Alliance

1.0

Generated by Doxygen 1.9.1

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 array.c File Reference

array manipulation module

```
#include "array.h"
#include "utils_fftw.h"
#include "space_config.h"
```

Macros

- #define **CHI_EM** 3
- #define **CHI_EL** 1
- #define **FFT_OFFSET** 2

Functions

- size_t get_flat_c (size_t is, size_t il, size_t im, size_t ix, size_t iy, size_t iz)

  returns flat index of the element of complex 6D array
- size_t getIndChiBufEM_c (size_t ix, size_t iy, size_t iz, size_t is, size_t ifield)

  returns flat index of an element of electromagnetic gyrokinetic potential in FOURIER SPACE
- size_t getIndChiBufEM_r (size_t ix, size_t iy, size_t iz, size_t is, size_t ifield)

  returns flat index of an element of electromagnetic gyrokinetic potential in POSITION SPACE
- size_t getIndChiBufEL_c (size_t ix, size_t iy, size_t iz, size_t is)

  returns returns flat index of an element of electrostatic gyrokinetic potential in FOURIER SPACE
- size_t getIndChiBufEL_r (size_t ix, size_t iy, size_t iz, size_t is)

  returns returns flat index of an element of electrostatic gyrokinetic potential in REAL SPACE
- size_t get_flat_r (size_t is, size_t il, size_t im, size_t ix, size_t iy, size_t iz)

  returns flat index of the element of real 6D array
- size_t get_flatIndexComplex3D (size_t ix, size_t iy, size_t iz)

  returns flat array of complex 3D array
- size_t getIndChi (size_t ix, size_t iy, size_t iz, size_t is)
- void multiply_ar_c (COMPLEX *ar1, COMPLEX *ar2, COMPLEX *ret)
- void multiply_ar_r (const double *ar1, const double *ar2, double *ret)

Variables

- struct array_size **array_local_size**
- struct array_size **array_global_size**
- struct offset_size **array_offset**
- struct offset_size **array_offset3D**

2.1.1 Detailed Description

array manipulation module

contains functions which are supposed to make array manipulation simpler

2.1.2 Function Documentation

2.1.2.1 get_flat_c()

```
size_t get_flat_c (
            size_t is,
            size_t il,
            size_t im,
            size_t ix,
            size_t iy,
            size_t iz )
```

returns flat index of the element of complex 6D array

Parameters

is	species type
il	Laguerre moment
im	Hermite moment
ix	kx index
iy	ky index
iz	kz index

returns flattened index of a complex array from its 6D index. Flattened index then can be passed to distribution function 6D array to get a required element at position (is,il,im,ix,iy,iz).

2.1.2.2 get_flat_r()

```
size_t get_flat_r (
            size_t is,
            size_t il,
            size_t im,
            size_t ix,
```

```
            size_t iy,
            size_t iz )
```

returns flat index of the element of real 6D array

Parameters

is	species type
il	Laguerre moment
im	Hermite moment
ix	x index
iy	y index
iz	z index

returns flattened index of a real array from its 6D index. Flattened index then can be passed to distribution function 6D array to get a required element at position (is,il,im,ix,iy,iz).

2.1.2.3 get_flatIndexComplex3D()

```
size_t get_flatIndexComplex3D (
            size_t ix,
            size_t iy,
            size_t iz )
```

returns flat array of complex 3D array

Parameters

ix	kx index
iy	ky index
iz	kz index

returns flattened index of a complex array from its 3D position index. Flattened index then can be passed to one of the fields ($\phi(\mathbf{k}), A_{||}(\mathbf{k}), B_{||}(\mathbf{k})$) 6D array to get a required element at position (ix,iy,iz).

2.1.2.4 getIndChi()

```
size_t getIndChi (
            size_t ix,
            size_t iy,
            size_t iz,
            size_t is )
```

getIndChi(size_t ix,size_t iy, size_t iz, size_t is)

2.1.2.5 getIndChiBufEL_c()

```
size_t getIndChiBufEL_c (
            size_t ix,
```

```
        size_t iy,
        size_t iz,
        size_t is )
```

returns returns flat index of an element of electrostatic gyrokinetic potential in FOURIER SPACE

Parameters

ix	kx index
iy	ky index
iz	kz index
is	particle species index

returns flattened index of a gyrokinetic potential $\chi^{\phi}(\mathbf{k})$ from its 4D index in FOURIER SPACE. flattened index is then can be used to access required value of the gyrokinetic potential at position (ix,iy,iz,is).

2.1.2.6 getIndChiBufEL_r()

```
size_t getIndChiBufEL_r (
             size_t ix,
             size_t iy,
             size_t iz,
             size_t is )
```

returns returns flat index of an element of electrostatic gyrokinetic potential in REAL SPACE

Parameters

ix	x index
iy	y index
iz	z index
is	particle species index

returns flattened index of a gyrokinetic potential $\chi^{\phi}(\mathbf{r})$ from its 4D index in REAL SPACE. flattened index is then can be used to access required value of the gyrokinetic potential at position (ix,iy,iz,is).

2.1.2.7 getIndChiBufEM_c()

```
size_t getIndChiBufEM_c (
             size_t ix,
             size_t iy,
             size_t iz,
             size_t is,
             size_t ifield )
```

returns flat index of an element of electromagnetic gyrokinetic potential in FOURIER SPACE

Parameters

ix	kx index
iy	ky index
iz	kz index
is	particle species index
ifield	field type

returns flattened index of a gyrokinetic potential $\chi^{\phi,A,B}$ from its 4D index in FOURIER SPACE. flattened index is then can be used to access required value of the gyrokinetic potential at position (ix,iy,iz,is). Type of gyrokinetic potential is specified by ifield parameter. Use 0 is to access $\chi^{\phi}(\mathbf{k})$, 1 to access $\chi^{A}(\mathbf{k})$ and 2 to access $\chi^{B}(\mathbf{k})$.

2.1.2.8 getIndChiBufEM_r()

```
size_t getIndChiBufEM_r (
            size_t ix,
            size_t iy,
            size_t iz,
            size_t is,
            size_t ifield )
```

returns flat index of an element of electromagnetic gyrokinetic potential in POSITION SPACE

Parameters

ix	x index
iy	y index
iz	z index
is	particle species index
ifield	field type

returns flattened index of a gyrokinetic potential $\chi^{\phi,A,B}(\mathbf{k})$ from its 4D index in POSITION SPACE. flattened index is then can be used to access required value of the gyrokinetic potential at position (ix,iy,iz,is). Type of gyrokinetic potential is specified by ifield parameter. Use 0 is to access $\chi^{\phi}(\mathbf{r})$, 1 to access $\chi^{A}(\mathbf{r})$ and 2 to access $\chi^{B}(\mathbf{r})$.

2.1.2.9 multiply_ar_c()

```
void multiply_ar_c (
            COMPLEX * ar1,
            COMPLEX * ar2,
            COMPLEX * ret )
```

multiply_ar_c(COMPLEX ∗ar1, COMPLEX ∗ar2, COMPLEX ∗ret)

2.1.2.10 multiply_ar_r()

```
void multiply_ar_r (
            const double * ar1,
            const double * ar2,
            double * ret )
```

multiply_ar_r(const double ∗ar1, const double ∗ar2, double ∗ret)

2.2 diagnostics.c File Reference

diagnostics module

```
#include "diagnostics.h"
#include "parameters_io.h"
```

Macros

- #define **TO_ROOT** 0
- #define **BUFFER_SIZE** 1

Functions

- void diag_computeSpectra (const COMPLEX *g, const COMPLEX *h, int timestep)
  general function to compute k or m spectra
- void diag_initSpec ()
  initialize spectra computation
- void diag_computeFreeEnergy (COMPLEX *g, COMPLEX *h)
  compute free energy
- void diag_computeKSpectrum (const COMPLEX *g, const COMPLEX *h, double *spec)
- void diag_computeMSpectrum (const COMPLEX *g, const COMPLEX *h, double *spec)
  computes free energy spectra in m space
- void diag_getShells ()
  computes shells from parameters
- double diag_computeFreeEnergyFields (COMPLEX *g, COMPLEX *fields)
  to be done later
- void diag_compute (COMPLEX *g, COMPLEX *h, int timestep)
  computes all diagnostics

Variables

- double * diag_kSpec = 0
  used to store free energy k spectra
- double * diag_mSpec = 0
  used to store free energy m spectra
- double * diag_shells = 0
  used to store positions of k shells required to compute k spectra
- double diag_freeEnergy
  free energy

2.2.1 Detailed Description

diagnostics module

different diagnostic tools are gathered in this module

2.2.2 Function Documentation

2.2.2.1 diag_compute()

```
void diag_compute (
            COMPLEX * g,
            COMPLEX * h,
            int timestep )
```

computes all diagnostics

Parameters

g	modified distribution function
h	distribution function
iter	current time step

2.2.2.2 diag_computeFreeEnergy()

```
void diag_computeFreeEnergy (
            COMPLEX * g,
            COMPLEX * h )
```

compute free energy

Parameters

g	modified gyrokinetic distribution function
h	gyrokintic distribution function

computes free energy as $W = 2.\Re(\sum_{k_x,k_y,k_z>0,m,l,s} g * \bar{h})$, taking into account reality condition.

2.2.2.3 diag_computeFreeEnergyFields()

```
diag_computeFreeEnergyFields (
            COMPLEX * g,
            COMPLEX * fields )
```

to be done later

Parameters

g	
fields	computes free energy from the fields and distribution function.

2.2.2.4 diag_computeKSpectrum()

```
void diag_computeKSpectrum (
            const COMPLEX * g,
            const COMPLEX * h,
            double * spec )
```

Parameters

g	modified gyrokinetic distribution function
h	gyrokinetic distribution function
spec	spectra array

computes free energy $k_\perp$ spectra $W(k_i^{shell}) = \frac{1}{N}\sum_{k_{i-1}^{shell}<|\mathbf{k}_\perp|<k_i^{shell}}\sum_{k_z,l,m,s} g\bar{h}$ where N is a number of wave vectors between shells k_{i-1}^{shell} and k_i^{shell}

2.2.2.5 diag_computeMSpectrum()

```
void diag_computeMSpectrum (
            const COMPLEX * g,
            const COMPLEX * h,
            double * spec )
```

computes free energy spectra in m space

Parameters

g	modified gyrokinetic distribution function
h	gyrokinetic distribution function
spec	spectra array

computes free energy m spectra as $W(m) = \sum_{k_x,k_y,k_z,l,s} g\bar{h}$

2.2.2.6 diag_computeSpectra()

```
void diag_computeSpectra (
            const COMPLEX * g,
            const COMPLEX * h,
            int timestep )
```

general function to compute k or m spectra

Parameters

g	gyrokinetic distribution function
h	distribution function
timestep	current time step

function computes spectra at timestep as given in parameter file. $k_\perp$ spectra is computed using diag_computeKSpectrum, and m spectra is computed using diag_computeMSpectrum

2.2.2.7 diag_getShells()

```
diag_getShells ( )
```

computes shells from parameters

computes positions of k_shells in between last_shell and first_shell as provided by user in parameter file. Position of i^{th} shell is computed as $k_i^{shell} = (last\_shell - first\_shell)/(k\_shells) \cdot i$

2.2.2.8 diag_initSpec()

```
void diag_initSpec ( )
```

initialize spectra computation

Prepares free energy spectra computation. For spectra in k: allocates diag_kSpec array used to store k spectra. Allocates diag_shells array and fills it with shell positions k^{shells}, used for binning of wave vectors when computing $k_\perp$ spectra. For spectra in m: allocates diag_mSpec array used to store m spectra. Called in init_start function

2.3 init.c File Reference

initialization module for alliance.

```
#include "init.h"
#include "distrib.h"
```

Macros

- #define RANK_IO 0

Functions

- void init_start (char *filename)
  initialization of ALLIANCE
- void init_printParameters ()
  parameter output
- void init_initEnums ()
  enumerator initialization
- void fill_rand (COMPLEX *ar1)
  fills the inital conditions randomly
- void fill_randM0 (COMPLEX *ar1)
  fill zeroth Hermite moment with random values
- void fill_randSingleKM (COMPLEX *ar1)
  fill single chosen wavevector and Hermite moment
- void init_conditions (COMPLEX *data)
  distribution function initialization
- double init_energySpec (double k, double m, double amp, double disp)
  returns energy spectrum

Variables

- enum adiabatic **kinetic**
- enum electromagnetic **systemType**
- enum initial **initialConditions**

2.3.1 Detailed Description

initialization module for alliance.

all the inititalization routines are here.

2.3.2 Macro Definition Documentation

2.3.2.1 RANK_IO

```
#define RANK_IO 0
```

defines rank of the processor used to output information to console

2.3.3 Function Documentation

2.3.3.1 fill_rand()

```
void fill_rand (
            COMPLEX * data )
```

fills the inital conditions randomly

Parameters

data	complex 6d array to fill initializes distribution with spectrum defined in init_energySpec This function is supposed to be used in-module only and should not be used elsewhere outside init.c file.

2.3.3.2 fill_randM0()

```
void fill_randM0 (
            COMPLEX * data )
```

fill zeroth Hermite moment with random values

Parameters

data	complex 6D array to fill

fills 0-th Hermite moment of a distribution function ar1 with random values This function is supposed to be used in-module only and should not be used elsewhere outside init.c file.

2.3.3.3 fill_randSingleKM()

```
void fill_randSingleKM (
            COMPLEX * ar1 )
```

fill single chosen wavevector and Hermite moment

Parameters

data	complex 6D array

initializes single wavevector and Hermite moment of a distribution function with random variable. This function is only for in-module use and should not be used elsewhere outside init.c file.

2.3.3.4 init_conditions()

```
void init_conditions (
            COMPLEX * data )
```

distribution function initialization

Parameters

data	complex 6D array

initializes distribution function with chosen method (see fill_rand, fill_randM0, fill_randSingleKM)

2.3.3.5 init_energySpec()

```
double init_energySpec (
            double k,
            double m,
            double amp,
            double disp )
```

returns energy spectrum

Parameters

k	a wavenumber at which spectrum is computed
m	Hermite moment at which amplitude is computed
amp	amplitude of the spectrum
disp	dispersion of the spectrum

computes spectrum of form $A \cdot k^2 exp(-2k^2/\sigma^2)$, where $\sigma = disp$, and $A = amp$ This function is supposed to be

used in-module only and should not be used elsewhere outside init.c file.

2.3.3.6 init_initEnums()

```
void init_initEnums ( )
```

enumerator initialization

initializes enumerators, which are then used to define if system is adiabatic or kinetic, electromagnetic or electrostatic, and type of initial conditions

2.3.3.7 init_printParameters()

```
void init_printParameters ( )
```

parameter output

prints parameters of the simulation

2.3.3.8 init_start()

```
void init_start (
            char * filename )
```

initialization of ALLIANCE

Parameters

filename	specifies parameter filename

initializes all the modules required for ALLIANCE to work.

Index