Alliance

1.0

Generated by Doxygen 1.9.1

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 array.c File Reference	3
2.1.1 Detailed Description	4
2.1.2 Function Documentation	4
2.1.2.1 get_flat_c()	4
2.1.2.2 get_flat_r()	4
2.1.2.3 get_flatIndexComplex3D()	5
2.1.2.4 getIndChi()	5
2.1.2.5 getIndChiBufEL_c()	5
2.1.2.6 getIndChiBufEL_r()	7
2.1.2.7 getIndChiBufEM_c()	7
2.1.2.8 getIndChiBufEM_r()	8
2.1.2.9 multiply_ar_c()	8
2.1.2.10 multiply_ar_r()	8
2.2 diagnostics.c File Reference	8
2.2.1 Detailed Description	9
2.2.2 Function Documentation	9
2.2.2.1 diag_compute()	9
2.2.2.2 diag_computeFreeEnergy()	10
2.2.2.3 diag_computeFreeEnergyFields()	10
2.2.2.4 diag_computeKSpectrum()	10
2.2.2.5 diag_computeMSpectrum()	11
2.2.2.6 diag_computeSpectra()	11
2.2.2.7 diag_getShells()	11
2.2.2.8 diag_initSpec()	12
2.3 distrib.c File Reference	12
2.3.1 Detailed Description	12
2.3.2 Function Documentation	12
2.3.2.1 distrib_enforceReality()	12
2.3.2.2 distrib_getG()	13
2.3.2.3 distrib_getH()	13
2.3.2.4 distrib_getXGrad()	13
2.3.2.5 distrib_getYGrad()	14
2.3.2.6 distrib_getZGrad()	14
2.3.2.7 distrib_setZeroNHalf()	14
2.4 equation.c File Reference	15
2.4.1 Detailed Description	15
2.4.2 Function Documentation	16
2.4.2.1 equation_getLinearTerm()	16

2.4.2.2 equation_getNonlinearElectromagnetic()	16		
2.4.2.3 equation_getNonlinearElectrostatic()	16		
2.4.2.4 equation_getNonlinearProduct()	17		
2.4.2.5 equation_getNonlinearTerm()	17		
2.5 fields.c File Reference	18		
2.5.1 Detailed Description	19		
2.5.2 Function Documentation	19		
2.5.2.1 fields_getA()	19		
2.5.2.2 fields_getAFromH()	20		
2.5.2.3 fields_getB()	20		
2.5.2.4 fields_getBFromH()	20		
2.5.2.5 fields_getChi()	21		
2.5.2.6 fields_getChiA()	21		
2.5.2.7 fields_getChiB()	21		
2.5.2.8 fields_getChiPhi()	21		
2.5.2.9 fields_getFields()	21		
2.5.2.10 fields_getFieldsFromH()	22		
2.5.2.11 fields_getGradX()	22		
2.5.2.12 fields_getGradY()	22		
2.5.2.13 fields_getPhi()	23		
2.5.2.14 fields_getPhiFromH()	23		
2.5.2.15 fields_init()	23		
2.5.2.16 fields_sendG()	23		
2.6 init.c File Reference	24		
2.6.1 Detailed Description	25		
2.6.2 Macro Definition Documentation	25		
2.6.2.1 RANK_IO	25		
2.6.3 Function Documentation	25		
2.6.3.1 fill_rand()	25		
2.6.3.2 fill_randM0()	25		
2.6.3.3 fill_randSingleKM()	26		
2.6.3.4 init_conditions()	26		
2.6.3.5 init_energySpec()	26		
2.6.3.6 init_initEnums()	27		
2.6.3.7 init_printParameters()	27		
2.6.3.8 init_start()	27		
2.7 parameters_io.c File Reference	27		
2.7.1 Detailed Description	28		
2.7.2 Function Documentation			
2.7.2.1 init_global_size()	28		
2.7.2.2 read_parameters()	28		
2.7.2.3 read_parametersFromFile()	28		

39

2.8 solver.c File Reference	29
2.8.1 Detailed Description	29
2.8.2 Function Documentation	29
2.8.2.1 solver_init()	29
2.8.2.2 solver_makeStep()	29
2.9 space_config.c File Reference	30
2.9.1 Detailed Description	31
2.9.2 Function Documentation	31
2.9.2.1 free_wavespace()	31
2.9.2.2 space_generateMSpace()	31
2.9.2.3 space_generateWaveSpace()	31
2.10 utils_fftw.c File Reference	31
2.10.1 Detailed Description	33
2.10.2 Function Documentation	33
2.10.2.1 cosinus()	33
2.10.2.2 dealiasing23()	33
2.10.2.3 fftw_c2r()	34
2.10.2.4 fftw_c2r_chi()	34
2.10.2.5 fftw_c2r_field()	34
2.10.2.6 fftw_copy_buffer_c()	34
2.10.2.7 fftw_copy_buffer_r()	35
2.10.2.8 fftw_copyChiBuf_c()	35
2.10.2.9 fftw_copyChiBuf_r()	35
2.10.2.10 fftw_copyFieldBuf_c()	35
2.10.2.11 fftw_copyFieldBuf_r()	36
2.10.2.12 fftw_kill()	36
2.10.2.13 fftw_normalise_chi_r()	36
2.10.2.14 fftw_normalise_data()	37
2.10.2.15 fftw_normalise_data_r()	37
2.10.2.16 fftw_normalise_field_r()	37
2.10.2.17 fftw_r2c()	37
2.10.2.18 fftw_r2c_chi()	37
2.10.2.19 fftw_r2c_field()	38
2.10.2.20 fftw_test_fill()	38

Index

Chapter 1

File Index

1.1 File List

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

array.c		
	Array manipulation module	3
diagnos	tics.c	
	Diagnostics module	8
distrib.c		
	Gyrokinetic distribution function module	12
equation	n.c	
	Equation module	15
fields.c		
	Field computation and manipulation module	18
init.c		
	Initialization module for alliance	24
paramet	ters_io.c	
	Reads inpuit parameters from parameter file provided by user	27
solver.c		
	Numerical solver	29
space_c		
	Space configuration module	30
utils_fftv	v.c	
	FFT module	31

2 File Index

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()

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 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()

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()

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

2.1.2.5 getIndChiBufEL_c()

```
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()

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()

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()

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()

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

2.1.2.10 multiply_ar_r()

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()

computes all diagnostics

Parameters

g	modified distribution function
h	distribution function
iter	current time step

2.2.2.2 diag_computeFreeEnergy()

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()

to be done later

Parameters

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

2.2.2.4 diag_computeKSpectrum()

Parameters

g modified gyrokinetic distribution fund	
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()

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()

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 distrib.c File Reference

```
gyrokinetic distribution function module
```

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

Functions

- void distrib_getH (COMPLEX *h, const COMPLEX *g)
 computes h from g
- $\bullet \ \ \text{void} \ \ \frac{\text{distrib_getG}}{\text{getG}} \ \ (\text{COMPLEX} \ *g, \ \text{const} \ \ \text{COMPLEX} \ *h) \\$

computes g from h

• void distrib_getXGrad (const COMPLEX *in, COMPLEX *out)

Computes gradient in kx direction.

void distrib_getYGrad (const COMPLEX *in, COMPLEX *out)

Computes gradient in ky direction.

• void distrib_getZGrad (const COMPLEX *in, COMPLEX *out)

Computes gradient in kz direction.

- void distrib_enforceReality (COMPLEX *f)
 - enforces reality condition on distribution function array
- void distrib_setZeroNHalf (COMPLEX *f)

sets all Nk/2 modes to zero

2.3.1 Detailed Description

gyrokinetic distribution function module

everything required to perform different manipulations to distribution functions

2.3.2 Function Documentation

2.3.2.1 distrib_enforceReality()

```
void distrib_enforceReality ( {\tt COMPLEX} * f )
```

enforces reality condition on distribution function array

2.3 distrib.c File Reference

Parameters

f complex array for which reality condition will be forced.

Enforces reality condition f(k) = conj(f(-k)) in plane kz = 0. For a given kx, it first checks where modes -kx are located using the mpi_w -where mpi_w -where

```
where_neg = mpi_whereIsX[kxNeg * 2];
```

If -kx is stored on a different processor, MPI_VECTOR with a 4D data slice f(kx,kz = 0) is sent to this processor, to the buffer array:

```
mpi_sendVector(&f[ind6D], buffer, where_pos, where_neg);
```

if the data stored on the same processor, no vector is being sent. Reality condition is fulfilled in a loop over all other coordinates:

```
f[ind6D_neg] = conj(buffer[ind6D_pos]);
```

2.3.2.2 distrib_getG()

```
distrib_getG (  \label{eq:complex} {\tt COMPLEX} \, * \, g, \\ {\tt const} \, {\tt COMPLEX} \, * \, h \, )
```

computes g from h

Parameters

g	complex array to store g
h	complex array with h

computes modified gyrokinetic distribution function g from gyrokinetic distribution function h. Please note that before calling this function gyrokinetic potentials must be computed

2.3.2.3 distrib_getH()

computes h from g

Parameters

h	complex array to store h
g	complex array with g

computes gyrokinetic distribution function h from modified gyrokinetic distribution function g. Please note that before calling this function gyrokinetic potentials must be computed

2.3.2.4 distrib getXGrad()

```
void distrib_getXGrad (
```

```
const COMPLEX * in,
COMPLEX * out )
```

Computes gradient in kx direction.

Parameters

in	complex array. Distribution function of which gradient will be taken
out	complex array, where gradient is stored

Computes gradient in kx direction as following: grad(f) = i * kx * f

2.3.2.5 distrib_getYGrad()

Computes gradient in ky direction.

Parameters

in	complex array. Distribution function of which gradient will be taken
out	complex array, where gradient is stored

Computes gradient in ky direction as following: grad(f) = i * ky * f

2.3.2.6 distrib_getZGrad()

Computes gradient in kz direction.

Parameters

in	complex array. Distribution function of which gradient will be taken
out	complex array, where gradient is stored

Computes gradient in kz direction as following: grad(f) = i * kz * f

2.3.2.7 distrib_setZeroNHalf()

```
void distrib_setZeroNHalf ( {\tt COMPLEX} \ * \ f \ )
```

sets all Nk/2 modes to zero

Parameters

f complex array

sets Nkx/2, Nky/2 and Nz/2 modes of distribution function to zero. Due to reality condition, for kz yhe last mode should be set to zero.

2.4 equation.c File Reference

equation module

```
#include "equation.h"
#include "array.h"
#include "fields.h"
#include "distrib.h"
```

Macros

- #define CHI EM 3
- #define CHI EL 1
- #define CHI_PHI 0
- #define CHI A 1
- #define CHI_B 2

Functions

void equation_getLinearTerm (const COMPLEX *in, const COMPLEX *plus_boundary, const COMPLEX *minus_boundary, COMPLEX *out)

computes linear term

• void equation_getNonlinearElectromagnetic (double *in, double *chiAr, double *out, double sign)

returns nonlinear electromagnetic term

void equation_getNonlinearElectrostatic (double *in, double *chiAr, double *out, double sign)

returns nonlinear electrostatic term

• void equation_getNonlinearProduct (double *in, double *chiAr, double *out, double sign)

chooses between computing electrostatic or electromagnetic term

 $\bullet \ \ void\ equation_getNonlinearTerm\ (const\ COMPLEX\ *h,\ COMPLEX\ *out)$

computes nonlinear term

void equation_getRHS (const COMPLEX *in_g, COMPLEX *in_h, COMPLEX *out)

2.4.1 Detailed Description

equation module

module required to compute RHS of the equation.

2.4.2 Function Documentation

2.4.2.1 equation_getLinearTerm()

computes linear term

Parameters

in	complex array
out	complex array
plus_boundary	complex array
minus_boundary	complex array

computes linear term ${\tt out}$ from distribution function ${\tt in}$.

2.4.2.2 equation_getNonlinearElectromagnetic()

returns nonlinear electromagnetic term

Parameters

in	input double array
chiAr	input double array
out	output double array
sign	should be 1 or -1

performs multiplication between input 6D complex array in and gyrokinetic potential array chiAr, in such way that the structure of the product is the same as nonlinear term of drift-kinetic equations. Used by equation_getNonlinearProduct. sign is used to determine the sign of the resulting product. See equation_getNonlinearTerm for explanation.

2.4.2.3 equation_getNonlinearElectrostatic()

```
double * chiAr,
double * out,
double sign )
```

returns nonlinear electrostatic term

Parameters

in	input double array
chiAr	input double array
out	output double array
sign	should be 1 or -1

 $see\ equation_getNonlinear Electromagnetic\ for\ explanation$

2.4.2.4 equation_getNonlinearProduct()

chooses between computing electrostatic or electromagnetic term

Parameters

in	input double array
chiAr	input double array
out	output double array
sign	should be 1 or -1

 $\label{thm:thm:depending} \begin{subarray}{l} depending on flag \verb| systemType| provided by user in parameter file, chooses between equation_getNonlinearElectrostatic and equation_getNonlinearElectromagnetic \\ \end{subarray}$

2.4.2.5 equation_getNonlinearTerm()

computes nonlinear term

Parameters

h	input complex array
out	output complex array

function returns nonlinear term. First it takes y gradient of distribution function h, and x gradient of gyrokinetic

```
potentials chi, and transforms them to real space:
```

```
distrib_getYGrad(h, fftw_hBuf); fields_getGradX(fftw_chiBuf); fftw_c2r(); fftw_c2r_chi(); after that, it computes \frac{\partial h}{\partial y} \frac{\partial x}{\partial x} part of the poisson brackets: equation_getNonlinearProduct((double *)fftw_hBuf, (double *)fftw_chiBuf, buffer, 1.); with the result stored in buffer after that, it computes x gradient of h and y gradient of gyrokinetic potential chi, and transforms results to real space: distrib_getXGrad(h, fftw_hBuf); fields_getGradY(fftw_chiBuf); fftw_c2r(); fftw_c2r_chi();
```

and computes second part of the poisson brackets $-\frac{\partial h}{\partial x}\frac{\partial \chi}{\partial y}$ and adds the result to buffer. buffer is then transformed back to Fourier space, and dealiasing is performed.

2.5 fields.c File Reference

field computation and manipulation module

```
#include "fields.h"
```

Macros

- #define CHI_PHI 0
- #define CHI A 1
- #define CHI_B 2

Functions

```
· void fields_init ()
     intializes fields

    void fields_getA (const COMPLEX *g)

      compute A field

    void fields_getB (const COMPLEX *g0, const COMPLEX *g1)

      computes B potential

    void fields_getPhi (const COMPLEX *g0, const COMPLEX *g1)

      computes phi potential

    void fields getFields (COMPLEX *g00, COMPLEX *g10, COMPLEX *g01)

      wrapper to get all the fields simultaneously

    void fields_getChi ()

     computes gyrokinetic potentials chi
void fields_getChiPhi ()
     computes chiPhi gyrokinetic potential from phi potential

    void fields getChiB ()

     computes chiB gyrokinetic potential from B potential

    void fields getChiA ()
```

computes chiA gyrokinetic potential from A potential

2.5 fields.c File Reference 19

```
    void fields_sendG (COMPLEX *g)
```

sends moments of gyrokinetic distribution function which are required to compute fields

• void fields_getFieldsFromH (COMPLEX *h00, COMPLEX *h10, COMPLEX *h01)

wrapper to get all the fields simultaneously, computed from gyrokinetic distribution function

void fields_getAFromH (const COMPLEX *h)

compute A field

• void fields_getBFromH (const COMPLEX *h0, const COMPLEX *h1)

computes B potential

void fields_getPhiFromH (const COMPLEX *h)

computes phi potential

void fields getGradX (COMPLEX *out)

computes chi gradient in x direction

void fields_getGradY (COMPLEX *out)

computes chi gradient in y direction

Variables

- · struct fields_fields fields_fields
- struct fields_chi fields_chi
- double * A denom
- double * qnvTsJ
- double * I B
- double * I_phi
- double * a_pot
- double * b_pot
- double * c_pot
- double * phiB_denom
- int * global_nm_index
- COMPLEX * g00
- COMPLEX * g10
- COMPLEX * g01

2.5.1 Detailed Description

field computation and manipulation module

Rerquired to compute $A_{||}(\mathbf{k}), B_{||}(\mathbf{k}), \phi(\mathbf{k})$ potentials, as well as gyrokinetic potentials $\chi_s^A(\mathbf{k}), \chi_s^B(\mathbf{k}), \chi_s^\phi(\mathbf{k})$

2.5.2 Function Documentation

2.5.2.1 fields_getA()

```
void fields_getA ( {\tt const~COMPLEX~*~g~)}
```

compute A field

Parameters

g

4D complex array (kx,ky,kz,s). Must be first Hermite and zeroth Laguerre moment of modified gyrokinetic distribution function g.

computes $A_{||}(\mathbf{k})$ potential from g^1_{s0} (g parameter)

2.5.2.2 fields_getAFromH()

```
void fields_getAFromH ( {\tt const~COMPLEX~*~h~)}
```

compute A field

Parameters

h 4D complex array (kx,ky,kz,s). Must be first Hermite and zeroth Laguerre moment of gyrokinetic distribution function h.

computes $A_{||}(\mathbf{k})$ potential from h^1_{s0} (h parameter)

2.5.2.3 fields_getB()

```
void fields_getB ( {\tt const~COMPLEX~*~g0,} {\tt const~COMPLEX~*~g1~)}
```

computes B potential

Parameters

g0	4D complex array (kx,ky,kz,s). Zeroth Hermite and Laguerre moment of modified gyrokinetic distribution
	function.
g1	4D complex array (kx,ky,kz,s). Zeroth Hermite and first Laguerre moment of the modified distribution
	function.

Computes $B_{\perp}(\mathbf{k})$ from g_{s0}^0 (g0 parameter) and g_{s0}^1 (g1 parameter).

2.5.2.4 fields_getBFromH()

computes B potential

Parameters

h0	4D complex array (kx,ky,kz,s). Zeroth Hermite and Laguerre moment of gyrokinetic distribution function.
h1	4D complex array (kx,ky,kz,s). Zeroth Hermite and first Laguerre moment of distribution function.

2.5 fields.c File Reference

Computes $B_{\perp}({\bf k})$ from h_{s0}^0 (h0 parameter) and h_{s0}^1 (h1 parameter).

2.5.2.5 fields_getChi()

```
void fields_getChi ( )
```

computes gyrokinetic potentials chi

Wrapper for functions fields_getChiPhi, fields_getChiA, fields_getChiB

2.5.2.6 fields_getChiA()

```
void fields_getChiA ( )
```

computes chiA gyrokinetic potential from A potential

computes $chi_s^A(\mathbf{k})$

2.5.2.7 fields_getChiB()

```
void fields_getChiB ( )
```

computes chiB gyrokinetic potential from B potential

computes $\chi_s^B(\mathbf{k})$

2.5.2.8 fields_getChiPhi()

```
fields_getChiPhi ( )
```

computes chiPhi gyrokinetic potential from phi potential

computes $chi_s^{\phi}(\mathbf{k})$

2.5.2.9 fields_getFields()

wrapper to get all the fields simultaneously

Parameters

g00	4D complex array (kx,ky,kz,s). Zeroth Hermite and Laguerre moment of modified gyrokinetic distribution function.
g10	4D complex array (kx,ky,kz,s). Must be first Hermite and zeroth Laguerre moment of modified gyrokinetic distribution function g.
Generate	4D complex array (kx,ky,kz,s). Zeroth Hermite and first Laguerre moment of the modified distribution function.

Wrapper for functions fields_getPhi, fields_getB, fields_getA.

2.5.2.10 fields getFieldsFromH()

wrapper to get all the fields simultaneously, computed from gyrokinetic distribution function

Parameters

h00	4D complex array (kx,ky,kz,s). Zeroth Hermite and Laguerre moment of gyrokinetic distribution function.
h10	4D complex array (kx,ky,kz,s). Must be first Hermite and zeroth Laguerre moment of gyrokinetic distribution
	function h.
h01	4D complex array (kx,ky,kz,s). Zeroth Hermite and first Laguerre moment of the distribution function.

Wrapper for functions fields_getPhiFromH, #fields_get_BFromH, fields_getAFromH.

2.5.2.11 fields getGradX()

computes chi gradient in x direction

Parameters

```
out output complex array of size (kx,ky,kz,s,Nfields).
```

computes gradient in x direction for chi potentials. Nfields<\tt> can be 1 or 3, and chosen automatically at start of the simulation depending on the simulation type (electrostatic or electromagnetic)

2.5.2.12 fields_getGradY()

computes chi gradient in y direction

Parameters

```
out output complex array of size (kx,ky,kz,s,Nfields).
```

computes gradient in y direction for chi potentials. Nfields < tt> can be 1 or 3, and chosen automatically at start of the simulation depending on the simulation type

2.5 fields.c File Reference 23

```
(electrostatic or electromagnetic)
```

2.5.2.13 fields getPhi()

computes phi potential

Parameters

g0	4D complex array (kx,ky,kz,s). Zeroth Hermite and Laguerre moment of modified gyrokinetic distribution
	function.
g1	4D complex array (kx,ky,kz,s). Zeroth Hermite and first Laguerre moment of the modified distribution
	function.

Computes $\phi(\mathbf{k})$ from g_{s0}^0 (g0 parameter) and g_{s0}^1 (g1 parameter).

2.5.2.14 fields_getPhiFromH()

```
void fields_getPhiFromH ( {\tt const~COMPLEX~*~h~)}
```

computes phi potential

Parameters

h | 4D complex array (kx,ky,kz,s). Zeroth Hermite and Laguerre moment of gyrokinetic distribution function.

Computes $\phi(\mathbf{k})$ from h_{s0}^0 (h parameter)

2.5.2.15 fields_init()

```
void fields_init ( )
```

intializes fields

pre-computes some comstants required to compute fields. Called in init_start function

2.5.2.16 fields_sendG()

```
fields_sendG ( {\tt COMPLEX} \ * \ g \ )
```

sends moments of gyrokinetic distribution function which are required to compute fields

Parameters

g complex array. Modified or non-modified gyrokinetic distribution function

```
sends g_{s0}^{1}(\mathbf{k})f, g_{s1}^{0}(\mathbf{k})f, g_{s0}^{0}(\mathbf{k})f
```

to all processes to compute potentials locally.

2.6 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

2.6 init.c File Reference 25

2.6.1 Detailed Description

initialization module for alliance.

all the inititalization routines are here.

2.6.2 Macro Definition Documentation

2.6.2.1 RANK_IO

```
#define RANK_IO 0
```

defines rank of the processor used to output information to console

2.6.3 Function Documentation

2.6.3.1 fill_rand()

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.

2.6.3.2 fill_randM0()

```
void fill_randM0 ( {\tt 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.

2.6.3.3 fill_randSingleKM()

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.

2.6.3.4 init_conditions()

```
void init_conditions ( {\tt COMPLEX} \ * \ {\tt data} \ )
```

distribution function initialization

Parameters

data	complex 6D array
------	------------------

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

2.6.3.5 init_energySpec()

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.

2.6.3.6 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

2.6.3.7 init_printParameters()

```
void init_printParameters ( )
```

parameter output

prints parameters of the simulation

2.6.3.8 init_start()

initialization of ALLIANCE

Parameters

lename specifies parameter filena	е
-----------------------------------	---

initializes all the modules required for ALLIANCE to work.

2.7 parameters_io.c File Reference

reads inpuit parameters from parameter file provided by user

```
#include "parameters_io.h"
#include "utils_fftw.h"
```

Macros

• #define VERBOSE 0

Functions

```
• void init_global_size ()
```

initializes global size of the 6D array

• void read_parameters (char *filename)

reads parameters from user parameter file.

• void read_parametersFromFile (char *filename)

Variables

• struct system_param parameters

2.7.1 Detailed Description

reads inpuit parameters from parameter file provided by user

2.7.2 Function Documentation

2.7.2.1 init_global_size()

```
void init_global_size ( )
```

initializes global size of the 6D array

 $initializes \verb| array_local_size| | \textit{structure with global simulation size}.$

2.7.2.2 read_parameters()

reads parameters from user parameter file.

Reads parameters from user parameter file. All the parameters are stored in the parameters structure

2.7.2.3 read_parametersFromFile()

read_parametersFromFile(char *filename):

2.8 solver.c File Reference

2.8 solver.c File Reference

```
numerical solver
```

```
#include "solver.h"
```

Macros

- #define **SOLVERTYPE** RK4
- #define IORANK 0

Functions

```
    void solver_init ()
        initializes solver
    void solver_makeStep (COMPLEX **g, COMPLEX *h)
        iterate solver forward
```

Variables

- enum solverType solverType
- struct solver solver
- struct rk4 rk4

2.8.1 Detailed Description

numerical solver

2.8.2 Function Documentation

2.8.2.1 solver_init()

```
void solver_init ( )
initializes solver
initializes solver with the solverType.
```

2.8.2.2 solver_makeStep()

```
void solver_makeStep ( \label{eq:complex} \mbox{COMPLEX ** } g \mbox{,} \mbox{COMPLEX ** } h \mbox{ )}
```

iterate solver forward

Parameters

g	address of the 6D complex array. Modified gyrokinetic distribution function
h	6D complex array. Gyrokinetic distribution function

solves one simulation time step

2.9 space_config.c File Reference

space configuration module

```
#include "space_config.h"
#include <complex.h>
```

Macros

• #define MINUS_I -1.j

Functions

```
    void space_init ()
        initializes wave space. Called in init_start() function.
    void space_generateWaveSpace ()
```

void space_generatevvaveopace (

generates wave space.

 void space_generateMSpace () generates Hermite space.

• void free_wavespace ()

deallocates all the arrays.

Variables

```
• double space_Lx = 100
```

- double space_Ly = 100
- double **space_Lz** = 100
- double * space_kx
- double * space_ky
- double * space_kz
- double * space_kPerp
- double * space_kPerp2
- double * space_kSq
- double * space_sqrtM
- size t * space_globalMIndex
- COMPLEX * space_iKx
- COMPLEX * space_iKy
- COMPLEX * space_iKz

2.9.1 Detailed Description

space configuration module

creates k and m spaces

2.9.2 Function Documentation

2.9.2.1 free_wavespace()

```
free_wavespace ( )
```

deallocates all the arrays.

to be added...

2.9.2.2 space_generateMSpace()

```
space_generateMSpace ( )
```

generates Hermite space.

to be added

2.9.2.3 space generateWaveSpace()

```
void space_generateWaveSpace ( )
```

generates wave space.

generates wave number arrays space_kx, space_ky, space_kz of lengths nkx,nky,nkz for a numerical box of size [lx, ly, lz] in kx,ky,kz directions as following:

```
[0, pi / lx, 2 pi / lx, ..., (n / 2 + 1) pi / lx, - (n / 2) pi / lx, ..., - pi / lx] generates arrays space_iKx, space_iKy, space_iKz, of lengths nkx,nky,nkz. These arrays are later used to compute gradients by fields_getGradX, fields_getGradY, distrib_getXGrad, distrib_getYGrad, distrib_getZGrad.
```

2.10 utils_fftw.c File Reference

```
FFT module.
```

```
#include "utils_fftw.h"
```

Macros

- #define FFTW_RANK 3
- · #define CHI EL 1
- #define CHI EM 3
- #define VERBOSE 0

Functions

```
• void fftw_init (MPI_Comm communicator)
     initializes fftw transform.
• void fftw r2c ()
     real to complex fft transform.
void fftw_c2r ()
     complex to real fft transform.
• void fftw r2c chi ()
     real to complex transform of chi potentials
· void fftw_c2r_chi ()
     complex to real transform of chi potentials
• void fftw r2c field ()
     real to complex transform of field potentials
void fftw_c2r_field ()
     complex to real transform of field potentials
• void fftw kill ()
     kills fftw

    void fftw_copy_buffer_r (double *to, double *from)

     copy 6D real array

    void fftw_copy_buffer_c (COMPLEX *to, COMPLEX *from)

     copy 6D complex array

    void fftw_copyChiBuf_r (double *ar1, double *ar2)

     copy 5D real array

    void fftw_copyChiBuf_c (COMPLEX *ar1, COMPLEX *ar2)

     copy 5D complex array.

    void fftw copyFieldBuf r (double *to, double *from)

     copy 3D real data array.

    void fftw_copyFieldBuf_c (COMPLEX *to, COMPLEX *from)

     copy 3D complex data array.
• double cosinus (double f, int ix)

    void fftw_test_fill (double *ar, double f)

    void fftw_normalise_data (COMPLEX *data)

    void fftw_normalise_data_r (double *data)

      normalise data.
• void fftw_normalise_chi_r (double *data)
     notmalase chi data
• void fftw_normalise_field_r (double *data)
     normalise real 3D data

    void dealiasing23 (COMPLEX *data_c)

     2/3 rule dealiasing
```

Variables

- ptrdiff_t size_c [3]
- ptrdiff_t size_r [3]
- · ptrdiff_t howmany
- ptrdiff_t howmany_chi
- · ptrdiff t local size
- ptrdiff_t local_n0
- ptrdiff_t local_0_start
- ptrdiff_t local_size_chi
- ptrdiff_t local_n0_chi
- ptrdiff_t local_0_start_chi
- ptrdiff_t local_size_field
- ptrdiff_t local_n0_field
- ptrdiff_t local_0_start_field
- fftw plan plan c2r
- fftw_plan plan_r2c
- fftw_plan plan_c2r_chi
- fftw_plan plan_r2c_chi
- fftw_plan plan_c2r_field
- fftw_plan plan_r2c_field
- COMPLEX * fftw_hBuf
- $COMPLEX * fftw_chiBuf$
- COMPLEX * fftw_field
- double fftw_norm
- void(* fftw_dealiasing)(COMPLEX *) = NULL
- int * global_nkx_index

2.10.1 Detailed Description

FFT module.

contains FFT related routines

2.10.2 Function Documentation

2.10.2.1 cosinus()

```
double cosinus ( \label{eq:double f, int } \mbox{double } f, \mbox{int } ix \mbox{)}
```

cosinus(double f,int ix)

2.10.2.2 dealiasing23()

```
void dealiasing23 ( {\tt COMPLEX * data\_c })
```

2/3 rule dealiasing

Parameters

data↩	complex 6D data array
_c	

2.10.2.3 fftw_c2r()

```
fftw_c2r ( )
```

complex to real fft transform.

Performs complex to real in-place fft transform on array fftw_hBuf. Used to transform 6D arrays (x,y,z,m,l,s).

2.10.2.4 fftw_c2r_chi()

```
void fftw_c2r_chi ( )
```

complex to real transform of chi potentials

Performs complex to real in-place fft transform on array fftw_chiBuf. Used to transform 5D arrays (kx,ky,kz,s,field).

2.10.2.5 fftw_c2r_field()

```
void fftw_c2r_field ( )
```

complex to real transform of field potentials

Performs complex to real in-place fft transform on array fftw_field. Used to transform 3D arrays (kx,ky,kz).

2.10.2.6 fftw_copy_buffer_c()

copy 6D complex array

Parameters

to	where to copy array	
from	array which will be copied	

copies complex data from array to array to

2.10.2.7 fftw_copy_buffer_r()

copy 6D real array

Parameters

to	where to copy array	
from	array which will be copied	

copies real data from array to array to

2.10.2.8 fftw_copyChiBuf_c()

copy 5D complex array.

Parameters

ar1	destination
ar2	source

copies complex χ array from ar1 to ar2.

2.10.2.9 fftw_copyChiBuf_r()

copy 5D real array

Parameters

ar1	destination
ar2	source

copies real χ array from ar1 to ar2.

2.10.2.10 fftw_copyFieldBuf_c()

```
fftw_copyFieldBuf_c (
```

```
COMPLEX * to,
COMPLEX * from )
```

copy 3D complex data array.

Parameters

to	
from	copies 3D complex data array from to

2.10.2.11 fftw_copyFieldBuf_r()

copy 3D real data array.

Parameters

to	
from	copies 3D data array from to

2.10.2.12 fftw_kill()

```
void fftw_kill ( )
```

kills fftw

to be added

2.10.2.13 fftw_normalise_chi_r()

notmalase chi data

Parameters

data	5D real data array

normalises data by #fftw_norm.

2.10.2.14 fftw_normalise_data()

fftw_normalise_data(double *data)

2.10.2.15 fftw_normalise_data_r()

normalise data.

Parameters

```
data 6D data array
```

normalises data by #fftw_norm.

2.10.2.16 fftw_normalise_field_r()

normalise real 3D data

Parameters

data	3D real array
------	---------------

normalises data by #fftw_norm.

2.10.2.17 fftw_r2c()

```
fftw_r2c ( )
```

real to complex fft transform.

Performs real to complex in-place fft transform of on array $fftw_hBuf$. Used to transform 6D arrays (kx,ky,kz,m,l,s).

2.10.2.18 fftw_r2c_chi()

```
void fftw_r2c_chi ( )
```

real to complex transform of chi potentials

 $Performs \ real \ to \ complex \ in-place \ fft \ transform \ on \ array \ \texttt{fftw_chiBuf.} \ Used \ to \ transform \ 5D \ arrays \ (x,y,z,s,field).$

2.10.2.19 fftw_r2c_field()

```
void fftw_r2c_field ( )
```

real to complex transform of field potentials

Performs real to complex in-place fft transform on array fftw_field. Used to transform 3D arrays (x,y,z).

2.10.2.20 fftw_test_fill()

fftw_test_fill(double *ar,double f)

Index

```
array.c, 3
                                                         distrib_enforceReality
                                                              distrib.c, 12
     get_flat_c, 4
     get flat r, 4
                                                         distrib getG
     get flatIndexComplex3D, 5
                                                              distrib.c, 13
     getIndChi, 5
                                                         distrib_getH
     getIndChiBufEL_c, 5
                                                              distrib.c, 13
     getIndChiBufEL r, 7
                                                         distrib getXGrad
     getIndChiBufEM_c, 7
                                                              distrib.c, 13
                                                         distrib_getYGrad
     getIndChiBufEM_r, 8
     multiply ar c, 8
                                                              distrib.c, 14
     multiply ar r, 8
                                                         distrib getZGrad
                                                              distrib.c, 14
cosinus
                                                         distrib_setZeroNHalf
     utils fftw.c, 33
                                                              distrib.c, 14
dealiasing23
                                                         equation.c, 15
     utils fftw.c, 33
                                                              equation getLinearTerm, 16
diag compute
                                                              equation getNonlinearElectromagnetic, 16
     diagnostics.c, 9
                                                              equation_getNonlinearElectrostatic, 16
diag computeFreeEnergy
                                                              equation_getNonlinearProduct, 17
     diagnostics.c, 10
                                                              equation_getNonlinearTerm, 17
diag_computeFreeEnergyFields
                                                         equation_getLinearTerm
     diagnostics.c, 10
                                                              equation.c, 16
diag_computeKSpectrum
                                                         equation_getNonlinearElectromagnetic
     diagnostics.c. 10
                                                              equation.c, 16
diag_computeMSpectrum
                                                          equation_getNonlinearElectrostatic
     diagnostics.c, 11
                                                              equation.c, 16
diag computeSpectra
                                                          equation_getNonlinearProduct
     diagnostics.c, 11
                                                              equation.c, 17
diag_getShells
                                                         equation_getNonlinearTerm
     diagnostics.c, 11
                                                              equation.c, 17
diag_initSpec
     diagnostics.c, 11
                                                         fftw c2r
diagnostics.c, 8
                                                              utils_fftw.c, 34
     diag compute, 9
                                                         fftw c2r chi
     diag computeFreeEnergy, 10
                                                              utils fftw.c, 34
     diag computeFreeEnergyFields, 10
                                                         fftw c2r field
     diag computeKSpectrum, 10
                                                              utils fftw.c, 34
     diag computeMSpectrum, 11
                                                         fftw copy buffer c
     diag computeSpectra, 11
                                                              utils fftw.c, 34
     diag_getShells, 11
                                                         fftw_copy_buffer_r
                                                              utils_fftw.c, 34
     diag_initSpec, 11
distrib.c, 12
                                                         fftw copyChiBuf c
     distrib_enforceReality, 12
                                                              utils fftw.c, 35
     distrib_getG, 13
                                                         fftw_copyChiBuf_r
                                                              utils fftw.c, 35
     distrib getH, 13
     distrib getXGrad, 13
                                                         fftw copyFieldBuf c
     distrib getYGrad, 14
                                                              utils fftw.c, 35
     distrib_getZGrad, 14
                                                         fftw_copyFieldBuf_r
                                                              utils_fftw.c, 36
     distrib_setZeroNHalf, 14
```

40 INDEX

fftw_kill	fields.c, 22
utils_fftw.c, 36	fields_getPhi
fftw_normalise_chi_r	fields.c, 23
utils_fftw.c, 36	fields_getPhiFromH
fftw_normalise_data	fields.c, 23
utils_fftw.c, 36	fields_init
fftw normalise data r	fields.c, 23
utils fftw.c, 37	fields sendG
fftw normalise field r	fields.c, 23
utils fftw.c, 37	fill rand
fftw r2c	init.c, 25
utils_fftw.c, 37	fill randM0
fftw_r2c_chi	init.c, 25
utils_fftw.c, 37	fill_randSingleKM
fftw_r2c_field	init.c, 26
utils_fftw.c, 37	free_wavespace
fftw_test_fill	space_config.c, 31
utils_fftw.c, 38	got flot o
fields.c, 18	get_flat_c
fields_getA, 19	array.c, 4
fields_getAFromH, 20	get_flat_r
fields_getB, 20	array.c, 4
fields_getBFromH, 20	get_flatIndexComplex3D
fields_getChi, 21	array.c, 5
fields_getChiA, 21	getIndChi
fields_getChiB, 21	array.c, 5
fields_getChiPhi, 21	getIndChiBufEL_c
fields getFields, 21	array.c, 5
fields getFieldsFromH, 22	getIndChiBufEL_r
fields_getGradX, 22	array.c, 7
fields_getGradY, 22	getIndChiBufEM c
fields_getPhi, 23	array.c, 7
fields getPhiFromH, 23	getIndChiBufEM r
fields_init, 23	array.c, 8
fields_sendG, 23	
fields getA	init.c, 24
	fill_rand, 25
fields.c, 19	fill randM0, 25
fields_getAFromH	fill_randSingleKM, 26
fields.c, 20	init_conditions, 26
fields_getB	init_energySpec, 26
fields.c, 20	init_initEnums, 27
fields_getBFromH	init_printParameters, 27
fields.c, 20	init start, 27
fields_getChi	- :
fields.c, 21	RANK_IO, 25
fields_getChiA	init_conditions
fields.c, 21	init.c, 26
fields_getChiB	init_energySpec
fields.c, 21	init.c, 26
fields_getChiPhi	init_global_size
fields.c, 21	parameters_io.c, 28
fields_getFields	init_initEnums
fields.c, 21	init.c, 27
fields_getFieldsFromH	init_printParameters
fields.c, 22	init.c, 27
fields_getGradX	init_start
fields.c, 22	init.c, 27
fields.c, 22 fields_getGradY	
ilisius_ystalau i	multiply_ar_c

INDEX 41

```
array.c, 8
multiply_ar_r
    array.c, 8
parameters_io.c, 27
    init_global_size, 28
    read_parameters, 28
    read parametersFromFile, 28
RANK IO
    init.c, 25
read parameters
    parameters_io.c, 28
read_parametersFromFile
    parameters_io.c, 28
solver.c, 29
     solver_init, 29
     solver_makeStep, 29
solver_init
    solver.c, 29
solver makeStep
    solver.c, 29
space_config.c, 30
    free wavespace, 31
     space generateMSpace, 31
    space_generateWaveSpace, 31
space_generateMSpace
    space config.c, 31
space_generateWaveSpace
     space_config.c, 31
utils_fftw.c, 31
    cosinus, 33
    dealiasing23, 33
    fftw c2r, 34
    fftw_c2r_chi, 34
    fftw_c2r_field, 34
    fftw_copy_buffer_c, 34
     fftw_copy_buffer_r, 34
     fftw_copyChiBuf_c, 35
     fftw_copyChiBuf_r, 35
     fftw_copyFieldBuf_c, 35
     fftw_copyFieldBuf_r, 36
    fftw_kill, 36
    fftw_normalise_chi_r, 36
    fftw normalise data, 36
    fftw normalise data r, 37
    fftw_normalise_field_r, 37
    fftw_r2c, 37
     fftw r2c chi, 37
     fftw_r2c_field, 37
    fftw_test_fill, 38
```