cmd_line

  Additional command line arguments for FFT generation utility.

Functions

- gen_cmd_line special_command_line (int argc, char ∗argv[ ])
- int main (int argc, char ∗argv[ ])

  Main program.

Variables

- const char PER_UTIL_HELP_ID = 'g'

5.8.1 Detailed Description

Utility to read files and perform Fourier transforms.

Opens files, extracts specified fields, does FFT, trims to specified boundaries and writes to file. Can optionally flatten the raw data before Ft-ing or the FT-d data before output. This routine can use multiple cores to process seperate spatial blocks.

```
Command line options:
-h Show help
-f <string> Prefix for all input files, including directory

-start <int> Starting dump number
-end <int> Ending dump number
-rows <int> For accumulated blocks, read no more than this many rows To get an exact number of rows set -end l
-block <string> Block id string
-n <int> Number of space blocks to divide into. This should be divisible by the number of processors used or i
-space <int int> Process one space block between int and int (use with single core only)

-flat_dat <int>  Flatten the raw data on specified dimension before processing (0 to n_dims-1)
-flat_fft <int float float> Flatten the FFT on specified dimension between these axis ranges (normalised to om
-lims <float float ...> Cutout to these limits. Should be one pair per dimension (after flattening if applicab

Sample usage:
make TYPE=double
./generate_ffts -f ./Run1/ -start 0 -end 100 -block bz -n 8 -lims -0.001 0.001 -2 2
or:
mpiexec -n 4 ./generate_ffts -f ./Run1/ -start 0 -end 100 -block bz -n 8 -lims -0.001 0.001 -2 2
```

Author

Heather Ratcliffe

Date

04/07/2016

5.8.2 Function Documentation

5.8.2.1 int main (int *argc,* char * *argv[ ]*)

Main program.

Generate an FFT

Parameters

argc	Command line argument count
argv	Command line arguments

Returns

System error code

5.8.2.2 gen_cmd_line special_command_line (int *argc*, char * *argv[ ]*)

Process special arguments to generate_ffts

Part of argument handling is shared with calculate_diffusion so we handle only the extras here and must pass the rest on. So we nullify those we handle here to enable warning for unknown arguments by setting the first character of them all to HANDLED_ARG

5.8.3 Variable Documentation

5.8.3.1 const char PER_UTIL_HELP_ID = 'g'

ID to identify help file for this utility

5.9 Available programs

Top-level programs available.

Modules

- Diffusion calculation utility

  Utility to calculate a particle diffusion coefficient.
- Growth rate calculation utility

  Utility to calculate growth rates.
- Distribution extraction utility

  Utility to extract distribution functions from data.
- FFT cutout utility

  Utility to trim FFTd data to specified axis limits.
- Field extractor utility

  Utility to read sdf files and extract single fields (at first time if accumulated)
- Spectrum generation utility

  Utility to generate a spectrum from Fourier transformed data.
- FFT generator utility

  Utility to read files and perform Fourier transforms.
- Info program

  Info program.

5.9.1 Detailed Description

Top-level programs available.

Todo Add install recipe to makefile

Since DOCS are now complex, consider just using two Doxyfiles...

Contains programs to perform FFTs, create spectra, calculate growth etc. Build with `make utils` List with `make list_utils` Command line argument help is available using `./{util_name} -h`

5.10 Info program

Info program.

Functions

- int main (int argc, char ∗argv[ ])
  Main program.

Variables

- const char PER_UTIL_HELP_ID = ' '

5.10.1 Detailed Description

Info program.

This prints information about the code or compiled in test mode it runs the tests and prints results.

5.10.2 Function Documentation

5.10.2.1 int main (int *argc,* char ∗ *argv[ ]*)

Main program.

Compiled in test MODE this runs the testing code. Otherwise it prints info.

Author

Heather Ratcliffe

Date

18/09/2015.

Parameters

argc	Command line argument count
argv	Command line arguments

Returns

System error code

5.10.3 Variable Documentation

5.10.3.1 const char PER_UTIL_HELP_ID = ' '

ID to identify help file for this utility

5.11 Bounce-averaging helpers

Helpers for bounce-averaging process.

Classes

- class bounce_av_data
  Bounce-averaging data.

Typedefs

- typedef enum bounce_av_type_specs bounce_av_type
  Specifies "parameters" of D to determine how to bounce average.

Enumerations

- enum bounce_av_type_specs {
  plain, **alpha_alpha**, **alpha_p**, **p_alpha**,
  p_p }
  Specifies "parameters" of D to determine how to bounce average.

Functions

- my_type solve_mirror_latitude (my_type alpha_eq, bool print_iters=false)
- my_type mirror_poly (my_type L, my_type s4alpha)
- my_type d_mirror_poly (my_type L, my_type s4alpha)
- my_type Newton_Raphson_iteration (my_type last_guess, my_type s4alpha)
- my_type bounce_period_approx (my_type alpha_eq)
- my_type f_latitude (my_type lat)
- my_type alpha_from_alpha_eq (my_type alpha_eq, my_type lat)

5.11.1 Detailed Description

Helpers for bounce-averaging process.

Functions to solve for mirror latitude, bounce period etc and various helpers for these. We assume the usual dipole magnetic field.

5.11.2 Function Documentation

5.11.2.1 my_type alpha_from_alpha_eq (my_type *alpha_eq*, my_type *lat*) [inline]

Calculate alpha at given latitude from alpha_eq Summers Et Al[6] Eq 22.

Parameters

alpha_eq	Equatorial pitch angle in radians
lat	Latitude in radians

Returns

Pitch angle at this latitude

5.11.2.2 my_type bounce_period_approx (my_type *alpha_eq*) [inline]

Bounce time alpha factor from Summers Et Al [6] Eq 29 or Glauert and Horne [2] Eq 27

Parameters

alpha_eq	Equatorial particle pitch angle

Returns

The alpha factor

5.11.2.3 my_type d_mirror_poly (my_type *L*, my_type *s4alpha*) [inline]

Derivative of mirror_poly

Parameters

L	cos^2 lambda_mirror
s4alpha	sin^4 alpha for alpha particle pitch angle

Returns

Value of the derivative of the mirror polynomial

5.11.2.4 my_type f_latitude (my_type *lat*) [inline]

Calculate f as in Summers Et Al [6] Eq 20.

Parameters

lat	Latitude in RADIANS

Returns

Value of f(lat)

5.11.2.5 my_type mirror_poly (my_type *L*, my_type *s4alpha*) [inline]

Polynomial describing the mirror latitude

Parameters

L	cos^2 lambda_mirror
s4alpha	sin^4 alpha for alpha particle pitch angle

Returns

Value of mirror polynomial

5.11.2.6 my_type Newton_Raphson_iteration (my_type *last_guess*, my_type *s4alpha*) [inline]

Iterate solution of mirror polynomial using Newton-Raphson

Parameters

last_guess	Previous guess for N-R iteration
s4alpha	sin^4 alpha for alpha particle pitch angle

Returns

Next guess for mirror poly root

5.11.2.7 my_type solve_mirror_latitude (my_type *alpha_eq*, bool *print_iters* = false)

Gets the mirror latitude for equatorial pitch angle alpha_eq

Solve the mirror latitude polynomial $L^6 + (3L - 4)\sin^4 \alpha_{eq} = 0$ where $L = \cos^2 \lambda_{mirror}$

Parameters

alpha_eq	Equatorial pitch angle in radians
print_iters	Flag to print results of all iterations

Returns

The value of mirror latitude in radians

5.12 Main Classes

Major classes.

Classes

- class controller

  Controls plasma, spectrum and d_coeff objects and their connections.
- class diffusion_coeff

  Diffusion coefficient object.
- class data_array

  Extended my_array class including axes.
- class my_array

  A basic array class.
- class non_thermal

  Nonthermal electron description.
- class plasma

  Plasma parameters and dispersion.
- class reader

  Reads SDF files into data_array.
- class spectrum

  A spectrum in omega and angle.

Enumerations

- enum spectrum::part { **B**, **ang** }

5.12.1 Detailed Description

Major classes.

The major classes we use, dealing with data, physics etc

5.12.2 Enumeration Type Documentation

5.12.2.1 enum spectrum::part [strong]

Identifier for parts of spectrum, B^2(omega) or g(omega, angle)

5.13 Spectrum access wrappers

Accessors for the two parts of spectrum, B and g.

Functions

- my_type spectrum::get_B_element (size_t n_om) const
- my_type spectrum::get_g_element (size_t n_ang) const
- my_type spectrum::get_g_element (size_t n_om, size_t n_ang) const
- void spectrum::set_B_element (size_t n_om, my_type val)
- void spectrum::set_g_element (size_t n_ang, my_type val)
- void spectrum::set_g_element (size_t n_om, size_t n_ang, my_type val)
- my_type spectrum::get_om_axis_element (size_t nx) const
- my_type spectrum::get_ang_axis_element (size_t nx) const
- long spectrum::get_om_axis_index_from_value (my_type omega) const
- long spectrum::get_ang_axis_index_from_value (my_type ang) const
- void spectrum::set_om_axis_element (size_t nx, my_type val)
- void spectrum::set_ang_axis_element (size_t nx, my_type val)
- size_t spectrum::get_g_dims () const
- size_t spectrum::get_g_dims (size_t i) const
- size_t spectrum::get_B_dims () const
- size_t spectrum::get_B_dims (size_t i) const
- size_t spectrum::get_angle_length () const
- size_t spectrum::get_omega_length () const

5.13.1 Detailed Description

Accessors for the two parts of spectrum, B and g.

Spectrum does not guarantee the internal representation of the B and g parts, so these should be used to get/set the data and axes for B and g parts

5.13.2 Function Documentation

5.13.2.1 my_type spectrum::get_ang_axis_element (size_t *nx*) const [inline]

Get angle axis element at index nx

5.13.2.2 long spectrum::get_ang_axis_index_from_value (my_type *ang*) const [inline]

Get angle axis index for value ang

5.13.2.3 size_t spectrum::get_angle_length ( ) const [inline]

Get length of angle axis

5.13.2.4 size_t spectrum::get_B_dims ( ) const [inline]

Get rank of B array

5.13.2.5 size_t spectrum::get_B_dims (size_t *i*) const [inline]

Get dimensions of B array

5.13.2.6 my_type spectrum::get_B_element (size_t *n_om*) const [inline]

Get B element frequency index n_om

5.13.2.7 size_t spectrum::get_g_dims ( ) const [inline]

Get rank of g array

5.13.2.8 size_t spectrum::get_g_dims (size_t *i*) const [inline]

Get dimensions of g array

5.13.2.9 my_type spectrum::get_g_element (size_t *n_ang*) const [inline]

Get g element at angle n_ang (separable spectra)

5.13.2.10 my_type spectrum::get_g_element (size_t *n_om*, size_t *n_ang*) const [inline]

Get g element at frequency n_om, angle n_ang (nonseparable spectra)

5.13.2.11 my_type spectrum::get_om_axis_element (size_t *nx*) const [inline]

Get frequency axis element at index nx

5.13.2.12 long spectrum::get_om_axis_index_from_value (my_type *omega*) const [inline]

Get frequency axis index for value omega

5.13.2.13 size_t spectrum::get_omega_length ( ) const [inline]

Get length of frequency axis

5.13.2.14 void spectrum::set_ang_axis_element (size_t *nx*, my_type *val*) [inline]

Set angle axis index for value ang

5.13.2.15 void spectrum::set_B_element (size_t *n_om*, my_type *val*) [inline]

Set B element frequency index n_om

5.13.2.16 void spectrum::set_g_element (size_t *n_ang*, my_type *val*) [inline]

Set g element at angle n_ang (separable spectra)

5.13.2.17 void spectrum::set_g_element (size_t *n_om*, size_t *n_ang*, my_type *val*) [inline]

Set g element at frequency n_om, angle n_ang (nonseparable spectra)

5.13.2.18 void spectrum::set_om_axis_element (size_t *nx*, my_type *val*) [inline]

Set frequency axis index for value omega

5.14 Spectrum calculations

Calculate normalised spectra.

Functions

- calc_type get_G1 (spectrum *my_spect, calc_type omega)
- calc_type get_G2 (spectrum *my_spect, calc_type omega, calc_type x)

5.14.1 Detailed Description

Calculate normalised spectra.

To calculate diffusion coefficients we need the spectrum at a point, with proper normalisation. These calculate the factors called G_1 and G_2 in Albert [1]

5.14.2 Function Documentation

5.14.2.1 calc_type get_G1 (spectrum * *my_spect,* calc_type *omega*)

G1 from [1]

Gets the value of B^2(w) (interpolated if necessary) and the normalising constant from norm_B. NB NB we omit the $\Delta\omega$ factor since it cancels in the D calc

Parameters

my_spect	Input spectrum to work on
omega	Frequency to eval. at

Returns

Normalised abs-square wave power

Caveat We assume omega symmetry here and just take abs(omega). If this is changed, best to add a omega_↩ symm flag to spectra and select based on that. Note also that this routine uses the given spectrums omega range and assumes that beyond this there is no wave power

5.14.2.2 calc_type get_G2 (spectrum * *my_spect,* calc_type *omega,* calc_type *x*)

Get G2 from [1]

Gets the value of g(w, x) and the normalising constant from norm_g

Parameters

my_spect	Input spectrum to work on
omega	Frequency to eval. at
x	Tan(theta) to eval. at

Returns

Normalised angular contribution

Caveat We assume omega symmetry here and just take abs(omega). See get_G1 for more

Todo Currently interpolates angle only. Perhaps interpolate on omega too?

5.15 Technical stuff

Types, constants, data structs etc.

Modules

- Type selection

  Handles type selection to match data.
- Constants

  Code and physical constants.
- Data Structures

  Data structures for constants etc.

5.15.1 Detailed Description

Types, constants, data structs etc.

5.16 Type selection

Handles type selection to match data.

Macros

- #define ADD_FFTW(x) fftwf_ ## x
- #define cplx_type ADD_FFTW(complex)
- #define my_type float
- #define other_type double
- #define my_sdf_type SDF_DATATYPE_REAL4
- #define MPI_MYTYPE MPI_FLOAT
- #define calc_type double
- #define MPI_CALCTYPE MPI_DOUBLE
- #define tiny_calc_type 1e-12

Variables

- const my_type tiny_my_type =1e-30

5.16.1 Detailed Description

Handles type selection to match data.

5.16.2 Macro Definition Documentation

5.16.2.1 #define ADD_FFTW(*x*) fftwf_ ## x

Add the correct FFTW library prefix to function/variable names

5.16.2.2 #define calc_type double

Type to do calculations in. This is independent of the input data type my_type

5.16.2.3 #define cplx_type ADD_FFTW(complex)

Suitable complex type for FFTW work

5.16.2.4 #define MPI_CALCTYPE MPI_DOUBLE

MPI type matching calc_type

5.16.2.5 #define MPI_MYTYPE MPI_FLOAT

MPI type matching my_type

5.16.2.6 #define my_sdf_type SDF_DATATYPE_REAL4

SDF type matching my_type

5.16.2.7 #define my_type float

Input data type

5.16.2.8 #define other_type double

The other of double and float

5.16.2.9 #define tiny_calc_type 1e-12

Tiny value for calc_type

5.16.3 Variable Documentation

5.16.3.1 const my_type tiny_my_type =1e-30

Value below which we assume 0

5.17 Constants

Code and physical constants.

Variables

- const size_t MAX_SIZE = 100000
- const size_t MAX_SIZE_TOT = MAX_SIZE∗MAX_SIZE
- const int MAX_FILENAME_DIGITS = 15
- const int GIT_VERSION_SIZE = 15
- const my_type io_verify = 3.0/32.0
- const calc_type pi = 3.14159265359
- const calc_type v0 = 2.997924e8
- const calc_type me = 9.10938291e-31
- const calc_type mp = me∗1836.15267
- const calc_type q0 = 1.602176565e-19
- const calc_type eps0 = 8.85418782e-12
- const calc_type kb = 1.3806488e-23
- const calc_type R_E = 6.371e6
- const calc_type GEN_PRECISION = 1e-6
- const int DEFAULT_N_ANG = 100
- const int ID_SIZE = 10
- const int WAVE_WHISTLER = 1
- const int WAVE_PLASMA = 2
- const int WAVE_O = 3
- const int WAVE_X_UP = 4
- const int WAVE_X_LOW = 5
- const int FUNCTION_NULL = 0
- const int FUNCTION_DELTA = 1
- const int FUNCTION_GAUSS = 2
- const int FUNCTION_ISO = 3
- const std::string OMEGA_CE = "wCe"
- const std::string OMEGA_PE = "wpe"
- const std::string DENS_RAT = "dens_rat"
- const std::string DENS_RATH = "dens_rath"
- const std::string DENS = "dens"
- const std::string PPC = "ppc"
- const std::string VPAR = "vtherm_par"
- const std::string VPERP = "vtherm_perp"
- const std::string CONSTANTS = " Constant block values after"
- const std::string CONSTANTS_END = "Deck state:"
- const std::string halp_file = "./files/help/help.txt"
- const std::string LOCAL = "loc"
- const std::string BOUNCE_AV = "bav"
- const std::string GLOBAL = "glb"
- const char HANDLED_ARG [2] = "∗"
- const my_type V_MIN = 0
- const my_type V_MAX = 0.99∗v0
- const my_type TAN_MIN = 0.0
- const my_type TAN_MAX = 4.0
- const my_type ANG_MIN = 0.0
- const my_type ANG_MAX = pi/2
- const my_type DEFAULT_SPECTRUM_ANG_STDDEV = 0.2
- const my_type SPECTRUM_THRESHOLD = 1e-3

5.17.1 Detailed Description

Code and physical constants.

The subset of these which might change should be logged by each main program on each run using log_code_↩ constants(std::string). They have to be added manually.

5.17.2 Variable Documentation

5.17.2.1 const my_type ANG_MAX = pi/2

Maximum angle for spectra etc

5.17.2.2 const my_type ANG_MIN = 0.0

Minimum angle for spectra etc. Generally should be 0 or -ANG_MAX

5.17.2.3 const std::string BOUNCE_AV = "bav"

Tag identifying diffusion coefficient processing level: bounce averaged

5.17.2.4 const std::string CONSTANTS = " Constant block values after"

String denoting start of constant value dump in deck.status

5.17.2.5 const std::string CONSTANTS_END = "Deck state:"

String denoting end of constant value dump in deck.status

5.17.2.6 const int DEFAULT_N_ANG = 100

Default number of wave normal angles to consider

5.17.2.7 const my_type DEFAULT_SPECTRUM_ANG_STDDEV = 0.2

"Std Dev" of angular distribution (in tan theta) of spectrum

5.17.2.8 const std::string DENS = "dens"

String specifying density in deck.status

5.17.2.9 const std::string DENS_RAT = "dens_rat"

String specifying density ratio in deck.status

5.17.2.10 const std::string DENS_RATH = "dens_rath"

String specifying density ratio in deck.status

5.17.2.11 const calc_type eps0 = 8.85418782e-12

Epsilon_0 permittivity of free space in F/m

5.17.2.12 const int FUNCTION_DELTA = 1

Code to id spectral angular distribution as delta function (with integral 1)

5.17.2.13 const int FUNCTION_GAUSS = 2

Code to id spectral angular distribution as gaussian (with integral 1)

5.17.2.14 const int FUNCTION_ISO = 3

Code to id spectral angular distribution as isotropic (with integral 1)

5.17.2.15 const int FUNCTION_NULL = 0

Code to id spectral angular distribution as absent

5.17.2.16 const calc_type GEN_PRECISION = 1e-6

General precision for equality etc

5.17.2.17 const int GIT_VERSION_SIZE = 15

Length of git version string

5.17.2.18 const std::string GLOBAL = "glb"

Tag identifying diffusion coefficient processing level: reduced over all space

5.17.2.19 const std::string halp_file = "./files/help/help.txt"

Name of command line options help file

5.17.2.20 const char HANDLED_ARG[2] = "∗"

Dummy string to flag command line arguments as having been handled

5.17.2.21 const int ID_SIZE = 10

Length of block ids

5.17.2.22 const my_type io_verify = 3.0/32.0

An exactly binary representable my_type to verify we're reading what we're writing.

5.17.2.23 const calc_type kb = 1.3806488e-23

Boltzman constant J/K

5.17.2.24 const std::string LOCAL = "loc"

Tag identifying diffusion coefficient processing level: local to space block

5.17.2.25 const int MAX_FILENAME_DIGITS = 15

Maximum number of digits in filename dump number string

5.17.2.26 const size_t MAX_SIZE = 100000

Maximum per-dim array size allowed (per processor if MPI in use)

5.17.2.27 const size_t MAX_SIZE_TOT = MAX_SIZE∗MAX_SIZE

Maximum overall array size allowed (per processor if MPI in use)

5.17.2.28 const calc_type me = 9.10938291e-31

Electron mass in kg

5.17.2.29 const calc_type mp = me*1836.15267

Proton mass in kg

5.17.2.30 const std::string OMEGA_CE = "wCe"

String specifying omega_ce in deck.status

5.17.2.31 const std::string OMEGA_PE = "wpe"

String specifying omega_pe in deck.status

5.17.2.32 const calc_type pi = 3.14159265359

Pi

5.17.2.33 const std::string PPC = "ppc"

String specifying ppc in deck.status

5.17.2.34 const calc_type q0 = 1.602176565e-19

Electron charge in C

5.17.2.35 const calc_type R_E = 6.371e6

Average Earth radius in m

5.17.2.36 const my_type SPECTRUM_THRESHOLD = 1e-3

Fraction of peak power considered to be "significant" spectral power

5.17.2.37 const my_type TAN_MAX = 4.0

Maximum angle (tan theta) for D, spectra etc

5.17.2.38 const my_type TAN_MIN = 0.0

Minimum angle (tan theta) for D, spectra etc. Generally should be 0 or -TAN_MAX

5.17.2.39 const calc_type v0 = 2.997924e8

Speed of light in m/s^2

5.17.2.40 const my_type V_MAX = 0.99*v0

Maximum particle velocity for D

5.17.2.41 const my_type V_MIN = 0

Minimum particle velocity for D

5.17.2.42 const std::string VPAR = "vtherm_par"

String specifying parallel thermal velocity in deck.status

5.17.2.43 const std::string VPERP = "vtherm_perp"

String specifying perpendicular thermal velocity in deck.status

5.17.2.44 const int WAVE_O = 3

Code to id wave as ordinary EM mode

5.17.2.45 const int WAVE_PLASMA = 2

Code to id wave as plasma/Langmuir wave

5.17.2.46 const int WAVE_WHISTLER = 1

Code to id wave as whistler mode

5.17.2.47 const int WAVE_X_LOW = 5

Code to id wave as X EM mode, lower branch

5.17.2.48 const int WAVE_X_UP = 4

Code to id wave as X EM mode, upper branch

5.18 Data Structures

Data structures for constants etc.

Classes

- struct deck_constants

  Constants read from deck.
- struct mpi_info_struc

  MPI information.
- struct mu_dmudom

  Reduced refractive index.
- struct setup_args

  General command line arguments.
- struct spect_args

  Command line arguments for spectra.
- struct d_report

  D coefficient report.

Variables

- deck_constants my_const
- const struct mpi_info_struc mpi_info_null = {0, -1}
- mpi_info_struc mpi_info

5.18.1 Detailed Description

Data structures for constants etc.

5.18.2 Variable Documentation

5.18.2.1 mpi_info_struc mpi_info

Global mpi_info structure

5.18.2.2 const struct mpi_info_struc mpi_info_null = {0, -1}

Null MPI struct for single threaded jobs, without having to compile the SDF libraries seperately

5.18.2.3 deck_constants my_const

Global deck_constants

5.19 Helper functions

Groups of helpers of various purpose.

Modules

- Bounce-averaging helpers

  Helpers for bounce-averaging process.
- Spectrum access wrappers

  Accessors for the two parts of spectrum, B and g.
- Spectrum calculations

  Calculate normalised spectra.
- Main Helper Functions

  General helpers.
- Global Helper and Maths Functions

  String handling, IO and maths helpers.

5.19.1 Detailed Description

Groups of helpers of various purpose.

5.20 Main Helper Functions

General helpers.

Functions

- int local_MPI_setup (int argc, char ∗argv[ ])
- void safe_exit ()
- void share_consts ()
- void get_deck_constants (std::string file_prefix)
- setup_args process_command_line (int argc, char ∗argv[ ])
- spect_args spect_process_command_line (int argc, char ∗argv[ ])
- void process_command_line_help_arg (int argc, char ∗argv[ ], char help_id)
- void print_help (char code=0)
- void log_code_constants (std::string file_prefix)
- int extract_num_time_part (std::string name)
- std::pair< int, int > extract_space_part (std::string name)
- std::vector< std::string > process_filelist (int argc, char ∗argv[ ])
- void divide_domain (std::vector< size_t > dims, size_t space[2], int per_proc, int block_num)
- my_type get_ref_Bx (std::string file_prefix, size_t space_in[2], size_t time_0)
- data_array get_Bx (std::string file_prefix, size_t space_in[2], size_t time_0)
- bool flatten_fortran_slice (my_type ∗src_ptr, my_type ∗dest_ptr, size_t n_dims_in, size_t ∗dims_in, size_t flatten_on_dim, size_t flat_start=0, size_t flat_stop=-1)
- int where (const my_type ∗ax_ptr, int len, my_type target)
- calc_type gamma_rel (calc_type v)

5.20.1 Detailed Description

General helpers.

Contains MPI helpers, argument processing, and some general data handling functions

5.20.2 Function Documentation

5.20.2.1 void divide_domain (std::vector< size_t > *dims*, size_t *space[2]*, int *per_proc*, int *block_num*)

Divide dims evenly between procs

Uses the number of space blocks from args (if specified) and the domain size from dims to ensure perfect subdivision and set current proc's bounds. We can ignore incoming space vals as they should be -1

Parameters

	dims	Vector of sizes to divide
out	*space*	2-element array giving the local space section for this processor
	per_proc	Number of blocks per processor See setup_args per_proc field or see process_command_line() for example of calculation
	block_num	Current block index on this processor

5.20.2.2 int extract_num_time_part (std::string *name*)

Extract text from name

Assumes name is of the form [stuff]_ntimes_space0_space1.[extension] and extracts the integer ntimes

5.20.2.3 std::pair<int, int> extract_space_part (std::string *name*)

Extract text from name

Assumes name is of the form [stuff]_space0_space1.[extension] and extracts the two integers space0 and space1

5.20.2.4 bool flatten_fortran_slice (my_type * *src_ptr*, my_type * *dest_ptr*, size_t *n_dims_in*, size_t * *dims_in*, size_t *flatten_on_dim*, size_t *flat_start* = 0, size_t *flat_stop* = -1)

Flatten a Fortran-style array on the specified dimension

The result is a Fortran-style array of rank n_dims_in - 1, containing the total along each value of the flattening dim. dest_ptr is assumed to point to an allocated block sufficient to hold the result. NB this produces a total not an average. NB flat_start and flat_stop must be valid indices into the dim to be flattened. They CANNOT be checked. If supplied only the slice they delimit is totalled

Parameters

src_ptr	Pointer to start of source data
dest_ptr	Pointer to start of destination memory block
n_dims_in	Rank (number of dimensions) of input "array"
dims_in	Dimensions of input
flatten_on_dim	Dimension to flatten on
flat_start	Start of slice to flatten (default 0)
flat_stop	End of slice to flatten (default MAX_SIZE_T)

Returns

0 for success, 1 if parameters invalid (usually a range error)

5.20.2.5 calc_type gamma_rel (calc_type *v*) [inline]

Relativistic gamma from velocity

Parameters

v	Particle velocity

Returns

Corresponding relativistic gamma

5.20.2.6 data_array get_Bx (std::string *file_prefix*, size_t *space_in[2]*, size_t *time_0*)

Read reference B_x from file at path file_prefix, dump number time_0. If space_in is not [-1, -1], only the slice it dictates is read

Parameters

file_prefix	File path
space_in	Limits on x-dimension to slice out
time_0	The dump time to read

Returns

data_array containing bx data

Extension Add 3-D space handling!

5.20.2.7 void get_deck_constants (std::string *file_prefix*)

Setup run specific constants

Reads deck.status and parses values for user defined constants etc. It will rely on using the specific deck, because it has to look for names. Any changes to deck may need updating here. Tag names are set as const strings in support.h. IMPORTANT: If we find additional density tags we fold those into om_pe. To prevent this, either remove from deck.status, or prefix their printed names with something so they do not match the strings in support.h

Parameters

file_prefix	File prefix prepended to "deck.status"

5.20.2.8 my_type get_ref_Bx (std::string *file_prefix,* size_t *space_in[2],* size_t *time_0*)

Read reference B_x from the specfied file prefix as given, dump number time_0

Parameters

file_prefix	File path
space_in	Limits on x-dimension to slice out
time_0	The dump time to read

Returns

Average bx over specified space range at given time

5.20.2.9 int local_MPI_setup (int *argc,* char * *argv[ ]*)

Do the MPI init

Calls MPI init and sets up communicator. Stores number of processors and each rank into mpi_info.

5.20.2.10 void log_code_constants (std::string *file_prefix*)

Log internal constants

Records ID codes etc as name value pairs

Parameters

file_prefix	File path to write to

5.20.2.11 void print_help (char *code* = 0)

Print command line help

Prints contents of halp_file from rank zero and calls safe exit.

Parameters

code	Input single character utility name code to get specific help. If non-empty file opened is halp_file with '_'+code inserted before extension

5.20.2.12 setup_args process_command_line (int *argc,* char * *argv[ ]*)

Set basic parameters

Sets defaults or those given via command line (see help.txt). Forces an constant integer number of space blocks on each core.

5.20.2.13 void process_command_line_help_arg (int *argc,* char * *argv[ ],* char *help_id*)

Handle help request at command line

If -h is supplied anywhere in arg list, print the appropriate help file and then exit

5.20.2.14 std::vector<std::string> process_filelist (int *argc,* char * *argv[]*)

Extracts files from list

Returns vector of filename strings. If one argument is supplied it is used. If multiple, they're assumed to be in form [stuff]_space0_space1.dat and will be ordered on space0

Todo Write sort rather than using map shortcut

5.20.2.15 void safe_exit ( )

Exit program

Does minimal cleanup and exits

5.20.2.16 void share_consts ( )

MPI Share deck constants

Share the deck constants read on root to all procs

5.20.2.17 spect_args spect_process_command_line (int *argc,* char * *argv[]*)

Process command line args for spectra

Process the spectrum specific args, setting those consumed to HANDLED_ARG

5.20.2.18 int where (const my_type * *ax_ptr,* int *len,* my_type *target*)

Find where ax_ptr exceeds target

Checks bounds and calls locally scoped recursive whereb to do the search. Special case if target equals bottom end

Parameters

ax_ptr	Pointer to start of axis to search
len	Length of axis we're searching
target	Target value to find

Returns

Index of target in axis

5.21 Global Helper and Maths Functions

String handling, IO and maths helpers.

Modules

- Auxilliary functions
  Named free functions.

Functions

- std::string read_wipps_version_string (std::string filename)
- bool check_wipps_version (std::string filename)
- void my_print (std::string text, int rank, int rank_to_write=0, bool noreturn=false)
- void my_print (std::fstream *handle, std::string text, int rank, int rank_to_write=0, bool noreturn=false)
- void my_error_print (std::string text, int rank, int rank_to_write=0, bool noreturn=false)
- void my_error_print (std::fstream *handle, std::string text, int rank, int rank_to_write=0, bool noreturn=false)
- std::string mk_str (int i)
- std::string mk_str (size_t i)
- std::string mk_str (bool b)
- std::string mk_str (double i, bool noexp=0)
- std::string mk_str (float i, bool noexp=0)
- std::string mk_str (long double i, bool noexp=0)
- std::string mk_str (char *str)
- void trim_string (std::string &str, char ch=' ')
- long checked_strtol (const char *str, bool quiet=false)
- float checked_strtof (const char *str, bool quiet=false)
- std::string replace_char (std::string str_in, char ch, char repl)
- std::string append_into_string (const std::string &in, const std::string &infix)
- bool parse_name_val (std::string in, std::string &name, std::string &val)
- std::string str_to_upper (std::string str)
- std::string str_to_lower (std::string str)
- int compare_as_version_string (std::string str, std::string vers_str=VERSION, bool minor=false)
- template<typename T >
  T integrator (T *start, int len, T *increment)
- template<typename T >
  void inplace_boxcar_smooth (T *start, int len, int width, bool periodic=0)
- calc_type square_integrator (calc_type *start, int len, calc_type *increment)
- std::vector< calc_type > cubic_solve (calc_type an, calc_type bn, calc_type cn)
- template<typename T >
  T interpolate_linear (T axis[2], T vals[2], T target)
- template<typename T >
  T interpolate_nearest (T axis[2], T vals[2], T target)

5.21.1 Detailed Description

String handling, IO and maths helpers.

5.21.2 Function Documentation

5.21.2.1 std::string append_into_string (const std::string & *in*, const std::string & *infix*)

Insert infix in string

Inserts the infix string into in BEFORE the last file extension. If no '.' is found in string, append to end. First char being . is not an extension.

Parameters

in	String to insert into. Usually ends in .[extension]
infix	String to insert

Returns

The resulting modified string

5.21.2.2 bool check_wipps_version (std::string *filename*)

Check wipps versioning

Reads version from specified file and compares, printing error and returning result.

Parameters

filename	File to read

Returns

1 if match, 0 else

5.21.2.3 float checked_strtof (const char * *str,* bool *quiet* = `false`)

Convert c-string to float

Converts c_str to float and reports out-of-range or erroneous characters

5.21.2.4 long checked_strtol (const char * *str,* bool *quiet* = `false`)

Convert c-string to long

Converts c_str to long and reports out-of-range or erroneous characters

5.21.2.5 int compare_as_version_string (std::string *str,* std::string *vers_str* = `VERSION`, bool *minor* = `false`)

Check str against version code

Checks string against git version code VERSION. Version strings are vx.y if present at all. By default check just the x, if minor is true, check y also. If either string doesn't match expected format we try comparing as just strings. This maintains behaviour from before I used version tags where just commit-id was checked

Parameters

str	String to check
vers_str	String to check against, defaults to VERSION
minor	Flag to check minor version number too

Returns

0 if equal, -1 if str is before vers_str, 1 if str is after vers_str unless strings don't have numeric version parts in which case return 0 for equal, 1 for unequal

5.21.2.6 std::vector<calc_type> cubic_solve (calc_type *an,* calc_type *bn,* calc_type *cn*)

Finds roots of cubic x^3 + an x^2 + bn x + cn = 0

Uses Num. Rec. equations, which are optimised for precision. Note that if x >>1 precision errors may result. Returns real solutions only

Parameters

an	Coefficient of x^2
bn	Coefficient of x
cn	Constant part

Returns

Vector of real roots

5.21.2.7 template<typename T > void inplace_boxcar_smooth (T * *start,* int *len,* int *width,* bool *periodic* = 0)

Boxcar smoothing of specified width

Smooths the array given by start and len using specified width. If periodic is set the ends wrap around. Otherwise they one-side

Parameters

start	Start of data interval to smooth
len	Length of interval to smooth
width	Boxcar smoothing width
periodic	Flag to set for wrap-around smoothing

5.21.2.8 template<typename T > T integrator (T * *start,* int *len,* T * *increment*)

Basic numerical integrator

Uses trapezium rule. WARNING this is working with contiguous memory.

Parameters

start	Pointer to start of data
len	Length of data
increment	Pointer to start of increment axis (e.g. x[1:end]-x[0:end-1])

Returns

The integrated value

5.21.2.9 template<typename T > T interpolate_linear (T *axis[2],* T *vals[2],* T *target*)

Interpolate vals on axis to target value.

Axis and vals should contain 2 values boxing the target. We use linear interpolation to obtain the axis value corresponding to target

Parameters

axis	Axis values for interpolation
vals	Values at axis values
target	Target value to interpolate to

Returns

The interpolated value

5.21.2.10 template<typename T > T interpolate_nearest (T *axis[2],* T *vals[2],* T *target*)

Interpolate vals on axis to target value.

Axis and vals should contain 2 values boxing the target. We select the nearest axis value as corresponding to target

Parameters

axis	Axis values for interpolation
vals	Values at axis values
target	Target value to interpolate to

Returns

The interpolated value

5.21.2.11 std::string mk_str (int *i*)

Converts int to string

5.21.2.12 std::string mk_str (size_t *i*)

Long int to string

5.21.2.13 std::string mk_str (bool *b*)

Converts bool to string

5.21.2.14 std::string mk_str (double *i*, bool *noexp* = 0)

Converts double to string

5.21.2.15 std::string mk_str (float *i*, bool *noexp* = 0)

Converts float to string

5.21.2.16 std::string mk_str (long double *i*, bool *noexp* = 0)

Converts long double to string

5.21.2.17 std::string mk_str (char * *str*)

Convert C string to std::string

5.21.2.18 void my_error_print (std::string *text*, int *rank*, int *rank_to_write* = 0, bool *noreturn* = false)

Write output

MPI aware screen error output. Prints from one or all processors to cerr

Parameters

text	Text to print
rank	Rank of this processor
rank_to_write	Which rank should do the printing, default 0. Set to -1 to print from all
noreturn	Set to not output a line break after text

5.21.2.19 void my_error_print (std::fstream * *handle*, std::string *text*, int *rank*, int *rank_to_write* = 0, bool *noreturn* = false)

Write output

MPI aware filestream error output. Prints from one or all processors to given filestream. If this is nullptr, cerr is used

Parameters

handle	Filestream to write to. If this is nullptr, cout is used
text	Text to print
rank	Rank of this processor
rank_to_write	Which rank should do the printing, default 0. Set to -1 to print from all
noreturn	Set to not output a line break after text

5.21.2.20 void my_print (std::string *text*, int *rank*, int *rank_to_write* = `0`, bool *noreturn* = `false`)

Write output

MPI aware screen output. Prints from one or all processors to cout

Parameters

text	Text to print
rank	Rank of this processor
rank_to_write	Which rank should do the printing, default 0. Set to -1 to print from all
noreturn	Set to not output a line break after text

Todo Check what happens when multi-cores print....

5.21.2.21 void my_print (std::fstream * *handle*, std::string *text*, int *rank*, int *rank_to_write* = `0`, bool *noreturn* = `false`)

Write output

MPI aware file output. Prints from one or all processors to given filestream

Parameters

handle	Filestream to write to. If this is nullptr, cout is used
text	Text to print
rank	Rank of this processor
rank_to_write	Which rank should do the printing, default 0. Set to -1 to print from all
noreturn	Set to not output a line break after text

5.21.2.22 bool parse_name_val (std::string *in*, std::string & *name*, std::string & *val*)

Parse x=y strings

Basic line parser. Takes a string and if it contains an '=' splits into the left and right segments, stripping leading and trailing spaces. Returns 0 if success, 1 if no equals sign. Standard comment character is # as first non-whitespace

Parameters

	in	The input string to parse
`out`	*name*	The name part
`out`	*val*	The value part

Returns

0 if line parsed, 1 if it can't be

5.21.2.23 std::string read_wipps_version_string (std::string *filename*)

Read code version from file

Reads the version string of code used to write the file given by (full path) filename

Parameters

filename	File to read

Returns

String containing code version

5.21.2.24 std::string replace_char (std::string *str_in,* char *ch,* char *repl*)

Replace all occurences of character ch in string Replace character in string

Replace one character with another in a string

Parameters

str_in	Input string
ch	Character to replace
repl	Character to replace with

Returns

The amended string

5.21.2.25 calc_type square_integrator (calc_type * *start,* int *len,* calc_type * *increment*)

Basic numerical integrator

Uses trapezium rule. WARNING this is working with contiguous memory.

Parameters

start	Pointer to start of data
len	Length of data
increment	Pointer to start of increment axis (e.g. x[1:end]-x[0:end-1])

Returns

The square-integrated value

5.21.2.26 std::string str_to_lower (std::string *str*) [inline]

Convert string to lower case

5.21.2.27 std::string str_to_upper (std::string *str*) [inline]

Convert string to upper case

5.21.2.28 void trim_string (std::string & *str,* char *ch* = ' ')

Trim all leading/trailing ch's from str Trim ch from ends of string

Trims leading and trailing occurences of character from string

Parameters

str	The string (will be changed)

ch	The character to trim

5.22 Auxilliary functions

Named free functions.

Functions

- my_type subtract (my_type lhs, my_type rhs)
- my_type add (my_type lhs, my_type rhs)
- my_type divide (my_type lhs, my_type rhs)
- my_type multiply (my_type lhs, my_type rhs)

5.22.1 Detailed Description

Named free functions.

These can be used with the various array apply functions to do arithmetic on arrays without having to create lambdas

5.22.2 Function Documentation

5.22.2.1 my_type add (my_type *lhs,* my_type *rhs*) [inline]

Element-wise addition

5.22.2.2 my_type divide (my_type *lhs,* my_type *rhs*) [inline]

Element-wise division

5.22.2.3 my_type multiply (my_type *lhs,* my_type *rhs*) [inline]

Element-wise multiplication

5.22.2.4 my_type subtract (my_type *lhs,* my_type *rhs*) [inline]

Element-wise subtraction

6 Class Documentation

6.1 bounce_av_data Class Reference

Bounce-averaging data.

`#include <controller.h>`

Public Member Functions

- void set_Bx_size (size_t len)

  Set size of Bx.
- bool set_Bx (data_array Bx_in)

  Set value of Bx.
- my_type get_Bx_at (my_type lat)

  Get Bx value.

Public Attributes

- bounce_av_type type {plain}
- my_type L_shell {4.0}
- my_type max_latitude {90.0}

6.1.1 Detailed Description

Bounce-averaging data.

Holds the needed stuff for bounce-averaging. Controller should only access B_x via the accessor, which returns the value from the B array. NB NB we don't currently use this B_x, rather an assumed dipole.

Extension Actually use this B_x rather than assuming a dipole

6.1.2 Member Function Documentation

6.1.2.1 my_type bounce_av_data::get_Bx_at (my_type *lat*) [inline]

Get Bx value.

Get the value of Bx

Parameters

lat	The latitude to get value at in radians

Returns

The value at given latitude

6.1.2.2 bool bounce_av_data::set_Bx (data_array *Bx_in*) [inline]

Set value of Bx.

Sets the Bx array to (a copy of) Bx_in

Parameters

Bx_in	Input array, must match current size of Bx

Returns

0 for success, 1 if problem

6.1.2.3 void bounce_av_data::set_Bx_size (size_t *len*) [inline]

Set size of Bx.

Resizes the Bx array and updates the stored length

Parameters

len	The new length

6.1.3 Member Data Documentation

6.1.3.1 my_type bounce_av_data::L_shell {4.0}

The L shell we're at

6.1.3.2 my_type bounce_av_data::max_latitude {90.0}

Maximum latitude of "field line" in degrees

6.1.3.3 bounce_av_type bounce_av_data::type {plain}

The specification for how to bounce average, a value from bounce_av_type

6.2 controller Class Reference

Controls plasma, spectrum and d_coeff objects and their connections.

`#include <controller.h>`

Public Member Functions

- controller (std::string file_prefix)
- **controller** (const controller &src)=delete
- **controller** (const controller &&src)=delete
- void clear_all ()
- ~controller ()
- bool is_good ()
- void set_plasma_B0 (my_type Bx_ref)
- bool add_spectrum (std::string file)
- bool add_spectrum (int n_om, int n_ang, bool separable)
- bool add_d (int n_v, int n_angs)
- void add_d_special (int n_v, int n_angs)
- void delete_current_spectrum ()
- spectrum * get_current_spectrum ()
- spectrum * get_spectrum_by_num (size_t indx)
- diffusion_coeff * get_current_d ()
- diffusion_coeff * get_d_by_num (size_t indx)
- diffusion_coeff * get_special_d ()

- const plasma & get_plasma ()
- void bounce_average (bounce_av_data bounce_dat)
- void handle_d_mpi ()
- bool save_spectra (std::string pref)
- bool save_D (std::string pref)

6.2.1 Detailed Description

Controls plasma, spectrum and d_coeff objects and their connections.

This is the public facing class controlling plasma, spectrum and d_coeff objects. It makes sure there is a plasma to provide needed functions for the latters and keeps each D_coeff attached to the spectrum used to generate it. Should also be responsible for supplying D's in order to whatever does the bounce-averaging. Because a spectrum is meaningless without a plasma, and a diffusion coefficient meaningless without both a plasma and a spectrum, the controller class is the only thing allowed to create or destroy spectra and diffusion coefficients. Plasma's have other purposes so are not restricted in this way. When a new spectrum is created, the get_current_spectrum is set to refer to it. Any subsequent add_d operation will update the D linked to this spectrum. Note: currently controllers aren't fully implemented as objects, so work with them as pointers.

Author

Heather Ratcliffe

Date

19/11/2015

Extension Consider adding copy and move constructors etc

6.2.2 Constructor & Destructor Documentation

6.2.2.1 controller::controller (std::string *file_prefix*) [explicit]

Setup

Create plasma object and initialise. A controller without a plasma is not meaningful. Plasma guarantees to be valid after construction, but may contain defaults if the specified file was not found.

Parameters

file_prefix	The file prefix to prepend to all file reads

6.2.2.2 controller::~controller ( )

Delete

Destroys all the D and spectrum objects

6.2.3 Member Function Documentation

6.2.3.1 bool controller::add_d (int *n_v*, int *n_angs*)

Create and add diffusion_coefficient

Creates a diffusion coefficient of size n_v x n_angs, paired with the last spectrum that was added.

Parameters

n_v	Size of velocity dimension of D
n_angs	Size of angle dimension of D

Returns

0 for success, 1 for failure

6.2.3.2 void controller::add_d_special (int *n_v*, int *n_angs*)

Create and add special diffusion_coefficient

Creates a diffusion coefficient of size nx x n_angs. Special coefficients do not have a matching spectrum. We use them mainly to hold results of bounce-averaging.

Parameters

n_v	Size of velocity dimension of D
n_angs	Size of angle dimension of D

6.2.3.3 bool controller::add_spectrum (std::string *file*)

Add spectrum from file

Add spectrum read from a file, created using e.g. data_array::write_to_file or write_data in IDL

Parameters

file	The full path to file to read

Returns

0 for success, 1 for failure

Todo Perhaps this should append the file_prefix to file??

6.2.3.4 bool controller::add_spectrum (int *n_om*, int *n_ang*, bool *separable*)

Create and add spectrum

Create a spectrum of the specified size and adds to list. See spectrum::spectrum(int n_om, int n_ang, bool separable) for details.

Parameters

n_om	Size of omega dimension of spectrum
n_ang	Size of angle dimension of spectrum
separable	Flag determining if spectrum is separable, see spectrum class

Returns

0 for success, 1 for failure

6.2.3.5 void controller::bounce_average (bounce_av_data *bounce_dat*)

Bounce average D

Assumes the list contains D in order across space and performs bounce average to create special D. We use bounce_data to inform the field shape etc etc. The end result on each processor should be something which just has to be plain-summed by the mpi part. Calls controller::handle_d_mpi() and various bounce helpers

Parameters

bounce_dat	The bounce averaging info such as the type info

6.2.3.6 void controller::clear_all ( )

Clear all the derived objects

Clears the spectrum-D list and the special D list (bounce averaged entries)

6.2.3.7 void controller::delete_current_spectrum ( )

Delete the current spectrum object

Deletes the current (last added) spectrum and any attached D_coeff

6.2.3.8 diffusion_coeff * controller::get_current_d ( )

Return current D

Get the current (i.e. the latest added) d_coeff

Returns

Pointer to the diffusion_coeff object, nullptr if list empty

6.2.3.9 spectrum * controller::get_current_spectrum ( )

Return current spectrum

Get the current (i.e. the latest added) spectrum

Returns

Pointer to the spectrum object, nullptr if list empty

6.2.3.10 diffusion_coeff * controller::get_d_by_num (size_t *indx*)

Return D by indx

Get the d_coeff indx ago

Parameters

indx	The index (counting backwards from the latest) of D to return

Returns

Pointer to the D object, nullptr if list empty or indx out of range

6.2.3.11 const plasma& controller::get_plasma ( ) [inline]

Get reference to the plasma object to use

Returns

Reference to the plasma object

6.2.3.12 diffusion_coeff * controller::get_special_d ( )

Return current special D

Get the current (i.e. the latest added) special d_coeff

Returns

Pointer to the diffusion_coeff object, nullptr if list empty

6.2.3.13 spectrum * controller::get_spectrum_by_num (size_t *indx*)

Return spectrum by indx

Get the spectrum indx ago.

Parameters

indx	The index (counting backwards from the latest) of spectrum to return

Returns

Pointer to the spectrum object, nullptr if list empty or indx out of range

6.2.3.14 void controller::handle_d_mpi ( )

MPI reduce diffusion coeffs

Creates an MPI Summed d on the root node

6.2.3.15 bool controller::is_good ( ) [inline]

Whether controller is fully setup

Returns

Boolean true for good state, false else

6.2.3.16 bool controller::save_D (std::string *pref*)

Save D's to files (one per chunk)

Writes each D object to a file, identified by space range and time. Note root will also write a bounce averaged file

Parameters

pref	File prefix, including path etc

Returns

0 for success, 1 for failure

6.2.3.17 bool controller::save_spectra (std::string *pref*)

Save spectra to files (one per chunk)

Writes each spectrum object to a file, identified by space range and time.

Parameters

pref	File prefix, including path etc

Returns

0 for success, 1 for failure

6.2.3.18 void controller::set_plasma_B0 (my_type *Bx_ref*) [inline]

Set the reference B field used by plasma, and thus the local om_ce value

Parameters

Bx_ref	The value to set

6.3 cutout_args Struct Reference

Command line arguments for FFT cutout utility.

Public Attributes

- std::string file_in
- std::string file_out
- std::string file_prefix
- std::vector< my_type > limits

6.3.1 Detailed Description

Command line arguments for FFT cutout utility.

6.3.2 Member Data Documentation

6.3.2.1 std::string cutout_args::file_in

Input FFT file

6.3.2.2 std::string cutout_args::file_out

Output FFT file

6.3.2.3 std::string cutout_args::file_prefix

File path prepended to all filenames

6.3.2.4 std::vector<my_type> cutout_args::limits

Limits to use for cutout

6.4 d_report Struct Reference

D coefficient report.

`#include <support.h>`

Public Attributes

- bool error
- size_t n_av
- size_t n_max
- size_t n_min
- bool single_n

6.4.1 Detailed Description

D coefficient report.

Contains information on D calculation such as resonances used

6.4.2 Member Data Documentation

6.4.2.1 bool d_report::error

Whether IO or setup errors occured

6.4.2.2 size_t d_report::n_av

Average n_max used for calc

6.4.2.3 size_t d_report::n_max

Max n_max used in calcs

6.4.2.4 size_t d_report::n_min

Min n_min used in calcs. Not '-' is omitted

6.4.2.5 bool d_report::single_n

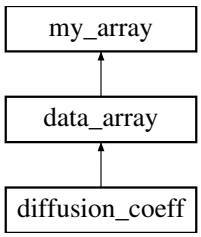
Flag showing that a single resonance, n_av, was used

6.5 data_array Class Reference

Extended my_array class including axes.

`#include <data_array.h>`

Inheritance diagram for data_array:



Public Member Functions

- data_array ()
- data_array (size_t nx, size_t ny=0, size_t nz=0, size_t nt=0)
- data_array (size_t n_dims, size_t *dims)
- data_array (std::string filename)
- virtual ~data_array ()
- virtual bool is_good () const
- data_array (const data_array &src)
- data_array (data_array &&src)
- data_array & operator= (const data_array &src)
- bool operator== (const data_array &rhs) const
- bool operator!= (const data_array &rhs) const

- data_array (const my_array &src)
- data_array & operator= (const my_array &src)
- bool operator== (const my_array &rhs) const
- bool operator!= (const my_array &rhs) const
- my_type * disown_axes ()
- void clone_empty (const data_array &src)
- void copy_ids (const data_array &src)
- bool check_ids (const data_array &src) const
- my_type get_axis_element (size_t dim, size_t pt) const
- bool set_axis_element (size_t dim, size_t pt, my_type val)
- const my_type * get_axis (size_t dim, size_t &length)
- float get_res (size_t i) const
- long get_axis_index_from_value (size_t dim, my_type value) const
- bool populate_axis (size_t dim, my_type *dat_in, size_t n_tot)
- void make_linear_axis (size_t dim, float res, long offset=0)
- bool write_to_file (std::fstream &file, bool close_file=true)
- bool read_from_file (std::fstream &file)
- bool write_section_to_file (std::fstream &file, std::vector< my_type > limits, bool close_file=true)
- bool write_raw_section_to_file (std::fstream &file, std::vector< size_t > index_limits, bool close_file=true)
- bool write_closer (std::fstream &file)
- bool resize (size_t dim, size_t sz, bool verbose=0)
- bool shift (size_t dim, long n_els, bool axis=1)
- data_array total (size_t dim)
- data_array total (size_t dim, my_type min, my_type max)
- data_array total (size_t dim, size_t min_ind, size_t max_ind)
- data_array average (size_t dim, my_type min, my_type max)
- data_array average (size_t dim)
- data_array average (size_t dim, size_t min, size_t max)

Public Attributes

- char block_id [ID_SIZE]
- my_type time [2]
- size_t space [2]
- my_type B_ref

Protected Member Functions

- virtual void construct ()
- void alloc_ax (const size_t els)
- size_t get_total_axis_elements () const
- long get_axis_index (size_t dim, size_t pt) const
- std::vector< size_t > get_bounds (std::vector< my_type > limits)

Protected Attributes

- my_type * axes

6.5.1 Detailed Description

Extended my_array class including axes.

Extends the my_array class to add axes and some data tags describing the data set derived from.

Move from file my_array by

Author

Heather Ratcliffe

Date

3/08/2016

6.5.2 Constructor & Destructor Documentation

6.5.2.1 data_array::data_array ( ) [explicit]

Default constructor

Create an empty, dimensionless array

6.5.2.2 data_array::data_array (size_t *nx*, size_t *ny* = 0, size_t *nz* = 0, size_t *nt* = 0) [explicit]

Construct a 1-4 D array

Adds axes to a normal rectangular my_array of correct size

Parameters

nx	Size of x dimension
ny	Size of y dimension, default 0
nz	Size of z dimension, default 0
nt	Size of t dimension, default 0

6.5.2.3 data_array::data_array (size_t *n_dims*, size_t * *dims*) [explicit]

Construct arbitrary dimension array

Adds axes to a normal rectangular my_array of correct size

Parameters

n_dims	Rank of array to create
dims	Array of dimensions of array to create

6.5.2.4 data_array::data_array (std::string *filename*) [explicit]

Create data array from file

Create a data array by reading from the named file. If the file does not exist nothing is done and this will be an empty array. Otherwise it reads the dimensions, sets up sizes and populates data and info

Parameters

filename	Full path of file to read from

6.5.2.5 data_array::~data_array ( ) [virtual]

Destructor

Free axis memory

6.5.2.6 data_array::data_array (const data_array & *src*)

Copy constructor

Copy src to a new instance, making a duplicate of data

Parameters

src	Array to copy

6.5.2.7 data_array::data_array (data_array && *src*)

Move constructor

Move src to new location. Copies data pointers but does not reallocate memory. Src is left empty

Parameters

src	Array to copy

6.5.2.8 data_array::data_array (const my_array & *src*)

Conversion operator

Convert a my_array into a data array. Data is deep copied, and axes are added

Parameters

src	My_array object to copy data and sizes from

6.5.3 Member Function Documentation

6.5.3.1 void data_array::alloc_ax (const size_t *els*) [protected]

Allocate axis memory

Alocate memory for axes

Parameters

els	Number of elements to allocate

6.5.3.2 data_array data_array::average (size_t *dim*, my_type *min*, my_type *max*)

Average array over dim dim

We guarantee to match total's behaviour if dim is out of range.

Parameters

dim	Dimension to average over
min	Minimum axis value emcompassing desired slice
max	Maximum axis value emcompassing desired slice

Returns

New array of rank n_dims-1 filled with the average values over the given dim.

6.5.3.3 data_array data_array::average (size_t *dim*)

Average array over dim dim

We guarantee to match total's behaviour if dim is out of range.

Parameters

dim	Dimension to average over

Returns

New array of rank n_dims-1 filled with the average values over the given dim.

6.5.3.4 data_array data_array::average (size_t *dim*, size_t *min*, size_t *max*)

Average array over dim dim

We guarantee to match total's behaviour if dim is out of range.

Parameters

dim	Dimension to average over
min	Minimum axis index emcompassing desired slice
max	Maximum axis index emcompassing desired slice

Returns

New array of rank n_dims-1 filled with the average values over the given dim.

6.5.3.5 bool data_array::check_ids (const data_array & *src*) const

Checks ID fields match src

Parameters

src	Array to check against

Returns

False for non-matching ids, true for complete match

6.5.3.6 void data_array::clone_empty (const data_array & *src*)

Initialise this to match sizes of src

This will be a valid empty array of size matching src.

Parameters

src	Array to copy from

6.5.3.7 void data_array::construct ( ) [protected], [virtual]

Common constructor logic

Sets fields for empty, dimensionless array

Reimplemented from my_array.

6.5.3.8 void data_array::copy_ids (const data_array & *src*)

Copies ID fields from src array to this

Parameters

src	Array to copy from

6.5.3.9 my_type * data_array::disown_axes ( )

Disown and return axes pointer

Surrenders ownership of memory pointed to by axes NB if this pointer is not kept an manually freed, memory will leak. To aquire ownership of both the axes and data, use disown_axes and my_array::disown_data in any order, but note that after the latter, this will not be a valid array and getter/setter for data and axes will return nothing.

Returns

Pointer to the axis memory

6.5.3.10 const my_type * data_array::get_axis (size_t *dim*, size_t & *length*)

Get pointer to axis

Returns pointer to given axis and its length. NB do not muck with this pointer. It's provided for ease of using where but is a const my_type * for a reason

Parameters

	dim	Dimension of axis to get
out	*length*	Returns the length of axis

Returns

Pointer to start of axis, or nullptr if axis doesn't exist or requested dim is out of range

6.5.3.11 my_type data_array::get_axis_element (size_t *dim*, size_t *pt*) const

Get axis value

Parameters

dim	Dimension
pt	Point to get

Returns

value at pt in dimension dim if in range, else 0.0

6.5.3.12 long data_array::get_axis_index (size_t *dim*, size_t *pt*) const [protected]

Get index of axis element

Get the position in the backing data store of point pt on dimension dim. Takes care of all bounds checking and disposition in memory

Parameters

dim	Dimension of axis
pt	Index into required axis

Returns

Index into 1-D array of pt, or -1 for any sort of out-of-range issue

6.5.3.13 long data_array::get_axis_index_from_value (size_t *dim*, my_type *value*) const

Get the index on axis dim of value value

Parameters

dim	Dimension for lookup
value	Value to find

Returns

Index of value in axis for dimension dim

6.5.3.14 std::vector< size_t > data_array::get_bounds (std::vector< my_type > *limits*) [protected]

Convert axis values to indices

For a vector of bounds, 2 per dimension, convert the required axis bounds into index bounds

Parameters

limits	The real physical axis values

Returns

The corresponding index values

6.5.3.15 float data_array::get_res (size_t *i*) const

Get (linear) axis resolution

Return resolution of axis on dimension i. Takes the total length and divides by the number of elements to avoid rounding errors. If axis is not linear, this is meaningless.

Parameters

i	Dimension of axis

Returns

0.0 if the axis is undefined or 0 or 1 in length, otherwise the calculated linear resolution

6.5.3.16 size_t data_array::get_total_axis_elements ( ) const [protected]

Return total axes length

Returns

The number of elements in all axes combined

6.5.3.17 virtual bool data_array::is_good ( ) const [inline], [virtual]

Whether array is useable

Returns

Boolean true for good state, false for bad

Reimplemented from my_array.

6.5.3.18 void data_array::make_linear_axis (size_t *dim,* float *res,* long *offset* = 0)

Make an axis

Generates a linear axis for dimension dim, with resolution res, starting at value of - offset∗res That allows one to guarantee that 0 appears regardless of the number of cells.

Parameters

dim	Dimension to build axis for
res	Axis resolution
offset	Number of grid cells to shift downwards (leftwards) by

6.5.3.19 bool data_array::operator!= (const data_array & *rhs*) const [inline]

See data_array::operator==()

6.5.3.20 bool data_array::operator!= (const my_array & *rhs*) const [inline]

See data_array::operator==()

6.5.3.21 data_array & data_array::operator= (const data_array & *src*)

Copy assignment

Set this array equal to src by (deep) copying src including data

Parameters

src	Array to copy

Returns

Copy of array

6.5.3.22 data_array & data_array::operator= (const my_array & *src*)

Conversion equality operator

Convert a my_array into a data array. Data is deep copied, and axes are added

Parameters

src	My_array object to copy data and sizes from

Returns

A data array containing copy of data in src plus empty axes

6.5.3.23 bool data_array::operator== (const data_array & *rhs*) const

Equality operator

Check this is equal to rhs. Since copies are always deep, we check values, not data pointers

Parameters

rhs	Array to compare to

Returns

Boolean true if equal, false else

6.5.3.24 bool data_array::operator== (const my_array & *rhs*) const [inline]

Equality (size and data only) with a my_array

Parameters

rhs	Array to compare to

6.5.3.25 bool data_array::populate_axis (size_t *dim*, my_type * *dat_in*, size_t *n_tot*)

Fill axis from dat_in

Populates axis from dat_in. Number of elements copied will be the smaller of n_tot and size of dimension dim.

Parameters

dim	Dimension of axis to fill
dat_in	Pointer to start of data to copy
n_tot	Size of input data array

Returns

0 (success) 1 (error)

6.5.3.26 bool data_array::read_from_file (std::fstream & *file*)

Read data array from file

Reads data from file. This array should have already been created in the correct shape, otherwise we return an error. After the layout in my_array::write_to_file is added Next_block axes "Footer:" Next_block time[2] space[2] B↩ _ref Next_block Block_id If file is to be closed then we finish with Footer_start, i.e. the position of the start of "Footer" section If multiple arrays are written, we write each without closing and the close the file with a final Block_id tag

Parameters

file	Filestram to read from

Returns

0 (success), 1 (error)

6.5.3.27 bool data_array::resize (size_t *dim*, size_t *sz*, bool *verbose* = 0)

Resize my_array on the fly

If sz $<$ dims[dim] the first sz rows will be kept and the rest deleted. If sz $>$ dims[dim] the new elements will be added zero initialised. Similarly for axis elements. See my_array::resize() for more.

Parameters

dim	The dimension to resize
sz	The new size
verbose	Whether to print some info

Returns

0 (success), 1 (error)

6.5.3.28 bool data_array::set_axis_element (size_t *dim*, size_t *pt*, my_type *val*)

Sets axis element

Sets elements at pt on dimension dim,

Parameters

dim	Dimension
pt	Point to set
val	Value to set

Returns

1 if out of range, 0 else.

6.5.3.29 bool data_array::shift (size_t *dim*, long *n_els*, bool *axis* = `1`)

Shift array on dim dim by n_els

Shift is cyclical

Parameters

dim	Dimension to shift on
n_els	Number of elements to shift by
axis	Whether to shift the corresponding axis

Returns

0 (success), 1 (error)

6.5.3.30 data_array data_array::total (size_t *dim*)

Total array on dim dim

Wraps function data_array::total(size_t dim, size_t min, size_t max).

Parameters

dim	Dimension to total on

Returns

A new array of rank n_dims -1, containing data summed over entire range of dimension dim. If dim is out of range, empty array is returned

6.5.3.31 data_array data_array::total (size_t *dim*, my_type *min*, my_type *max*)

Total array on dim dim

Wraps function data_array::total(size_t dim, size_t min, size_t max).

Parameters

dim	Dimension to total on
min	Minimum physical axis value of slice to total. Note this expects a monotonic axis.
max	Maximum physical axis value of slice to total.

Returns

A new array of rank n_dims -1, containing data summed over entire range of dimension dim. If dim is out of range, empty array is returned

6.5.3.32 data_array data_array::total (size_t *dim*, size_t *min_ind*, size_t *max_ind*)

Total along dim dim

Parameters

dim	Dimension to total on
min_ind	Minimum axis index of slice to total.
max_ind	Maximum axis index of slice to total.

Returns

A new array of rank n_dims -1, containing data summed over entire range of dimension dim. If dim is out of range, empty array is returned. Note totalling a 1-d array gives a 1-element array

6.5.3.33 bool data_array::write_closer (std::fstream & *file*)

Write close tag of file

Writes the final footer into a file

Parameters

file	Filestream to write to

Returns

0 (success), 1 (error)

6.5.3.34 bool data_array::write_raw_section_to_file (std::fstream & *file*, std::vector< size_t > *index_limits*, bool *close_file* = `true`)

Write section of array to file

Write section defined by the limits to supplied file. If limits are out of range then the entire dimension is used. See data_array::write_to_file for file details

Parameters

file	Filestram to write to
index_limits	Vector of min and max axis indices for section. Must be 2 per dimension
close_file	Whether to write file final data

Returns

0 (success), 1 (error)

6.5.3.35 bool data_array::write_section_to_file (std::fstream & *file*, std::vector< my_type > *limits*, bool *close_file* = `true`)

Write array section to file

Write section between given AXIS values to file. To use one dimension entire supply values less/greater than min and max axis values.

Parameters

file	Filestram to write to
limits	Vector of min and max physical axis values for section. Must be 2 per dimension
close_file	Whether to write file final data

Returns

0 (success), 1 (error)

6.5.3.36 bool data_array::write_to_file (std::fstream & *file*, bool *close_file* = `true`)

Write data array to file

After the layout in my_array::write_to_file is added Next_block axes "Footer:" Next_block time[2] space[2] B_ref Next_block Block_id If file is to be closed then we finish with Footer_start, i.e. the position of the start of "Footer" section If multiple arrays are written, we write each without closing and the close the file with a final Block_id tag

Parameters

file	Filestram to write to
close_file	Whether to write file final data

Returns

0 (success), 1 (error)

Todo Test read/write with double, also IDL

6.5.4 Member Data Documentation

6.5.4.1 my_type* data_array::axes [protected]

Axes data

6.5.4.2 my_type data_array::B_ref

Reference average B field by location

6.5.4.3 char data_array::block_id[ID_SIZE]

ID describing data

6.5.4.4 size_t data_array::space[2]

Space range over which data are taken

6.5.4.5 my_type data_array::time[2]

Time range over which data are taken

6.6 deck_constants Struct Reference

Constants read from deck.

`#include <support.h>`

Public Attributes

- float v_t
- float omega_pe
- float omega_ce
- float omega_ci
- int ppc
- float dens_factor

6.6.1 Detailed Description

Constants read from deck.

Holds run parameters extracted from deck file such as temperature, reference frequencies etc

6.6.2 Member Data Documentation

6.6.2.1 float deck_constants::dens_factor

Ratio of total plasma density to background plasma density

6.6.2.2 float deck_constants::omega_ce

Electron cyclotron frequency (reference)

6.6.2.3 float deck_constants::omega_ci

Ion cyclotron frequency (reference)

6.6.2.4 float deck_constants::omega_pe

Plasma frequency (reference)

6.6.2.5 int deck_constants::ppc

Particles per cell used

6.6.2.6 float deck_constants::v_t

Electron thermal velocity

6.7 diff_cmd_line Struct Reference

Command line arguments for diffusion calculation utility.

Public Attributes

- std::string file_prefix
- size_t d [2]
- bool is_list
- bool is_spect
- size_t ref
- std::string ref_name
- std::vector< std::string > file_list
- int fuzz
- int smth
- size_t n_ang
- int wave
- int ang
- float ang_sd
- std::vector< float > ang_lims
- std::vector< float > om_lims

6.7.1 Detailed Description

Command line arguments for diffusion calculation utility.

6.7.2 Member Data Documentation

6.7.2.1 int diff_cmd_line::ang

Angular function type (can be FUNCTION_NULL)

6.7.2.2 std::vector<float> diff_cmd_line::ang_lims

Truncation limits for angle

6.7.2.3 float diff_cmd_line::ang_sd

Width for angular function (if applicable)

6.7.2.4 size_t diff_cmd_line::d[2]

Dimensions of D to produce

6.7.2.5 std::vector<std::string> diff_cmd_line::file_list

List of filenames to read

6.7.2.6 std::string diff_cmd_line::file_prefix

Prefix part of file names

6.7.2.7 int diff_cmd_line::fuzz

Fuzz for spectral cutout

6.7.2.8 bool diff_cmd_line::is_list

Use FFT or spectrum list input

6.7.2.9 bool diff_cmd_line::is_spect

Use spectrum list

6.7.2.10 size_t diff_cmd_line::n_ang

Number of angles for output spectrum

6.7.2.11 std::vector<float> diff_cmd_line::om_lims

Truncation limits for omega

6.7.2.12 size_t diff_cmd_line::ref

Reference sdf file number to get Bx info from

6.7.2.13 std::string diff_cmd_line::ref_name

Reference file name to use for Bx info

6.7.2.14 int diff_cmd_line::smth

Smoothing width for output spectrum

6.7.2.15 int diff_cmd_line::wave

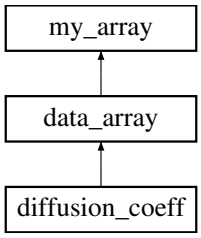
Wave type ID (see support.h WAVE_*)

6.8 diffusion_coeff Class Reference

Diffusion coefficient object.

`#include <d_coeff.h>`

Inheritance diagram for diffusion_coeff:



Public Member Functions

- void set_ids (float time1, float time2, int space1, int space2, int wave_id, char block_id[ID_SIZE])
- void set_single_n (int n)

  Set single resonance to consider.
- void set_max_n (int n)

  Set max/min resonant number to consider.
- bool write_to_file (std::fstream &file)
- bool read_from_file (std::fstream &file)
- d_report calculate (D_type_spec type_of_D=D_type_spec::alpha_alpha, bool quiet=0)
- my_type get_element_by_values (my_type p, my_type alpha)
- my_type get_axis_element_ang (size_t ind)
- my_type angle_to_stored_angle (my_type alpha)
- my_type stored_angle_to_angle (my_type value)

Public Attributes

- int latitude
- int wave_id
- std::string tag

Friends

- class **controller**

Additional Inherited Members

6.8.1 Detailed Description

Diffusion coefficient object.

Specialised data_array containing the calculated coefficient plus relevant ids. Can be made/destroyed only by controller object. In general should be a 2-D array with first axis momentum/velocity, second pitch-angle

Author

Heather Ratcliffe

Date

23/09/2015

6.8.2 Member Function Documentation

6.8.2.1 my_type diffusion_coeff::angle_to_stored_angle (my_type *alpha*) [inline]

Convert actual angle in radians to stored angle axis value

Parameters

alpha	Actual angle value

Returns

Converted value in axis

6.8.2.2 d_report diffusion_coeff::calculate (D_type_spec *type_of_D* = D_type_spec::alpha_alpha, bool *quiet* = 0)

Calculate D from wave spectrum and plasma

Uses the data available via my_controller to calculate D, the raw diffusion coefficient as function of particle velocity and pitch angle. For more details of the calculation see Derivations::Calculation_of_D Note that here we use a default number of wave-normal angle points, NOT the number present in the parent spectrum. See get_G2 for details of how interpolation is done.

Parameters

type_of_D	Which D to calculate (pitch angle, momentum, mixed)
quiet	True to suppress display of progress

Returns

d_report structure containing info on calcs

Caveat We re-range the D calculation so what is actually calculated is D/p^2

Caveat We currently use plasma::get_high_dens_phi_mu_om which uses the high-density approximation to the plasma dispersion, as does the resonant frequency solver plasma::get_resonant_omega. This missed some solutions for smaller om_pe/om_ce. Above 3 or so seems to be alright

Todo Check for double counting

6.8.2.3 my_type diffusion_coeff::get_axis_element_ang (size_t *ind*)

Get angle axs element

Returns an ANGLE value, rather than raw axis entry

Parameters

ind	Index of element wanted

Returns

Angle on axis at ind

6.8.2.4 my_type diffusion_coeff::get_element_by_values (my_type *p*, my_type *alpha*)

Lookup D element using axis values

Get D element by values of p and alpha by looking up those values in the axes and returning the nearest value on grid.

Parameters

p	Momentum value
alpha	Angle value

Returns

Diffusion coeff. value at location

Todo Q? try interpolating?

6.8.2.5 bool diffusion_coeff::read_from_file (std::fstream & *file*)

Read diffusion coeff from file

Reads file dump of a diffusion coefficient. D should have been created to the correct size already

Parameters

file	Filestream to read from

Returns

0 for success, 1 for failure (file access problem)

6.8.2.6 void diffusion_coeff::set_ids (float *time1,* float *time2,* int *space1,* int *space2,* int *wave_id,* char *block_id[ID_SIZE]*)

Set parameters

Sets the time and space ranges, wave type etc attached to the spectrum used to calculate this D. Times in seconds. Space in terms of grid points.

Parameters

time1	Initial time of data used
time2	End time of data used
space1	Start index of space range of data used
space2	End index of space range of data used
wave_id	Wave type (see support.h)
block_id	Name of block used to calculate this D

6.8.2.7 my_type diffusion_coeff::stored_angle_to_angle (my_type *value*) [inline]

Convert stored angle axis value to actual angle in radians

Parameters

value	Stored angle value

Returns

Actual angle value

6.8.2.8 bool diffusion_coeff::write_to_file (std::fstream & *file*)

Write diffusion coeff to file

Writes file dump of a diffusion coefficient. NB the file passed in must be opened for input and output.

Parameters

file	Filestream to write to

Returns

0 for success, 1 for failure (usually file access) /todo Write n_info or perhaps whole report?

6.8.3 Member Data Documentation

6.8.3.1 int diffusion_coeff::latitude

Latitude of calculation

6.8.3.2 std::string diffusion_coeff::tag

Identifies as local, averaged etc

6.8.3.3 int diffusion_coeff::wave_id

ID of wave mode considered

6.9 dist_cmd_line Struct Reference

Command line arguments for distribution utility.

Public Attributes

- bool list
- std::string file_prefix
- int dump
- int blocks

6.9.1 Detailed Description

Command line arguments for distribution utility.

6.9.2 Member Data Documentation

6.9.2.1 int dist_cmd_line::blocks

Number of space blocks to divide x dimension into

6.9.2.2 int dist_cmd_line::dump

Dump number to read

6.9.2.3 std::string dist_cmd_line::file_prefix

File path prepended to all files

6.9.2.4 bool dist_cmd_line::list

Flag to just list the available distribs

6.10 extractor_args Struct Reference

Command line arguments for field extractor utility.

Public Attributes

- std::string file_out
- int flat_dim

6.10.1 Detailed Description

Command line arguments for field extractor utility.

6.10.2 Member Data Documentation

6.10.2.1 std::string extractor_args::file_out

Output file

6.10.2.2 int extractor_args::flat_dim

Dimension to average on

6.11 fft_spect_args Struct Reference

Command line arguments for spectrum generation utility.

Public Attributes

- std::string file_prefix
- std::string file_in
- std::string file_out
- int fuzz
- int smth
- size_t n_ang
- float ang_sd
- int wave
- int ang
- bool mask

6.11.1 Detailed Description

Command line arguments for spectrum generation utility.

6.11.2 Member Data Documentation

6.11.2.1 int fft_spect_args::ang

Angular function type (can be FUNCTION_NULL)

6.11.2.2 float fft_spect_args::ang_sd

Width for angular function (if applicable)

6.11.2.3 std::string fft_spect_args::file_in

Input FFT filename

6.11.2.4 std::string fft_spect_args::file_out

Output spectrum filename (optional)

6.11.2.5 std::string fft_spect_args::file_prefix

Filepath prepended to all files

6.11.2.6 int fft_spect_args::fuzz

Fuzz for spectral cutout

6.11.2.7 bool fft_spect_args::mask

Flag to output spectrum extraction mask to file also

6.11.2.8 size_t fft_spect_args::n_ang

Number of angles for output spectrum

6.11.2.9 int fft_spect_args::smth

Smoothing width for output spectrum

6.11.2.10 int fft_spect_args::wave

Wave type ID (see support.h WAVE_*)

6.12 g_args Struct Reference

Additional command line arguments for growth calculation utility.

Public Attributes

- bool real
- std::string spect_file
- std::string outfile

6.12.1 Detailed Description

Additional command line arguments for growth calculation utility.

6.12.2 Member Data Documentation

6.12.2.1 std::string g_args::outfile

Output filename

6.12.2.2 bool g_args::real

Whether to derive real growth from spectra list too

6.12.2.3 std::string g_args::spect_file

File listing spectra if using this option

6.13 gen_cmd_line Struct Reference

Additional command line arguments for FFT generation utility.

Public Attributes

- int flat_dim
- bool flat_fft
- my_type flat_fft_min
- my_type flat_fft_max
- std::vector< my_type > limits

6.13.1 Detailed Description

Additional command line arguments for FFT generation utility.

6.13.2 Member Data Documentation

6.13.2.1 int gen_cmd_line::flat_dim

Flattening dimension number

6.13.2.2 bool gen_cmd_line::flat_fft

Flatten after FFT

6.13.2.3 my_type gen_cmd_line::flat_fft_max

Upper band limit for FFT flattening

6.13.2.4 my_type gen_cmd_line::flat_fft_min

Lower band limit for FFT flattening

6.13.2.5 std::vector<my_type> gen_cmd_line::limits

Limits to trim output FFT to

6.14 mpi_info_struc Struct Reference

MPI information.

`#include <support.h>`

Public Attributes

- int rank
- int n_procs

6.14.1 Detailed Description

MPI information.

Holds info on MPI: processor ranks etc

6.14.2 Member Data Documentation

6.14.2.1 int mpi_info_struc::n_procs

Global number of processors

6.14.2.2 int mpi_info_struc::rank

Rank of current processor

6.15 mu_dmudom Struct Reference

Reduced refractive index.

`#include <support.h>`

Public Attributes

- calc_type mu
- calc_type dmudom
- calc_type dmudtheta
- calc_type phi
- int err
- calc_type cone_ang

6.15.1 Detailed Description

Reduced refractive index.

Contains refractive index mu, the two derivative needed for diffusion coefficient calculation, the phi function of e.g. Albert [1] and error flag

6.15.2 Member Data Documentation

6.15.2.1 calc_type mu_dmudom::cone_ang

Resonance cone angle

6.15.2.2 calc_type mu_dmudom::dmudom

d mu / d omega (wave frequency)

6.15.2.3 calc_type mu_dmudom::dmudtheta

d mu / d theta (wave normal angle)

6.15.2.4 int mu_dmudom::err

0 if mu found successfully, 1 else

6.15.2.5 calc_type mu_dmudom::mu

Refractive index

6.15.2.6 calc_type mu_dmudom::phi

Phi from Albert [1]

6.16 my_args Struct Reference

Command line arguments for example utility.

Public Attributes

- std::string file_prefix
- int num_in
- std::vector< std::string > **file_list**
- std::string file_in

6.16.1 Detailed Description

Command line arguments for example utility.

6.16.2 Member Data Documentation

6.16.2.1 std::string my_args::file_in

Input file

6.16.2.2 std::string my_args::file_prefix

Prefix part of file names (path plus any common prefix)

6.16.2.3 int my_args::num_in

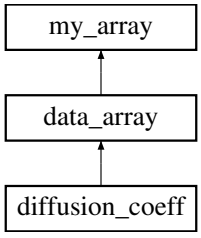
An integer

6.17 my_array Class Reference

A basic array class.

`#include <my_array.h>`

Inheritance diagram for my_array:



Public Member Functions

- my_array ()
- my_array (size_t nx, size_t ny=0, size_t nz=0, size_t nt=0)
- my_array (size_t n_dims, size_t *dims)
- virtual ~my_array ()
- virtual bool is_good () const
- my_array (const my_array &src)
- my_array (my_array &&src)
- my_array & operator= (const my_array &src)
- bool operator== (const my_array &rhs) const
- bool operator!= (const my_array &rhs) const
- my_type * disown_data ()
- void clone_empty (const my_array &src)
- bool copy_data (my_type *destination) const
- void zero_data ()
- size_t get_dims () const
- size_t get_dims (size_t dim) const
- my_type get_element (size_t nx) const
- my_type get_element (size_t nx, size_t ny) const
- my_type get_element (size_t nx, size_t ny, size_t nz) const
- my_type get_element (size_t nx, size_t ny, size_t nz, size_t nt) const
- my_type get_element (size_t n_dims, size_t *dim) const
- size_t get_total_elements () const
- bool set_element (size_t nx, my_type val)
- bool set_element (size_t nx, size_t ny, my_type val)
- bool set_element (size_t nx, size_t ny, size_t nz, my_type val)
- bool set_element (size_t nx, size_t ny, size_t nz, size_t nt, my_type val)
- bool set_element (size_t n_dims, size_t *dim, my_type val)
- template<typename T >
  bool populate_data (T dat_in, size_t n_tot)
- bool populate_slice (my_type *dat_in, size_t n_dims, size_t *offsets)
- bool populate_complex_slice (my_type *dat_in, size_t n_dims, size_t *offsets, size_t *sizes)
- bool write_to_file (std::fstream &file)
- bool read_from_file (std::fstream &file)
- std::vector< size_t > read_dims_from_file (std::fstream &file)
- bool write_section_to_file (std::fstream &file, std::vector< size_t > bounds)
- bool resize (size_t dim, size_t sz, bool verbose=0)
- bool shift (size_t dim, long n_els)
- my_type minval (size_t offset=0)
- my_type maxval (size_t offset=0)
- my_type minval (std::vector< size_t > &ind, size_t offset=0)
- my_type maxval (std::vector< size_t > &ind, size_t offset=0)
- my_type partial_maxval (std::vector< std::pair< size_t, size_t > > ranges, std::vector< size_t > &ind)
- void smooth_1d (int n_pts)
- void apply (std::function< my_type(my_type arg)> func)
- void apply (std::function< my_type(my_type arg, my_type arg2)> func, my_type arg)
- void apply (std::function< my_type(my_type arg)> func, const my_array &rhs)
- void apply (std::function< my_type(my_type, my_type) > func, const my_array &rhs)

Protected Member Functions

- virtual void construct ()
- virtual void alloc_all (const size_t n_dims, const size_t *const dims)
- virtual std::vector< size_t > get_indices_from_offset (size_t offset) const
- virtual long get_index (size_t n_dims, size_t *dim) const
- long get_index (size_t nx) const
- long get_index (size_t nx, size_t ny) const
- long get_index (size_t nx, size_t ny, size_t nz) const
- long get_index (size_t nx, size_t ny, size_t nz, size_t nt) const
- my_type get_element_from_index (size_t ind) const

Protected Attributes

- size_t n_dims
- size_t * dims
- my_type * data

Friends

- bool populate_mirror_fastest (data_array &data_out, my_type *result_in, size_t total_els)

6.17.1 Detailed Description

A basic array class.

Contains dimension information and data. Can be rectangular of any n_dims or ragged of 2 (rows of different lengths). Get_index and get_total_elements account for all details of internal layout in memory. For 1-4 dims individual getter/setter functions are given. For larger arrays one must construc the array of indexes. NOTE the backing memory is old style with Fortran style internal ordering (for ease of SDF interfacing). But contigous memory and pointer arithmetic give major speed advantage and we very rarely change size on the fly. However nothing outside this class should need to do anything except access by index and populate by element, slice or entire. Internal ordering is Fortran style (for ease of SDF interfacing).

Author

Heather Ratcliffe

Date

21/09/2015

6.17.2 Constructor & Destructor Documentation

6.17.2.1 my_array::my_array ( ) [explicit]

Default constructor

6.17.2.2 my_array::my_array (size_t *nx*, size_t *ny* = 0, size_t *nz* = 0, size_t *nt* = 0) [explicit]

1 to 4 d rectangular array creator

Sets up a n-d rectangular array for n = 1 to 4. Helper avoids user having to construct size_t array of dims. Default values mean any dimensions not supplied will be 0

Parameters

nx	Size of x dimension
ny	Size of y dimension, default 0
nz	Size of z dimension, default 0
nt	Size of t dimension, default 0

6.17.2.3 my_array::my_array (size_t *n_dims,* size_t * *dims*) [explicit]

Arbitrary dim rectangular array

Sets up internals of array including memory allocation

Parameters

n_dims	Rank of array to create
dims	Array of dimensions of array to create

6.17.2.4 my_array::~my_array ( ) [virtual]

Destructor

Clean up explicit allocations

6.17.2.5 my_array::my_array (const my_array & *src*)

Copy constructor

Copy src to a new instance, making a duplicate of data

Parameters

src	Array to copy from

6.17.2.6 my_array::my_array (my_array && *src*)

Move constructor

Move src to a new instance i.e. copy fields but don't move memory. Src becomes empty afterwards

Parameters

src	Array to move

6.17.3 Member Function Documentation

6.17.3.1 void my_array::alloc_all (const size_t *n_dims,* const size_t *const *dims*) [protected], [virtual]

Takes care of memory allocation

Allocate dims array and data. If any size is zero, any exceeds MAX_SIZE, or overall exceeds MAX_SIZE_TOT, print error and stop. NB inputs will not be modified, will be copied

Parameters

n_dims	Rank of array to allocate
dims	Dimensions of array to allocate

6.17.3.2 void my_array::apply (std::function< my_type(my_type arg)> *func*) [inline]

Apply a function to each element of array. func must take and return a my_type or type convertible to this

Parameters

func	Function to apply

6.17.3.3 void my_array::apply (std::function< my_type(my_type arg, my_type arg2)> *func,* my_type *arg*) [inline]

Apply a function to each element of array. func must take two my_type and return one my_type or type convertible to this

Parameters

func	Function to apply
arg	Second argument to function

6.17.3.4 void my_array::apply (std::function< my_type(my_type arg)> *func,* const my_array & *rhs*) [inline]

Fill one array from another mapping elements using function. func must take and return a my_type or type convertible to this. Each element of the rhs is tranformed with func and placed into this array

Parameters

func	Transform function to apply to elements
rhs	Array to transform

6.17.3.5 void my_array::apply (std::function< my_type(my_type, my_type) > *func,* const my_array & *rhs*) [inline]

Transform one array using another. func must take a pair of my_type and return a my_type or types convertible to this. Func is called with each element of this array and each element of rhs and the result placed back into this.

Parameters

func	Function to apply
rhs	Second parameters to func

6.17.3.6 void my_array::clone_empty (const my_array & *src*)

Initialise this to same sizes as src

This will be a valid empty array of size matching src.

Parameters

src	Array to copy dims from

6.17.3.7 void my_array::construct ( ) [protected], [virtual]

Shared contructor code

Sets default values

Reimplemented in data_array.

6.17.3.8 bool my_array::copy_data (my_type * *destination*) const

Copy the data into destination array

Data is not lost, but a direct copy is made. Destination size will NOT be checked, the entirety of data is copied.

Parameters

out	*destination*	Pointer to destination to copy to

Returns

0, error checking not yet implemented

6.17.3.9 my_type * my_array::disown_data ( )

Disown and return data pointer

Surrenders ownership of memory pointed to by data, nullifies dimensions and returns pointer. NB if this pointer is not kept and manually freed, memory will leak. This array will then be an empty array

Returns

Pointer to the disowned data array

6.17.3.10 size_t my_array::get_dims ( ) const

Return rank of array

6.17.3.11 size_t my_array::get_dims (size_t *dim*) const

Return size of dimension dim

6.17.3.12 my_type my_array::get_element (size_t *nx*) const

Get element

Return element at nx. Out of range etc will return 0.0

6.17.3.13 my_type my_array::get_element (size_t *nx*, size_t *ny*) const

As my_array::get_element(size_t nx) const but for 2-D arrays

6.17.3.14 my_type my_array::get_element (size_t *nx*, size_t *ny*, size_t *nz*) const

As my_array::get_element(size_t nx) const but for 3-D arrays

6.17.3.15 my_type my_array::get_element (size_t *nx*, size_t *ny*, size_t *nz*, size_t *nt*) const

As my_array::get_element(size_t nx) const but for 4-D arrays

6.17.3.16 my_type my_array::get_element (size_t *n_dims*, size_t * *dim*) const

As my_array::get_element(size_t nx) const but for arbitrary dimension arrays. Supply n_dims and an array of the required indices

6.17.3.17 my_type my_array::get_element_from_index (size_t *ind*) const [protected]

Get element by 1-d offset

Returns the element at index in the 1-D backing array. Ind should be found using one of the get_index options

6.17.3.18 long my_array::get_index (size_t *n_dims*, size_t * *dim*) const [protected], [virtual]

Convert n-d index into 1-d index

This requires passing integer array and loop so is slower than the dedicated functions (get_index(nx, ...)) that follow

6.17.3.19 long my_array::get_index (size_t *nx*) const [protected]

Get index of element at nx

Takes care of all bounds checking and disposition in memory. Returns -1 if out of range of any sort, otherwise, the index into backing data array. This function is called often so we make it as simple as possible and write one for each number of args.

6.17.3.20 long my_array::get_index (size_t *nx,* size_t *ny*) const [protected]

See my_array::get_index(size_t nx) const

6.17.3.21 long my_array::get_index (size_t *nx,* size_t *ny,* size_t *nz*) const [protected]

See my_array::get_index(size_t nx) const

6.17.3.22 long my_array::get_index (size_t *nx,* size_t *ny,* size_t *nz,* size_t *nt*) const [protected]

See my_array::get_index(size_t nx) const

6.17.3.23 std::vector< size_t > my_array::get_indices_from_offset (size_t *offset*) const [protected], [virtual]

Get vector of indices from 1-d index

Takes a 1-d index and returns the n-d index vector. If offset is out of range, empty vector is returned.

6.17.3.24 size_t my_array::get_total_elements ( ) const

Return total size of array

Returns

The total number of data elements in array

6.17.3.25 virtual bool my_array::is_good ( ) const [inline], [virtual]

Check memory allocation etc worked

Returns

True if good, false else

Reimplemented in data_array.

6.17.3.26 my_type my_array::maxval (size_t *offset* = 0)

Find maximum value of data

Finds the maximum of the array after offset, using linear search through contiguous memory. Offset should be obtained using one of the get_index functions.

Parameters

offset	1-D array offset to start search from

Returns

The maximum value of data

6.17.3.27 my_type my_array::maxval (std::vector< size_t > & *ind,* size_t *offset* = 0)

Find maximum value of data

Finds the maximum of the array after offset, using linear search through contiguous memory. Offset should be obtained using one of the get_index functions.

Parameters

	offset	1-D array offset to start search from
out	*ind*	The indices where max value is located

Returns

The maximum value of data

6.17.3.28 my_type my_array::minval (size_t *offset* = 0)

Find minimum value of data

Finds the minimum of the array after offset, using linear search through contiguous memory. Offset should be obtained using one of the get_index functions.

Parameters

offset	1-D array offset to start search from

Returns

The minimum value of data

6.17.3.29 my_type my_array::minval (std::vector< size_t > & *ind,* size_t *offset* = 0)

Find minimum value of data

Finds the minimum of the array after offset, using linear search through contiguous memory. Offset should be obtained using one of the get_index functions.

Parameters

	offset	1-D array offset to start search from
out	*ind*	The indices where min value is located

Returns

The minimum value of data

6.17.3.30 bool my_array::operator!= (const my_array & *rhs*) const [inline]

See my_array::operator==()

6.17.3.31 my_array & my_array::operator= (const my_array & *src*)

Copy assignment

Sets this equal to a (deep) copy of source

Parameters

src	Array to copy from

Returns

Copy of input

6.17.3.32 bool my_array::operator== (const my_array & *rhs*) const

Equality operator

Check this is equal to rhs. Since copies are always deep, we check values, not data pointers

Parameters

rhs	Array to compare to

Returns

Boolean true if equal, false else

6.17.3.33 my_type my_array::partial_maxval (std::vector< std::pair< size_t, size_t > > *ranges*, std::vector< size_t > & *ind*)

Maximum value over part of range

WARNING: this is slow. Perhaps very slow. I'm using routines I have to knock it up quickly. Beware!!!

Parameters

	ranges	The indices to consider between on each dimension
out	*ind*	The indices where max was located

Returns

The maximum value over given ranges

6.17.3.34 bool my_array::populate_complex_slice (my_type * *dat_in*, size_t *n_dims*, size_t * *offsets*, size_t * *sizes*)

Populate a slice of array from the input array.

Extends populate_slice to fill an array slice from a larger array. As populate slice, assumes destination is a section of dimension m, with some finite offset in dimensions from m to n only. Assuming dat_in is a 1-d array in Fortran order (see get_element) this will read the proper subsection. Note this works fine for simple slices but costs more

Parameters

dat_in	pointer to data
n_dims	Dimensionality of input (must be less than dimension of array)
offsets	Offsets in the other dimensions
sizes	Sizes of input array

Returns

0 (success), 1 else

Todo Add testing of this

6.17.3.35 template<typename T > template bool my_array::populate_data (T *dat_in*, size_t *n_tot*)

Fill array

Populates this array from dat_in. n_tot should be the total number of elements in dat_in. The smaller of n_tot and the total number of elements in this array are copied. dat_in must match this array in row-column ordering and rank.

Parameters

dat_in	Source of copy
n_tot	Total number of elements to copy

Returns

0 (sucess) 1 (error).

6.17.3.36 bool my_array::populate_slice (my_type * *dat_in,* size_t *n_dims,* size_t * *offsets*)

Populate a slice of array from the input array.

Fill an array slice, that is a section of rank m < n_dims, with some finite offset in dimensions from m to n_dims only. offsets_n_dims = n_dims - m is the size of the array of offsets provided. E.g. to fill a row of a 3-d array call with n_dims_in = 3-1=2 and offsets={column, plane}. Or to fill an array of shape (x, y, t) at a single time value t_0, use n_dims_in=1, offsets={t_0}.

Parameters

dat_in	pointer to data
n_dims	Dimensionality of input (must be less than dimension of array)
offsets	Offsets in the other dimensions

Returns

0 for success, 1 for error

6.17.3.37 std::vector< size_t > my_array::read_dims_from_file (std::fstream & *file*)

Read dimensions from array file

Reads dims from file into vector. Returns empty vector on read error The layout is: sizeof(size_t) sizeof(my_type) io_verification_code Version string Next_block n_dims dims[n_dims] Next_block data IMPORTANT: the VERSION specifier links output files to code. If the file output is changed, commit and clean build with a bumped major version number tag to correctly specify this

Parameters

file	Filestream to read from

Returns

Vector of dimensions

6.17.3.38 bool my_array::read_from_file (std::fstream & *file*)

Read array from file

Reads data from file. This array should have already been created in the correct shape, otherwise we return an error.

The layout is: sizeof(size_t) sizeof(my_type) io_verification_code Version string Next_block n_dims dims[n_dims] Next_block data IMPORTANT: the VERSION specifier links output files to code. If the file output is changed, commit and clean build with a bumped major version number tag to correctly specify this

Parameters

file	Filestream to read from

Returns

0 (success), 1 else

6.17.3.39 bool my_array::resize (size_t *dim,* size_t *sz,* bool *verbose* = 0)

Resize my_array on the fly

dim is the dimension to resize, sz the new size. If sz < dims[dim] the first sz rows will be kept and the rest deleted. If sz > dims[dim] the new elements will be added zero initialised. Note due to using 1-d memory layout both cases require copying all data and therefore briefly memory to store the old and new arrays. However shinking the last dimension does not necessarily require a copy.

Parameters

dim	Dimension to resize
sz	New size for dimension
verbose	Flag to print extra info

Returns

0 (nothing to report) 1 else, including "new size matches, nothing done"

6.17.3.40 bool my_array::set_element (size_t *nx*, my_type *val*)

Sets array element

Sets elements at nx.

Returns

1 if nx is out of range or array is not rank 1, 0 else.

6.17.3.41 bool my_array::set_element (size_t *nx*, size_t *ny*, my_type *val*)

As my_array::set_element(size_t nx, my_type val) for 2-D arrays

6.17.3.42 bool my_array::set_element (size_t *nx*, size_t *ny*, size_t *nz*, my_type *val*)

As my_array::set_element(size_t nx, my_type val) for 3-D arrays

6.17.3.43 bool my_array::set_element (size_t *nx*, size_t *ny*, size_t *nz*, size_t *nt*, my_type *val*)

As my_array::set_element(size_t nx, my_type val) for 4-D arrays

6.17.3.44 bool my_array::set_element (size_t *n_dims*, size_t * *dim*, my_type *val*)

As my_array::set_element(size_t nx, my_type val) for N-D arrays, using array of indexes

6.17.3.45 bool my_array::shift (size_t *dim*, long *n_els*)

Shift array on dimension dim by n_els

Because of individual getter/setter per dimensionality, we use the 1-d backing to do this.

Parameters

dim	Dimension to shift, (0 to n_dims-1)
n_els	Number of elements to shift by.

Returns

0 (success) 1 else

Todo Fix special case

6.17.3.46 void my_array::smooth_1d (int *n_pts*)

Smooth a 1-d data_array

Smooths the data backing array. This does strange things for non-1d data at the ends.

Parameters

n_pts	Smoothing width

6.17.3.47 bool my_array::write_section_to_file (std::fstream & *file,* std::vector< size_t > *bounds*)

Write a subsection of array to file

We write only the section delimited by bounds, which should have two elements for each dimension of this array. The layout is: sizeof(size_t) sizeof(my_type) io_verification_code Version string Next_block n_dims dims[n_dims] Next↩ _block data IMPORTANT: the VERSION specifier links output files to code. If the file output is changed, commit and clean build with a bumped major version number tag to correctly specify this

Parameters

file	Filestream to write to
bounds	Vector of indices delimiting subsection to write

Returns

0 (success) 1 (error)

6.17.3.48 bool my_array::write_to_file (std::fstream & *file*)

Write array to file

Writes array to file. Data is in a few blocks each starting with a number defining their end position in the file. The layout is: sizeof(size_t) sizeof(my_type) io_verification_code Version string Next_block n_dims dims[n_dims] Next↩ _block data IMPORTANT: the VERSION specifier links output files to code. If the file output is changed, commit and clean build with a bumped major version number tag to correctly specify this

Parameters

file	Filestream to write to

Returns

0 (success) 1 (error)

6.17.3.49 void my_array::zero_data ( )

Reset data to 0

6.17.4 Friends And Related Function Documentation

6.17.4.1 bool populate_mirror_fastest (data_array & *data_out,* my_type * *result_in,* size_t *total_els*) [friend]

Copy FFTW data into array

For real data an FFT has a redundant half so FFTW returns array od size (dims[0]/2 +1)∗dims[...]∗dims[n-1] (Note that our first dim is FFTWs last). Since we want all k we have to mirror this to obey H(-f, -g) = H(f, g). For simplicity we enforce this on k_x, omega pair only. Data is assumed unshifted. Should work for 1-3 dimensions. 1st dimension will be returned shifted to 0-in-centre

Todo Which side is negative k?

Todo Can we replace reverse_copy with slice fillers and remove friendship?

6.17.5 Member Data Documentation

6.17.5.1 my_type* my_array::data [protected]

The data

6.17.5.2 size_t* my_array::dims [protected]

Array dimensions

6.17.5.3 size_t my_array::n_dims [protected]

Number of dimensions

6.18 non_thermal Class Reference

Nonthermal electron description.

`#include <non_thermal.h>`

Public Member Functions

- non_thermal (std::string file_prefix)
- ~non_thermal ()
- calc_type f_p (calc_type p_par, calc_type p_perp)
- calc_type d_f_p (calc_type p_par, calc_type p_perp, bool parallel)
- void set_dp (calc_type dp)
- calc_type get_total_dens ()
- bool get_norely ()

Public Attributes

- my_type * lookup_data

6.18.1 Detailed Description

Nonthermal electron description.

Small class to hold a non-thermal electron distribution we can operate on and with. Distribution is defined by reading a {filepath}nonthermal.conf file. This either specifies a functional form and corresponding constants as references into the deck.status file, or a lookup file containing a data_array. The interface remains the same, accessing either f_p(p_par, p_perp) or df/dp.

An example conf file is

```
ncomps = 1
nonrely = 1
hot:
function = max
dens=dens_rath
vpar =vtherm_parh
vperp=vtherm_perph
end:
```

specifying a single Bimaxwellian with density dens_rath from deck.status, etc. Note the "end:" line. Any number of components with any form can be supplied and the resulting f_p is their sum. The nonrely flag requests to use the non-relativistic calculations. An example of a lookup based nonthermal.conf is

```
ncomps = 1
hot:
function =lookup
lookup = my_data.dat
end:
```

where the file my_data contains a data_array. For example, the output of compress_distributions utility can be used, or the IDL routines in refit_distribs.

To add new functional forms, create a function such as bimax, below, and create the binding in non_thermal↩ ::configure_from_file under "Binding function free parameters to create f_p"

Caveat This is written for the input.deck files I used, so assumes, for example, that the term called "dens" in deck.↩ status is the cold plasma density. The names of deck constants for additional species etc are set using the conf files.

Todo Consider using .conf to set background params too

6.18.2 Constructor & Destructor Documentation

6.18.2.1 non_thermal::non_thermal (std::string *file_prefix*) [explicit]

Construct non-thermal distrib

Sets default params

Parameters

file_prefix	Prefix prepended to all files used

6.18.2.2 non_thermal::∼non_thermal ( )

Clean up. Calls clean_lookup() which can be used to do anything needed to cleanup after a lookup function Clean up.

Calls clean_lookup() which can be used to do anything needed to cleanup after a lookup function

6.18.3 Member Function Documentation

6.18.3.1 calc_type non_thermal::d_f_p (calc_type *p_par,* calc_type *p_perp,* bool *parallel*)

Get derivative of distribution

Get first derivative using two-point scheme

Parameters

p_par	Parallel momentum value
p_perp	Perpendicular momentum value
parallel	Flag for whether to do parallel (true) or perpendicular (false) p deriv

Returns

Value of first deriv of f

6.18.3.2 calc_type non_thermal::f_p (calc_type *p_par,* calc_type *p_perp*)

Return value of f(p_par, p_perp)

Evaluates current specification of f at p_perp, p_par and returns result. If f has multiple components these are summed.

Parameters

p_par	Parallel momentum value
p_perp	Perpendicular momentum value

Returns

Value of f at location

6.18.3.3 bool non_thermal::get_norely ( ) [inline]

Return state of flag for non-relativistic calculation

Returns

Boolean true if calculations are non-relativistic, false else

6.18.3.4 calc_type non_thermal::get_total_dens ( ) [inline]

Get the total density of non-thermal components

Returns

Ratio of density of all components to background density

6.18.3.5 void non_thermal::set_dp (calc_type *dp*) [inline]

Set the dp used to get numerical derivative

Parameters

dp	Value to set

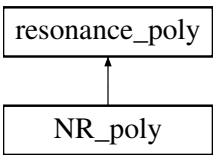
6.18.4 Member Data Documentation

6.18.4.1 my_type* non_thermal::lookup_data

Data pointer for use with a lookup type function backend. Note type matched to MY EPOCH data

6.19 NR_poly Class Reference

Inheritance diagram for NR_poly:



Public Member Functions

- **NR_poly** (double om_ce, double om_pe, double om_ce_ref, double om_ci, double om_pi)
- **NR_poly** (double om_ce, double om_pe, double om_ce_ref)
- NRVals **operator()** (const double x)
- bool sign_changes (double min, double max)
- bool is_root (double val, double tol=1e-4)
- std::pair< double, double > **refine_interval** (std::pair< double, double > init, int n_steps)
- void **dump_vals** ()

Additional Inherited Members

6.19.1 Member Function Documentation

6.19.1.1 bool NR_poly::is_root (double *val,* double *tol* = `1e-4`)

Check if val is a root

Checks if value is a root, by checking for a sign change in the interval val∗(1-tol) to val∗(1+tol)

Parameters

min	Value to check
tol	(optional) Fractional width of interval to check, default 1e-4

Returns

True if val is (probably) a root, false else

6.19.1.2 bool NR_poly::sign_changes (double *min,* double *max*)

Check if sign changes on interval

Checks for sign change between min and max. Does not assume min <= max

Parameters

min	Minimum of interval
max	Maximum of interval

Returns

True if sign changes between min and max, false else

6.20 plasma Class Reference

Plasma parameters and dispersion.

`#include <plasma.h>`

Public Member Functions

- plasma ()
- plasma (std::string file_prefix, my_type Bx_local=-1)
- bool is_good ()
- calc_type get_omega_ref (std::string code) const
- calc_type get_B0 ()
- void set_B0 (my_type B0)
- mu_dmudom get_mu (calc_type w, calc_type psi) const

  Solve plasma dispersion only.
- mu_dmudom get_high_dens_mu (calc_type w, calc_type psi) const

  Solve plasma dispersion only.
- mu_dmudom get_phi_mu_om (calc_type w, calc_type psi, calc_type alpha, int n, calc_type gamma_particle, bool skip_phi=false, bool Righthand=true) const
- mu_dmudom get_high_dens_phi_mu_om (calc_type w, calc_type psi, calc_type alpha, int n, calc_type gamma_particle, bool skip_phi=false, bool Righthand=true) const
- std::vector< calc_type > get_resonant_omega (calc_type theta, calc_type v_par, calc_type gamma_particle, int n) const

- std::vector< calc_type > get_resonant_omega_full (calc_type theta, calc_type v_par, calc_type gamma_↩ particle, int n) const
- bool **check_resonant_omega** (calc_type theta, calc_type v_par, calc_type gamma_particle, int n, calc_type omega, calc_type &result) const
- bool **check_resonant_omega_full** (calc_type theta, calc_type v_par, calc_type gamma_particle, int n, calc↩ _type omega, calc_type &result) const
- calc_type get_dispersion (my_type k, int wave_type, bool reverse=0, bool deriv=0, my_type theta=0.0) const

6.20.1 Detailed Description

Plasma parameters and dispersion.

Plasma objects contain specifications for a plasma, including density, B field, and species composition. In general we configure them from a file, plasma.conf, and no other constructor is provided. After construction, the plasma will be valid, but if given file is not found or reading fails, default values will be used. Two cyclotron frequencies are available, local and reference. For varying B fields, a reference B field should be given and the local om_↩ ce will match this. The reference value always matches that in the deck_constants struct. We offer functions to solve plasma dispersion "exactly" using the various get_root, get_phi∗ etc functions, or get_dispersion which uses various analytic approximations, usually high density ones. For the high-density approximations, the FIRST species is assumed to be the electrons. The former functions are modified from file mufunctions3.f90 author Clare E. J. Watt (18/05/10). From there: Note that mu is calculated using the Appleton-Hartree relation, and the choice of sign is obtained from Albert[1]. An additional function to simultaneously solve the Doppler type resonance condition, and the approximate dispersion relation are provided. IMPORTANT: Plasma setup relies on my_consts being defined!!

Author

Heather Ratcliffe

Date

07/10/2015

Caveat Note that no spatial variations in density or ambient B field are included in the dispersion calculations here

6.20.2 Constructor & Destructor Documentation

6.20.2.1 plasma::plasma ( ) [inline], [explicit]

Default constructor, create useless plasma object

6.20.2.2 plasma::plasma (std::string *file_prefix,* my_type *Bx_local* = −1) [explicit]

Set up plasma

Sets up components from {file_prefix}plasma.conf. If a Bx_local is given, store and calc local cyclotron frequency from this. Else use the cyclotron frequency from deck constants. IMPORTANT: Make sure my_consts is defined (read_deck and share_consts) before creating plasma!

Parameters

file_prefix	File prefix prepended to all files read
Bx_local	Local x-component of magnetic field

Caveat The ion frequencies assume a single ion species right now

6.20.3 Member Function Documentation

6.20.3.1 calc_type plasma::get_B0 ( ) [inline]

Return B0. This can vary in space

Returns

Value of B0 for this plasma

6.20.3.2 calc_type plasma::get_dispersion (my_type *k*, int *wave_type*, bool *reverse* = 0, bool *deriv* = 0, my_type *theta* = 0.0) const

Solve analytic dispersion (approx)

By default returns omega for a given k (see reverse and deriv params param). Uses local reference cyclotron and plasma frequencies and works with UNNORMALISED quantitites. NB: parameters out of range will silently return 0.

Parameters

k	Wavenumber
wave_type	wave species (see support.h)
reverse	Return k for input omega
deriv	Whether to instead return anayltic v_g
theta	Wavenormal angle, default 0.0

Returns

Value of omega, or k if reverse is set

Todo Complete Xmode?

Caveat For Whistler modes this is an approximation and intended to be perfectly reversible.

6.20.3.3 mu_dmudom plasma::get_high_dens_mu (calc_type *w*, calc_type *psi*) const [inline]

Solve plasma dispersion only.

Solve dispersion, omitting extended phi calcs, using high_dens approximation

Parameters

w	Wave frequency
psi	Wave normal angle

Returns

mu_dmudom struct containing mu and derivs

6.20.3.4 mu_dmudom plasma::get_high_dens_phi_mu_om (calc_type *w*, calc_type *psi*, calc_type *alpha*, int *n*, calc_type *gamma_particle*, bool *skip_phi* = false, bool *Righthand* = true) const

Solve plasma dispersion and extensions

Duplicates plasma::get_phi_mu_om but using reduced form of Stix parameters corresponding to a high-density assumption assuming the first species is the electrons. This is mainly for comparison with the exact solution to validate this assumption.

Parameters

w	Wave frequency
psi	Wave normal angle

alpha	particle pitch angle (for phi)
n	Resonance number
gamma_particle	Relativistic gamma for resonant particle
skip_phi	Omit phi calculation
Righthand	True for Righthand wave mode, false for left

Returns

mu_dmudom object containing mu info

Caveat Unsurprisingly this routine uses a high density approximation to the dispersion, which assumes $\omega_{pe} >> \omega_{ce}$

Caveat This routine assumes that electrons are the first species of the plasma, i.e. the first species in the plasma.↩ conf file

6.20.3.5 mu_dmudom plasma::get_mu (calc_type *w,* calc_type *psi*) const [inline]

Solve plasma dispersion only.

Solve dispersion, omitting extended phi calcs

Parameters

w	Wave frequency
psi	Wave normal angle

Returns

mu_dmudom struct containing mu and derivs

6.20.3.6 calc_type plasma::get_omega_ref (std::string *code*) const

Reference plasma and cyclotron frequencies

Get value of omega at local position

Parameters

code	two character code string. ce is actual Cyclotron freq. c0 is a reference value. pe is plasma frequency

Returns

Value of reference omega

6.20.3.7 mu_dmudom plasma::get_phi_mu_om (calc_type *w,* calc_type *psi,* calc_type *alpha,* int *n,* calc_type *gamma_particle,* bool *skip_phi* = false, bool *Righthand* = true) const

Solve plasma dispersion and extensions

Solves Appleton-Hartree plasma dispersion and returns struct containing mu, its derivatives and error code. Also returns the Phi defined by Lyons [4]. I.e. the set of values needed to calculate D See mu_dmudom

Duplicated from mufunctions by CEJ Watt

Parameters

w	Wave frequency
psi	Wave normal angle
alpha	particle pitch angle (for phi)
n	Resonance number
gamma_particle	Relativistic gamma for resonant particle
skip_phi	Omit phi calculation
Righthand	True for Righthand wave mode, false for left

Returns

mu_dmudom object containing mu info

On notation: within this routine we use notation as from mufunctions3.f90. In the return values as defined in support.h we match with Lyons [4] and Albert [1] . Thus in my_mu, we have lat, r, theta, omega for polar coordinate, r, wave normal angle and wave frequency

6.20.3.8 std::vector< calc_type > plasma::get_resonant_omega (calc_type *theta*, calc_type *v_par*, calc_type *gamma_particle*, int *n*) const

Solve plasma dispersion and doppler resonance simultaneously

Obtains solutions of the Doppler resonance condition omega - k_par v_par = -n Omega_ce and a high-density approximation to the Whistler mode dispersion relation simultaneously. Assumes pure electron-proton plasma and uses cubic_solve. ONLY solutions between -om_ce_local and om_ce_local, excluding omega = 0, are considered. "Zero" solutions are those less than the GEN_PRECISION constant in support.h.

Note that since k_parallel and v_parallel in resonant condition are signed, we will get multiple entries of ± omega for the corresponding ±k and ±n. These should be handled by the calling code, as k may or may not be handled with both signs

Parameters

theta	Wave normal angle
v_par	Particle velocity to solve with
gamma_particle	Relativistic gamma for resonant particle
n	Resonance number

Returns

Vector of solutions for resonant omega, or empty vector if no solutions are found

Extension Extend this to use full solution rather than the high density approx

6.20.3.9 std::vector< calc_type > plasma::get_resonant_omega_full (calc_type *theta*, calc_type *v_par*, calc_type *gamma_particle*, int *n*) const

Solve plasma dispersion and doppler resonance simultaneously

Obtains solutions of the Doppler resonance condition omega - k_par v_par = -n Omega_ce and the Whistler mode dispersion relation simultaneously. Assumes pure electron-proton plasma and uses cubic_solve. ONLY solutions between -om_ce_local and om_ce_local, excluding omega = 0, are considered. "Zero" solutions are those less than the GEN_PRECISION constant in support.h.

Note that since k_parallel and v_parallel in resonant condition are signed, we will get multiple entries of ± omega for the corresponding ±k and ±n. These should be handled by the calling code, as k may or may not be handled with both signs

Parameters

theta	Wave normal angle
v_par	Particle velocity to solve with
gamma_particle	Relativistic gamma for resonant particle
n	Resonance number

Returns

Vector of solutions for resonant omega, or empty vector if no solutions are found

Caveat I am assuming the solutions move but no more appear in the full equation. This may not be accurate, but we need a first guess for the solver

Caveat We set a hard minimum for resonant frequencies of interest, NR_min_om

6.20.3.10 bool plasma::is_good ( ) [inline]

Whether everything is setup

Returns

Boolean true is good, false else

6.20.3.11 void plasma::set_B0 (my_type *B0*)

Set B0

Sets the local reference B field value and thus om_ce_local value

Parameters

B0	Input B0 value

6.21 reader Class Reference

Reads SDF files into data_array.

`#include <reader.h>`

Public Member Functions

- reader ()
- reader (std::string file_prefix_in, const std::string block_id_in="", int ref_file_num_in=0)
- void update_ref_filenum (int num)
- int get_file_size ()
- bool change_block_id (std::string new_id)
- std::vector< std::pair< std::string, std::string > > list_blocks ()
- bool current_block_is_accum ()
- bool has_accum_data ()
- bool read_dims (size_t &n_dims, std::vector< size_t > &dims)
- bool read_dims (size_t &n_dims, std::vector< size_t > &dims, std::string b_id)
- int read_data (data_array &my_data_in, size_t time_range[3], size_t space_range[2], int flatten_on=-1)
- bool read_distrib (data_array &my_data_in, std::string dist_id, int dump_number)

Public Attributes

- std::string file_prefix
- size_t space_range [2]
- int time_range [3]
- char block_id [ID_SIZE]

6.21.1 Detailed Description

Reads SDF files into data_array.

Takes file prefixes, block id (see SDF documentation) and time and space ranges and a data_array to fill and does so.

Author

Heather Ratcliffe

Date

02/10/2015

6.21.2 Constructor & Destructor Documentation

6.21.2.1 reader::reader ( ) [explicit]

Create empty reader

Set fields to default null values

6.21.2.2 reader::reader (std::string *file_prefix_in*, const std::string *block_id_in* = " ", int *ref_file_num_in* = 0) [explicit]

Create reader

Sets up ids, sets n_chars etc. NOTE block_id_in and n_chars must be correctly set before any reads are done. Use update_ref_filenum(int num) and change_block_id(std::string new_id) to set these after construction.

Parameters

file_prefix_in	File prefix to prepend to all file names
block_id_in	String containing desired block id (e.g. ex) Note only the first ID_SIZE-1 chars are kept
ref_file_num_in	Reference file number to use for reading dimensions etc

6.21.3 Member Function Documentation

6.21.3.1 bool reader::change_block_id (std::string *new_id*)

Change block id

Change block id to new string

Parameters

new_id	String containing desired block id (e.g. ex) Note only the first ID_SIZE-1 chars are kept

Returns

Boolean true if value valid and set, false else

6.21.3.2 bool reader::current_block_is_accum ( )

Check if current block is accumulated

Checks whether the block named in reader is accumulated

Returns

Boolean true if accumulated, false else

6.21.3.3 int reader::get_file_size ( )

Get recorded file size

Reads the final block_end position from file. Acts as basic check of SDF integrity and reading. This should match size on disk.

Returns

Size of file in bytes

6.21.3.4 bool reader::has_accum_data ( )

Check file for accumulated data

Check if reference file contains accumulated blocks (named a[x/y/z] or ab[x/y/z]

6.21.3.5 std::vector< std::pair< std::string, std::string > > reader::list_blocks ( )

List blocks in reference file

Lists blocks in the file given by ref_file_num

Returns

Vector containing pairs of block name and the internal id string

6.21.3.6 int reader::read_data (data_array & *my_data_in*, size_t *time_range[3]*, size_t *space_range[2]*, int *flatten_on* = -1)

Read data into given array

Open files dictated by time_range sequentially, and populates the data_array. Data_array should be set to correct dimensions already, else we return with error and leave data_array in partially updated state.

Parameters

out	*my_data_in*	Data array to read into, already set to have correct dimensions for data
	time_range	Time specs. Time_range[0,1] are the file numbers to read, [3] is a number of times (rows) for accumulated data and is ignored for normal blocks. Reading stops when time_range[2] is reached (plain blocks), file number time_range[2] or row time_range[3] is reached (accumulated blocks), or no more files are available on disk.
	space_range	x-dimension space to cover. Can be set to cut out a part of the x-dimension, or set to the entire x_size.
	flatten_on	Flatten read data on this dimension before storing

Returns

0 for success, 1 for error 2 for unusual exit, i.e. early termination

6.21.3.7 bool reader::read_dims (size_t & *n_dims*, std::vector< size_t > & *dims*)

Read dimensions of current block (this->block_id). See reader::read_dims(size_t &n_dims, std::vector<size_t> &dims, std::string b_id)

Parameters

out	*n_dims*	Rank of data block
out	*dims*	Dimensions of data block

Returns

0 (success), 1 else)

6.21.3.8 bool reader::read_dims (size_t & *n_dims,* std::vector< size_t > & *dims,* std::string *b_id*)

Gets dimensions of the block specified by b_id

Looks up block b_id in file numbered ref_file_num and gets dimension info. Note we don't have to read the data, only the block list.

Parameters

out	*n_dims*	Rank of data block
out	*dims*	Dimensions of data block
	b_id	Name of block to read

Returns

0 (success), 1 else)

6.21.3.9 bool reader::read_distrib (data_array & *my_data_in,* std::string *dist_id,* int *dump_number*)

Read distribution function

Reads the distribution function dist_id into data_in. ID has dist_fn removed and is trimmed to 10 chars max. If data type does not match compiled type we convert, assuming it is either a double or float type.

Parameters

out	*my_data_in*	Array to read distribution into. Should be set to correct size already
	dist_id	String name of required distrib block
	dump_number	Number of file to read

Returns

0 (success), 1 else

6.21.3.10 void reader::update_ref_filenum (int *num*)

Update reference file number

Set new reference file number and update n_chars parameter

Parameters

num	New reference file number

6.21.4 Member Data Documentation

6.21.4.1 char reader::block_id[ID_SIZE]

Name of block to extract

6.21.4.2 std::string reader::file_prefix

Prefix of files before dump number

6.21.4.3 size_t reader::space_range[2]

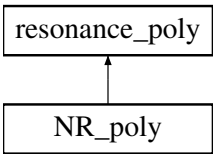
Space range in x to extract

6.21.4.4 int reader::time_range[3]

Time range to extract

6.22 resonance_poly Class Reference

Inheritance diagram for resonance_poly:



Public Member Functions

- **resonance_poly** (double om_ce, double om_pe, double om_ce_ref)
- **resonance_poly** (double om_ce, double om_pe, double om_ce_ref, double om_ci, double om_pi)
- void calculate_coeffs_no_ion (double psi, double v_par, int n, double gamma)
- void calculate_coeffs_full (double psi, double v_par, int n, double gamma)

Protected Attributes

- double **om_ce**
- double **om_pe**
- double **om_ce_ref**
- double **om_ci**
- double **om_pi**
- std::vector< double > **coeff**
- bool **first_calc**

6.22.1 Member Function Documentation

6.22.1.1 void resonance_poly::calculate_coeffs_full (double *psi,* double *v_par,* int *n,* double *gamma*)

Calculate coeffs for general case

Calculate the 10th order coefficients in the limit of omega, omega_ce, Omega_e >> omega_ci, Omega_i. These are normalised, so the polynomial returns solutions for omega/omega_ce_ref

Parameters

psi	Wave normal angle
v_par	Parallel particle velocity
n	Resonant number
gamma	Particle gamma factor

Todo Pre-calc and stash as much as possible, in particular consider psi in A, B, C

Coefficents in limit of electron freqs >> ion

6.22.1.2 void resonance_poly::calculate_coeffs_no_ion (double *psi,* double *v_par,* int *n,* double *gamma*)

Calculate coeffs for negligible ion frequencies

Calculate the 10th order coefficients in the limit of omega, omega_ce, Omega_e >> omega_ci, Omega_i. These are normalised, so the polynomial returns solutions for omega/omega_ce_ref

Parameters

psi	Wave normal angle
v_par	Parallel particle velocity
n	Resonant number
gamma	Particle gamma factor

Todo Pre-calc and stash as much as possible, in particular consider psi in A, B, C

Coefficents in limit of electron freqs >> ion

6.23 running_report Struct Reference

Progress info structure.

`#include <d_coeff.h>`

Public Attributes

- size_t last_report
- size_t report_interval
- bool quiet

6.23.1 Detailed Description

Progress info structure.

6.23.2 Member Data Documentation

6.23.2.1 size_t running_report::last_report

Index of last report printed

6.23.2.2 bool running_report::quiet

Flag set to disable report printing

6.23.2.3 size_t running_report::report_interval

Interval to print reports at

6.24 setup_args Struct Reference

General command line arguments.

`#include <support.h>`

Public Attributes

- size_t time [3]
- bool use_row_time
- int space [2]
- std::string block
- std::string file_prefix
- int n_space
- size_t per_proc

6.24.1 Detailed Description

General command line arguments.

Processed command line arguments used across several programs

6.24.2 Member Data Documentation

6.24.2.1 std::string setup_args::block

Block ID to use (ex, bz etc)

6.24.2.2 std::string setup_args::file_prefix

Prefix part of file names

6.24.2.3 int setup_args::n_space

Number of space blocks in global x direction

6.24.2.4 size_t setup_args::per_proc

Resulting number of space blocks per proc

6.24.2.5 int setup_args::space[2]

Local space block start and end

6.24.2.6 size_t setup_args::time[3]

Start and end dump numbers

6.24.2.7 bool setup_args::use_row_time

Whether to use time[2] for sizing

6.25 spect_args Struct Reference

Command line arguments for spectra.

`#include <support.h>`

Public Attributes

- int fuzz
- int smth
- size_t n_ang

- int wave
- int ang
- float ang_sd
- bool mask

6.25.1 Detailed Description

Command line arguments for spectra.

Processed command line arguments for spectra used across several programs

6.25.2 Member Data Documentation

6.25.2.1 int spect_args::ang

Angular function type (can be FUNCTION_NULL)

6.25.2.2 float spect_args::ang_sd

Width for angular function (if applicable)

6.25.2.3 int spect_args::fuzz

Fuzz for spectral cutout

6.25.2.4 bool spect_args::mask

Flag to output spectrum extraction mask to file also

6.25.2.5 size_t spect_args::n_ang

Number of angles for output spectrum

6.25.2.6 int spect_args::smth

Smoothing width for output spectrum

6.25.2.7 int spect_args::wave

Wave type ID (see support.h WAVE_*)

6.26 spectrum Class Reference

A spectrum in omega and angle.

`#include <spectrum.h>`

Public Types

- enum part { **B**, **ang** }

Public Member Functions

- bool get_g_is_angle_only ()
- spectrum & operator= (const spectrum &src)
- spectrum (const spectrum &src)
- spectrum (spectrum &&src)=default

- ~spectrum ()
- bool operator== (const spectrum &rhs) const
- bool operator!= (const spectrum &rhs) const
- bool is_good () const
- bool generate_spectrum (data_array &parent, int om_fuzz, int angle_type, my_type std_dev, data_array *mask=nullptr)
- void set_ids (float time1, float time2, int space1, int space2, int wave_id, char block_id[10], int function_type=FUNCTION_DELTA)
- void set_extra_ids (int wave_id, int function_type=FUNCTION_DELTA)
- void copy_ids (const data_array &src)
- bool check_ids (const data_array &src) const
- void copy_tags (const spectrum &src)
- bool check_tags (const spectrum &src) const
- my_type get_omega (my_type k, int wave_type, bool deriv=0, my_type theta=0.0)
- my_type get_k (my_type omega, int wave_type, bool deriv=0, my_type theta=0.0)
- void smooth_B (int n_pts)
- bool truncate_om (my_type om_min, my_type om_max)
- bool truncate_x (my_type x_min, my_type x_max)
- calc_type check_upper ()
- calc_type get_peak_omega ()
- bool calc_norm_B ()
- bool calc_norm_g (size_t om_ind)
- my_type get_norm_B ()
- my_type get_norm_g (size_t om_ind)
- void apply (spectrum::part subarr, std::function< my_type(my_type arg)> func)
- void apply (spectrum::part subarr, std::function< my_type(my_type arg, my_type arg2)> func, my_type arg)
- void renormalise ()

  Re-do normalising of spectrum.
- bool write_to_file (std::fstream &file)
- bool read_from_file (std::fstream &file)
- data_array copy_out_B ()
- data_array copy_out_g ()
- my_type get_B_element (size_t n_om) const
- my_type get_g_element (size_t n_ang) const
- my_type get_g_element (size_t n_om, size_t n_ang) const
- void set_B_element (size_t n_om, my_type val)
- void set_g_element (size_t n_ang, my_type val)
- void set_g_element (size_t n_om, size_t n_ang, my_type val)
- my_type get_om_axis_element (size_t nx) const
- my_type get_ang_axis_element (size_t nx) const
- long get_om_axis_index_from_value (my_type omega) const
- long get_ang_axis_index_from_value (my_type ang) const
- void set_om_axis_element (size_t nx, my_type val)
- void set_ang_axis_element (size_t nx, my_type val)
- size_t get_g_dims () const
- size_t get_g_dims (size_t i) const
- size_t get_B_dims () const
- size_t get_B_dims (size_t i) const
- size_t get_angle_length () const
- size_t get_omega_length () const

Public Attributes

- char block_id [ID_SIZE]
- my_type time [2]
- size_t space [2]
- int wave_id

Friends

- class controller

6.26.1 Detailed Description

A spectrum in omega and angle.

Holds data on the omega and angle distributions. If fed an FFTd data array this will be X^2(omega, theta) where X is E or B. The latter can depend on omega! Can be created/destroyed only by controllers, so has no public constructor/destructors. IMPORTANT: because we are working with FFT data, we assume the angle/frequency axis either covers some small cutout in +ve domain, or is symmetrical in positive and negative values. A few of the specific routines here use this to simplify things. The sign of omega is simply copied from the sign of k. The "angle" axis is stored as tan(theta) for theta the wave normal angle. Access to elements should use the wrappers at the bottom of spectrum.h, described in Spectrum access wrappers because internal layout could change in future. Note that prior to v1.1 the wave power was the TOTAL of the B^2 array, for v1.1 and later it is the INTEGRAL

Author

Heather Ratcliffe

Date

24/09/2015

6.26.2 Constructor & Destructor Documentation

6.26.2.1 spectrum::spectrum (const spectrum & *src*)

Copy constructor

(Deep) copy src to a new instance.

Parameters

src	Spectrum to copy from

6.26.2.2 spectrum::spectrum (spectrum && *src*) [default]

Move a spectrum object

6.26.2.3 spectrum::~spectrum ( )

Delete spectrum

Free any allocated memory

6.26.3 Member Function Documentation

6.26.3.1 void spectrum::apply (spectrum::part *subarr*, std::function< my_type(my_type arg)> *func*)

Apply a function to each element of selected spectrum part. func must take and return a my_type or type convertible to this

Parameters

subarr	Which part to apply to, see spectrum::part
func	Function to apply

6.26.3.2 void spectrum::apply (spectrum::part *subarr*, std::function< my_type(my_type arg, my_type arg2)> *func*, my_type *arg*)

Apply a function to each element of selected spectrum part. func must take two my_type and return one my_type or type convertible to this

Parameters

subarr	Which part to apply to, see spectrum::part
func	Function to apply
arg	Second argument to function

6.26.3.3 bool spectrum::calc_norm_B ( )

Calculate norming of B(w)

Calculate the total square integral of values over range, int_{om_min}^{om_max} B^2(omega) d omega.

Returns

0 (success), 1 (error)

6.26.3.4 bool spectrum::calc_norm_g (size_t *om_ind*)

Normalise g_w(x)

Calculate the norm of g used in e.g. denom of Albert [1] Eq 3 or calc'd in Derivations.tex [5]. Contains one value for each omega entry.

Parameters

om_ind	Omega index to calculate norm at

Returns

0 (success), 1 (error e.g. out of range)

6.26.3.5 bool spectrum::check_ids (const data_array & *src*) const

Check ids match

Checks ID fields match src. ID fields are the block_id, and the space and time values. See also spectrum::check_tags()

Parameters

src	Array to check ids against

Returns

True if ids are equal, false else

6.26.3.6 bool spectrum::check_tags (const spectrum & *src*) const

Check tags

Check tag fields match src. Tags are the g_is_angle_only, the function_type and the wave_id. See also spectrum::check_ids()

Parameters

src	Spectrum to compare tags against

Returns

True if equal, false else

6.26.3.7 calc_type spectrum::check_upper ( )

Check upper k limit of spectral power

Checks the upper bound of region of significant spectral power, i.e. above SPECTRUM_THRESHOLD∗peak_power

Returns

The physical wavenumber where spectrum drops below threshold

6.26.3.8 void spectrum::copy_ids (const data_array & *src*)

Copy id fields

Copies ID fields from src array to this. ID fields are the block_id, and the space and time values. These are attached to the spectrum AND to the B and g arrays it holds. See also spectrum::copy_tags()

Parameters

src	Array to copy ids from

6.26.3.9 data_array spectrum::copy_out_B ( )

Return a copy of B array

Make a copy of the B part of data

Returns

A data array containing a copy of the B data

6.26.3.10 data_array spectrum::copy_out_g ( )

Return a copy of g array Make a copy of the g part of data

Returns

A data array containing a copy of the g data

6.26.3.11 void spectrum::copy_tags (const spectrum & *src*)

Copy tags

Copies tag fields from src array to this. Tags are the g_is_angle_only, the function_type and the wave_id. See also spectrum::copy_ids()

Parameters

src	Spectrum to copy tags from

6.26.3.12 bool spectrum::generate_spectrum (data_array & *parent*, int *om_fuzz*, int *angle_type*, my_type *std_dev*, data_array ∗ *mask* = `nullptr`)

Generate spectrum from data

Takes a parent data array and generates the corresponding spectrum. Windows using the specified wave dispersion and integrates over frequency using om_fuzz percent band. Axes are copied from the parent. If the spectrum is of separable type (g_is_angle_only = true), the angular distribution is generated with functional form specified by angle_type. IMPORTANT: when using real angular data we roughly fuzz around the correct k values, but this is not uniform! Non-smooth or rapidly varying data may give odd results

Parameters

parent	Data array to read from. Spectrum will have the same units as this
om_fuzz	Band width around dispersion curve in percent of central frequency
angle_type	Angular distribution functional form
std_dev	Standard deviation of angular functional form (where applicable)
mask	(optional) data_array matching sizes of parent, will be filled with the masking array used for spectrum generation. If nullptr or nothing is supplied, no mask is output.

Returns

0 for successful calculation, 1 for error

Todo 2-d and 3-d extractions don't quite agree at k=0. factor ∼10 and variations near 0

6.26.3.13 bool spectrum::get_g_is_angle_only ( ) [inline]

Get flag showing if spectrum is separable

Returns

True if g is a function of only angle (and not frequency) false else

6.26.3.14 my_type spectrum::get_k (my_type *omega,* int *wave_type,* bool *deriv* = 0, my_type *theta* = 0.0)

Gets k for given omega

Uses dispersion relation for given wave_type to convert omega to k. Calls to plasma because approximations for density etc etc should be made there.

Parameters

omega	Frequency
wave_type	Wave species
deriv	Return v_g instead
theta	Wave normal angle

Returns

Value of k for given omega and wave type etc

6.26.3.15 my_type spectrum::get_norm_B ( ) [inline]

Get the normalising constant for B part of spectrum

Returns

Current value of norm_B

6.26.3.16 my_type spectrum::get_norm_g (size_t *om_ind*) [inline]

Get the normalising constant for g part of spectrum

Parameters

om_ind	Frequency index to get from, 0 for separable spectra

Returns

Current value of norm_g at specified location

6.26.3.17 my_type spectrum::get_omega (my_type *k*, int *wave_type*, bool *deriv* = 0, my_type *theta* = 0.0)

Gets omega for given k

Uses dispersion relation for given wave_type to convert k to omega. Calls to plasma because approximations for density etc etc should be made there.

Parameters

k	Wavenumber
wave_type	Wave species
deriv	Return v_g instead
theta	Wave normal angle

Returns

Value of omega for given k and wave type etc

6.26.3.18 calc_type spectrum::get_peak_omega ( )

Find position of spectral peak

Finds location of highest peak in spectrum.

Returns

The physical axis value where the spectral peak occurs

6.26.3.19 bool spectrum::is_good ( ) const [inline]

Check if a spectrum is complete and useable

Returns

Boolean true if good, false else

6.26.3.20 bool spectrum::operator!= (const spectrum & *rhs*) const [inline]

See spectrum::operator==()

6.26.3.21 spectrum & spectrum::operator= (const spectrum & *src*)

Copy assignment

Sets this equal to a (deep) copy of source, i.e duplicates the B and g arrays and all other fields

Parameters

src	Spectrum to copy from

Returns

Copy of input spectrum

6.26.3.22 bool spectrum::operator== (const spectrum & *rhs*) const

Equality operator

Check this is equal to rhs. Since copies are always deep, we check values, not data pointers. We ignore the derived things such as smooth and norm_g

Parameters

rhs	Spectrum to compare to

Returns

True if equal, false else

6.26.3.23 bool spectrum::read_from_file (std::fstream & *file*)

Initialise spectrum from file

Reads a dump file which is expected to contain two arrays, first B then g, as written by spectrum->write_to_file, and constructs spectrum from data

Parameters

file	Filestream to read from

Returns

0 (success), 1 (error)

6.26.3.24 void spectrum::renormalise ( ) [inline]

Re-do normalising of spectrum.

Recalculate norm_B and norm_g and some other stored norm data

6.26.3.25 void spectrum::set_extra_ids (int *wave_id,* int *function_type* = FUNCTION_DELTA)

Set id fields

Sets the wave type and angle type attached to spectra (i.e. the ids that can't be inherited from parent data)

Parameters

wave_id	Wave type (see support.h)
function_type	Functional form of spectrum in angle

6.26.3.26 void spectrum::set_ids (float *time1,* float *time2,* int *space1,* int *space2,* int *wave_id,* char *block_id[10],* int *function_type* = FUNCTION_DELTA)

Set id fields

Sets the time and space ranges, wave type etc attached to this spectrum. Times should be in seconds. Space in terms of grid points.

Parameters

time1	Initial time of data used
time2	End time of data used

space1	Start index of space range of data used
space2	End index of space range of data used
wave_id	Wave type (see support.h)
block_id	Name of block used to derive this spectrum
function_type	Functional form of spectrum in angle

6.26.3.27 void spectrum::smooth_B (int *n_pts*)

Smooth B

Apply a box-car smoothing to B with specified number of pts. Store n_pts in smooth field

Parameters

n_pts	Boxcar smoothing width to apply

6.26.3.28 bool spectrum::truncate_om (my_type *om_min,* my_type *om_max*)

Truncate omega distribution at om_min and om_max.

Zeros all elements outside the range [om_min, om_max]. Zeros are ignored. om_min must be $<$ om_max. Om_min or max out of axis range does nothing on that end. NB B is renormalised after the truncation

Parameters

om_min	Minimum physical omega to truncate at
om_max	Maximum physical omega to truncate at

Returns

0 (success), 1 (range error)

6.26.3.29 bool spectrum::truncate_x (my_type *x_min,* my_type *x_max*)

Truncate angle distribution at x_min and x_max.

Zeros all elements outside the range [x_min, x_max]. x_min must be $<$ x_max. If x_min or max are out of range, nothing is done at that end

Parameters

x_min	Minimum tan theta to truncate at
x_max	Maximum tan theta to truncate at

Returns

0 (success), 1 (range error)

6.26.3.30 bool spectrum::write_to_file (std::fstream & *file*)

Write to file

Spectra are written by writing out the B array, the g array, and then writing a single closing footer containing the id values again.

Parameters

file	Filestream to write to

Returns

0 (success), 1 (error)

6.26.4 Friends And Related Function Documentation

6.26.4.1 friend class controller [friend]

Controllers can create/destroy spectra and access their internals

6.26.5 Member Data Documentation

6.26.5.1 char spectrum::block_id[ID_SIZE]

The field name id from SDF file

6.26.5.2 size_t spectrum::space[2]

Space range over which data are taken

6.26.5.3 my_type spectrum::time[2]

Time range over which data are taken

6.26.5.4 int spectrum::wave_id

ID for which wave mode cutout we're going for. See support.h

References

[1] J. M. Albert. Evaluation of quasi-linear diffusion coefficients for whistler mode waves in a plasma with arbitrary density ratio. *JGR (Space Physics)*, 110:A03218, March 2005. 7, 35, 87, 88, 104, 107, 118

[2] S. A. Glauert and R. B. Horne. Calculation of pitch angle and energy diffusion coefficients with the PADIE code. *JGR (Space Physics)*, 110:A04206, April 2005. 29

[3] L. R. Lyons. General relations for resonant particle diffusion in pitch angle and energy. *Journal of Plasma Physics*, 12:45–49, August 1974. 7

[4] L. R. Lyons. Pitch angle and energy diffusion coefficients from resonant interactions with ion-cyclotron and whistler waves. *Journal of Plasma Physics*, 12:417–432, December 1974. 7, 106, 107

[5] H. Ratcliffe. Derivations involved in Diffusion calculations. 2017. 118

[6] D. Summers, B. Ni, and N. P. Meredith. Timescales for radiation belt electron acceleration and loss due to resonant wave-particle interactions: 1. Theory. *JGR (Space Physics)*, 112:A04206, April 2007. 29, 30

Index