Kolmogorov41

Generated by Doxygen 1.8.6

Wed Jan 22 2020 20:44:10

Contents

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 src/Kolmogorov41.cc File Reference

Code to compute structure functions using velocity or scalar field data.

```
#include "h5si.h"
#include <yaml-cpp/yaml.h>
#include <iostream>
#include <fstream>
#include <hdf5.h>
#include <sstream>
#include <blitz/array.h>
#include <omp.h>
#include <mpi.h>
#include <sys/time.h>
#include <limits.h>
#include <unistd.h>
```

Functions

- void Read_para ()

  Function to open the yaml file and parse the parameters.
- void write_2D (Array< double, 2 > A, string file)

  Function to write the structure functions as function of l as a 2D hdf5 file.
- void write_3D (Array< double, 3 > A, string file, int q)

  Function to write the structure functions as function of lx,lz as a 2D hdf5 file.
- void write_4D (Array< double, 4 > A, string file, int q)

  Function to write the structure functions as function of lx,ly,lz as a 3D hdf5 file.
- void read_2D (Array< double, 2 > A, string file)

  Function to read a 2D field from an hdf5 file.
- string int_to_str (int num)

  Function to convert an integer type value to string.
- void compute_time_elapsed (timeval start_t, timeval end_t, double &elapsed)

  Function to compute the time elapsed.
- double powInt (double x, int n)

  Function which conducts exponentiation with an integer as an exponent.
- void read_3D (Array< double, 3 > A, string file)

  Function to read a 3D field from an hdf5 file.

- void SFunc2D (Array< double, 2 > Ux, Array< double, 2 > Uz, Array< double, 2 > &SF_Node, Array< double, 2 > &SF_Node_p, Array< double, 2 > &counter_Node)

  Function to calculate structure functions using 2D velocity field.
- void SFunc2D (Array< double, 2 > Ux, Array< double, 2 > Uz, Array< double, 2 > &SF_Node, Array< double, 2 > &SF_p_Node, Array< double, 2 > &counter_Node, Array< double, 3 > &SF_Grid2D_pll_Node, Array< double, 3 > &SF_Grid2D_perp_Node, Array< double, 3 > &counter_Grid2D_Node)

  Function to calculate structure functions using 2D velocity field as function of (lx, lz) in addition to the structure functions as function of (l).
- void SFunc_long_2D (Array< double, 2 > Ux, Array< double, 2 > Uz, Array< double, 2 > &SF_Node, Array< double, 2 > &counter_Node)

  A less computationally intensive function to calculate only the longitudinal structure functions using 2D velocity field.
- void SFunc_long_2D (Array< double, 2 > Ux, Array< double, 2 > Uz, Array< double, 2 > &SF_Node, Array< double, 2 > &counter_Node, Array< double, 3 > &SF_Grid2D_pll_Node, Array< double, 3 > &counter_Grid2D_Node)

  A less computationally intensive function to calculate only the longitudinal structure functions as functions of (lx,lz) in addition to function of (l) using 2D velocity field.
- void SFunc3D (Array< double, 3 > Ux, Array< double, 3 > Uy, Array< double, 3 > Uz, Array< double, 2 > &SF_Node, Array< double, 2 > &SF_Node_p, Array< double, 2 > &counter_Node)

  Function to calculate structure functions using 3D velocity field.
- void SFunc3D (Array< double, 3 > Ux, Array< double, 3 > Uy, Array< double, 3 > Uz, Array< double, 2 > &SF_Node, Array< double, 2 > &SF_p_Node, Array< double, 2 > &counter_Node, Array< double, 4 > &SF_Grid_pll_Node, Array< double, 4 > &SF_Grid_perp_Node, Array< double, 4 > &counter_Grid_Node)

  Function to calculate structure functions using 3D velocity field as function of (lx, ly, lz) in addition to the structure functions as function of (l).
- void SFunc_long_3D (Array< double, 3 > Ux, Array< double, 3 > Uy, Array< double, 3 > Uz, Array< double, 2 > &SF_Node,Array< double, 2 > &counter_Node)

  A less computationally intensive function to calculate only the longitudinal structure functions using 3D velocity field.
- void SFunc_long_3D (Array< double, 3 > Ux, Array< double, 3 > Uy, Array< double, 3 > Uz, Array< double, 2 > &SF_Node, Array< double, 2 > &counter_Node, Array< double, 4 > &SF_Grid_pll_Node, Array< double, 4 > &counter_Grid_Node)

  A less computationally intensive function to calculate only the longitudinal structure functions as functions of (lx,ly,lz) in addition to function of (l) using 3D velocity field.
- void Read_Init (Array< double, 2 > &Ux, Array< double, 2 > &Uz)

  Function to assign an exponential function to the 2D velocity field.
- void Read_Init (Array< double, 3 > &Ux, Array< double, 3 > &Uy, Array< double, 3 > &Uz)

  Function to assign an exponential function to the 3D velocity field.
- void Read_Init (Array< double, 2 > &T)

  Function to assign an exponential function to a 2D scalar field.
- void Read_Init (Array< double, 3 > &T)

  Function to assign an exponential function to a 3D scalar field.
- void SF_scalar_3D (Array< double, 3 > T, Array< double, 2 > &SF_Node, Array< double, 2 > &counter_Node)

  Function to calculate structure functions using 3D scalar field.
- void SF_scalar_3D (Array< double, 3 > T, Array< double, 2 > &SF_Node, Array< double, 2 > &counter_Node, Array< double, 4 > &SF_Grid_Node, Array< double, 4 > &counter_Grid_Node)

  Function to calculate structure functions using 3D scalar field as function of (lx, ly, lz) in addition to the structure functions as function of (l).
- void SF_scalar_2D (Array< double, 2 > T, Array< double, 2 > &SF_Node, Array< double, 2 > &counter_Node)

  Function to calculate structure functions using 2D scalar field.
- void SF_scalar_2D (Array< double, 2 > T, Array< double, 2 > &SF_Node, Array< double, 2 > &counter_Node, Array< double, 3 > &SF_Grid_Node, Array< double, 3 > &counter_Grid_Node)

  Function to calculate structure functions using 2D scalar field as function of (lx, lz) in addition to the structure functions as function of (l).

- int main (int argc, char ∗argv[])

  The main function of the "Strunc" code.
- void magnitude (TinyVector< double, 3 > A, double &mag)

  Function to compute the magnitude of a 3D vector A.
- void magnitude (TinyVector< double, 2 > A, double &mag)

  Function to compute the magnitude of a 2D vector A.

Variables

- Array< double, 3 > T

  3D array storing the scalar field.
- Array< double, 3 > V1

  3D array storing the x-component of the 3D velocity field.
- Array< double, 3 > V2

  3D array storing the y-component of the 3D velocity field.
- Array< double, 3 > V3

  3D array storing the z-component of the 3D velocity field.
- Array< double, 2 > T_2D

  2D array storing the 2D scalar field.
- Array< double, 2 > V1_2D

  3D array storing the x-component of the 3D velocity field.
- Array< double, 2 > V3_2D

  2D array storing the z-component of the 2D velocity field.
- Array< double, 2 > SF

  2D array storing the computed longitudinal structure functions or scalar structure functions and their corresponding orders.
- Array< double, 2 > SF_perp

  2D array storing the computed transverse structure functions and their corresponding orders.
- Array< double, 2 > counter

  2D array storing the values to divide SF and SF_perp for averaging.
- Array< double, 4 > SF_Grid_pll

  4D array storing the computed longitudinal structure functions as function of (lx, ly, lz, p), where p is the order.
- Array< double, 4 > SF_Grid_perp

  4D array storing the computed transverse structure functions as function of (lx, ly, lz, p), where p is the order.
- Array< double, 4 > counter_Grid

  4D array storing the counter array for averaging SF_Grid_pll and SF_Grid_perp
- Array< double, 3 > SF_Grid2D_pll

  3D array storing the computed longitudinal structure functions as function of (lx, lz, p), where p is the order.
- Array< double, 3 > SF_Grid2D_perp

  3D array storing the computed transverse structure functions as function of (lx, ly, lz, p), where p is the order.
- Array< double, 3 > counter_Grid2D

  3D array storing the counter array for averaging SF_Grid_pll and SF_Grid_perp
- bool two_dimension_switch

  This variable decides whether the structure functions are to be calculated using 2D or 3D velocity field data.
- bool scalar_switch

  This variable decides whether the scalar or velocity structure functions are to be evaluated.
- bool grid_switch

  This variable decides whether the structure functions as function of (lx, ly, lz), or of (lx, lz) in case of 2D, needs to be calculated in addition to the structure functions as function of (l).
- int Nx

  Number of gridpoints in the x direction.

- int Ny

  Number of gridpoints in the y direction.
- int Nz

  Number of gridpoints in the z direction.
- int num

  Number of OpenMP threads.
- int Nr

  Total number of gridpoints in the diagonal direction.
- int q1

  The first order of the range of orders of the structure functions to be computed.
- int q2

  The last order of the range of orders of the structure functions to be computed.
- int field_procedure

  This variable decides whether the code needs to generate velocity field data or read velocity field data from HDF5 files.
- bool longitudinal

  This variable decides whether to calculate both transverse and longitudinal structure functions or only the longitudinal structure functions using a less computationally expensive technique.
- double dx

  This variable stores the distance between two consecutive gridpoints in the x direction.
- double dy

  This variable stores the distance between two consecutive gridpoints in the y direction.
- double dz

  This variable stores the distance between two consecutive gridpoints in the z direction.
- int rank_mpi

  This variable stores the rank of the MPI process.
- double lx

  This variable stores the length of the domain.
- double ly

  This variable stores the breadth of the domain.
- double lz

  This variable stores the height of the domain.
- int P

  This variable stores the number of the MPI process.

2.1.1 Detailed Description

Code to compute structure functions using velocity or scalar field data.

Author

Shashwat Bhattacharya, Shubhadeep Sadhukhan

Date

Jan 2020

Copyright

2.1.2 Function Documentation

2.1.2.1 void compute_time_elapsed (timeval *start_t*, timeval *end_t*, double & *elapsed*)

Function to compute the time elapsed.

This function computes the time elapsed in seconds between a start point and an end point during the execution of the program

Parameters

start_t	is the time corresponding to the start point
end_t	is the time corresponding to the end point
elapsed	stores the time elapsed in seconds

2.1.2.2 string int_to_str (int *num*)

Function to convert an integer type value to string.

This function converts an integer type value to a string.

Parameters

num	is the integer value of to be converted.

Returns

The value as a string.

2.1.2.3 void magnitude (TinyVector< double, 3 > *A*, double & *mag*)

Function to compute the magnitude of a 3D vector A.

Parameters

A	is a tiny vector representing the 3D velocity field.
mag	is the variable that stores the magnitude calculated in this function.

2.1.2.4 void magnitude (TinyVector< double, 2 > *A*, double & *mag*)

Function to compute the magnitude of a 2D vector A.

Parameters

A	is a tiny vector representing the 2D velocity field.
mag	is the variable that stores the magnitude calculated in this function.

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

The main function of the "Strunc" code.

This function is the main function of the "Strunc" code for computing the velocity and scalar structure functions. The MPI decomposition and integration are also carried out in this function.

2.1.2.6 double powInt (double *x*, int *n*)

Function which conducts exponentiation with an integer as an exponent.

This function calculates x raised to the power n, where n is an integer. This function is faster than the standard pow(x,n) function for n>2. Note that this function cannot accept a non-integer exponent.

Parameters

x	is the base of double-precision floating point datatype.
n	is the exponent of integer datatype

Returns

The value as a string.

2.1.2.7 void read_2D (Array< double, 2 > *A*, string *file*)

Function to read a 2D field from an hdf5 file.

This function reads an hdf5 file containing a 2D field, which can be the x or z component of a 2D velocity field. The dimensions of the 2D field is $(N_x \times N_z)$, where N_x and N_z are the number of gridpoints in x and z directions respectively. The hdf5 file should have only one dataset, and the names of the hdf5 file and the dataset must be identical. This function make use of the H5SI library for reading the hdf5 file.

Parameters

A	is the 2D array to store the field that is read from the file.
file	is a string storing the name of the file to be read.

2.1.2.8 void read_3D (Array< double, 3 > *A*, string *file*)

Function to read a 3D field from an hdf5 file.

This function reads an hdf5 file containing a 4D field, which can be the x y, or z component of a 3D velocity field. The dimensions of the 3D field is $(N_x \times N_y \times N_z)$, where N_x, N_y, and N_z are the number of gridpoints in x, y, and z directions respectively. The hdf5 file should have only one dataset, and the names of the hdf5 file and the dataset must be identical. This function make use of the H5SI library for reading the hdf5 file.

Parameters

A	is the 3D array to store the field that is read from the file.
file	is a string storing the name of the file to be read.

2.1.2.9 void Read_Init (Array< double, 2 > & *Ux*, Array< double, 2 > & *Uz*)

Function to assign an exponential function to the 2D velocity field.

This function assigns the following exponential function to the 2D velocity field. $U_x = x, \quad U_z = z$.

Parameters

Ux	is a 2D array representing the x-component of 2D velocity field.
Uz	is a 2D array representing the z-component of 2D velocity field.

2.1.2.10 void Read_Init (Array< double, 3 > & *Ux*, Array< double, 3 > & *Uy*, Array< double, 3 > & *Uz*)

Function to assign an exponential function to the 3D velocity field.

This function assigns the following exponential function to the 3D velocity field. $U_x = x, \quad U_y = y, \quad U_z = z$.

Parameters

Ux	is a 3D array representing the x-component of 3D velocity field.
Uy	is a 3D array representing the y-component of 3D velocity field.
Uz	is a 3D array representing the z-component of 3D velocity field.

2.1.2.11 void Read_Init (Array< double, 2 > & *T*)

Function to assign an exponential function to a 2D scalar field.

This function assigns the following exponential function to the scalar field. $T = x + z$

Parameters

T	is a 2D array representing the x-component of 2D velocity field.

2.1.2.12 void Read_Init (Array< double, 3 > & *T*)

Function to assign an exponential function to a 3D scalar field.

This function assigns the following exponential function to the scalar field. $T = x + y + z$

Parameters

T	is a 3D array representing the x-component of 2D velocity field.

2.1.2.13 void Read_para ( )

Function to open the yaml file and parse the parameters.

The function opens the parameters.yaml file and parses the simulation parameters into its member variables that are publicly accessible.

2.1.2.14 void SF_scalar_2D (Array< double, 2 > *T,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *counter_Node*)

Function to calculate structure functions using 2D scalar field.

The following function computes both the nodal structure functions of 2D scalar field using four nested for-loops. The outer two for-loops correspond to the vector $\mathbf{r}$ and the inner two loops correspond to the vector $\mathbf{r}+\mathbf{l}$. The second for-loop is parallelized using OpenMP.

Parameters

T	is a 2D array representing the scalar field
SF_Node	is a 2D array containing the values of the nodal structure functions for a range of orders specified by the user.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node so as to get the average.

2.1.2.15 void SF_scalar_2D (Array< double, 2 > *T,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *counter_Node,* Array< double, 3 > & *SF_Grid_Node,* Array< double, 3 > & *counter_Grid_Node*)

Function to calculate structure functions using 2D scalar field as function of (lx, lz) in addition to the structure functions as function of (l).

The following function computes both the nodal structure functions of 2D scalar field using four nested for-loops. The outer two for-loops correspond to the vector $\mathbf{r}$ and the inner two loops correspond to the vector $\mathbf{r}+\mathbf{l}$. The

second for-loop is parallelized using OpenMP.

Parameters

T	is a 2D array representing the scalar field
SF_Node	is a 2D array containing the values of the nodal structure functions as function of (l) for a range of orders specified by the user.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node so as to get the average.
SF_Grid_Node	is a 3D array containing the values of the nodal structure functions as function of (lx,lz) for a range of orders specified by the user.
counter_Grid_Node	is a 3D array containing the numbers for dividing the values of SF_Grid_pll_Node so as to get the average.

2.1.2.16 void SF_scalar_3D (Array< double, 3 > *T,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *counter_Node*)

Function to calculate structure functions using 3D scalar field.

The following function computes both the nodal structure functions of 3D scalar field using six nested for-loops. The outer three for-loops correspond to the vector $\mathbf{r}$ and the inner three loops correspond to the vector $\mathbf{r}+\mathbf{l}$. The second for-loop is parallelized using OpenMP.

Parameters

T	is a 3D array representing the scalar field
SF_Node	is a 2D array containing the values of the nodal structure functions for a range of orders specified by the user.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node so as to get the average.

2.1.2.17 void SF_scalar_3D (Array< double, 3 > *T,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *counter_Node,* Array< double, 4 > & *SF_Grid_Node,* Array< double, 4 > & *counter_Grid_Node*)

Function to calculate structure functions using 3D scalar field as function of (lx, ly, lz) in addition to the structure functions as function of (l).

The following function computes both the nodal structure functions of 3D scalar field using six nested for-loops. The outer three for-loops correspond to the vector $\mathbf{r}$ and the inner three loops correspond to the vector $\mathbf{r}+\mathbf{l}$. The second for-loop is parallelized using OpenMP.

Parameters

T	is a 3D array representing the scalar field
SF_Node	is a 2D array containing the values of the nodal structure functions as function of (l) for a range of orders specified by the user.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node so as to get the average.
SF_Grid_Node	is a 4D array containing the values of the nodal structure functions as function of (lx,ly,lz) for a range of orders specified by the user.
counter_Grid_Node	is a 4D array containing the numbers for dividing the values of SF_Grid_pll_Node so as to get the average.

2.1.2.18 void SFunc2D (Array< double, 2 > *Ux,* Array< double, 2 > *Uz,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *SF_Node_p,* Array< double, 2 > & *counter_Node*)

Function to calculate structure functions using 2D velocity field.

The following function computes both the nodal longitudinal and transverse structure functions of 2D velocity field using four nested for-loops. The outer two for-loops correspond to the vector $\mathbf{r}$ and the inner two loops correspond

to the vector $\mathbf{r}+\mathbf{l}$. The second for-loop is parallelized using OpenMP.

Parameters

Ux	is a 2D array representing the x-component of velocity field
Uz	is a 2D array representing the z-component of velocity field
SF_Node	is a 2D array containing the values of the nodal longitudinal structure functions for a range of orders specified by the user.
SF_Node_p	is a 2D array containing the values of the nodal transverse structure functions for orders q1 to q2.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node and SF_Node_p so as to get the average.

2.1.2.19 void SFunc2D (Array< double, 2 > *Ux,* Array< double, 2 > *Uz,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *SF_p_Node,* Array< double, 2 > & *counter_Node,* Array< double, 3 > & *SF_Grid2D_pll_Node,* Array< double, 3 > & *SF_Grid2D_perp_Node,* Array< double, 3 > & *counter_Grid2D_Node*)

Function to calculate structure functions using 2D velocity field as function of (lx, lz) in addition to the structure functions as function of (l).

The following function computes both the nodal longitudinal and transverse structure functions of 2D velocity field using six nested for-loops. The outer three for-loops correspond to the vector $\mathbf{r}$ and the inner three loops correspond to the vector $\mathbf{r}+\mathbf{l}$. The second for-loop is parallelized using OpenMP.

Parameters

Ux	is a 2D array representing the x-component of velocity field
Uz	is a 2D array representing the z-component of velocity field
SF_Node	is a 2D array containing the values of the nodal longitudinal structure functions as function of (l) for a range of orders specified by the user.
SF_p_Node	is a 2D array containing the values of the nodal transverse structure functions for as function of (l) for a range of orders specified by the user.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node and SF_Node_p so as to get the average.
SF_Grid2D_pll_Node	is a 3D array containing the values of the nodal longitudinal structure functions as function of (lx,lz) for a range of orders specified by the user.
SF_Grid2D_perp_Node	is a 3D array containing the values of the nodal transverse structure functions for as function of (lx,lz) for a range of orders specified by the user.
counter_Grid2D_Node	is a 3D array containing the numbers for dividing the values of SF_Grid_pll_Node and SF_Grid_perp_Node_p so as to get the average.

2.1.2.20 void SFunc3D (Array< double, 3 > *Ux,* Array< double, 3 > *Uy,* Array< double, 3 > *Uz,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *SF_Node_p,* Array< double, 2 > & *counter_Node*)

Function to calculate structure functions using 3D velocity field.

The following function computes both the nodal longitudinal and transverse structure functions of 3D velocity field using six nested for-loops. The outer three for-loops correspond to the vector $\mathbf{r}$ and the inner three loops correspond to the vector $\mathbf{r}+\mathbf{l}$. The second for-loop is parallelized using OpenMP.

Parameters

Ux	is a 3D array representing the x-component of velocity field
Uy	is a 3D array representing the y-component of velocity field
Uz	is a 3D array representing the z-component of velocity field
SF_Node	is a 2D array containing the values of the nodal longitudinal structure functions for a range of orders specified by the user.

SF_Node_p	is a 2D array containing the values of the nodal transverse structure functions for orders q1 to q2.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node and SF_Node_p so as to get the average.

2.1.2.21 void SFunc3D (Array< double, 3 > *Ux,* Array< double, 3 > *Uy,* Array< double, 3 > *Uz,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *SF_p_Node,* Array< double, 2 > & *counter_Node,* Array< double, 4 > & *SF_Grid_pll_Node,* Array< double, 4 > & *SF_Grid_perp_Node,* Array< double, 4 > & *counter_Grid_Node*)

Function to calculate structure functions using 3D velocity field as function of (lx, ly, lz) in addition to the structure functions as function of (l).

The following function computes both the nodal longitudinal and transverse structure functions of 3D velocity field using six nested for-loops. The outer three for-loops correspond to the vector $\mathbf{r}$ and the inner three loops correspond to the vector $\mathbf{r}+\mathbf{l}$. The second for-loop is parallelized using OpenMP.

Parameters

Ux	is a 3D array representing the x-component of velocity field
Uy	is a 3D array representing the y-component of velocity field
Uz	is a 3D array representing the z-component of velocity field
SF_Node	is a 2D array containing the values of the nodal longitudinal structure functions as function of (l) for a range of orders specified by the user.
SF_p_Node	is a 2D array containing the values of the nodal transverse structure functions for as function of (l) for a range of orders specified by the user.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node and SF_Node_p so as to get the average.
SF_Grid_pll_Node	is a 4D array containing the values of the nodal longitudinal structure functions as function of (lx,ly,lz) for a range of orders specified by the user.
SF_Grid_perp_Node	is a 4D array containing the values of the nodal transverse structure functions for as function of (lx,ly,lz) for a range of orders specified by the user.
counter_Grid_Node	is a 4D array containing the numbers for dividing the values of SF_Grid_pll_Node and SF_Grid_perp_Node_p so as to get the average.

2.1.2.22 void SFunc_long_2D (Array< double, 2 > *Ux,* Array< double, 2 > *Uz,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *counter_Node*)

A less computationally intensive function to calculate only the longitudinal structure functions using 2D velocity field.

The following function computes the only longitudinal structure function using 2D velocity field data. This function is less computationally expensive compared to SFunc3D. The function exploits the fact that $\langle du(l)^p \rangle = \langle du(-l)^p \rangle$, where $du(l)$ is the component parallel to l. Thus, although this function uses four nested loops, the innermost loop starts from z1 instead of 0, where z1 is the iteration number of the second for-loop. The rest of the structure is similar to the function SFunc2D.

Parameters

Ux	is a 3D array representing the x-component of velocity field
Uz	is a 3D array representing the z-component of velocity field
SF_Node	is a 2D array containing the values of the nodal longitudinal structure functions for a range of orders specified by the user.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node so as to get the average.

2.1.2.23 void SFunc_long_2D (Array< double, 2 > *Ux,* Array< double, 2 > *Uz,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *counter_Node,* Array< double, 3 > & *SF_Grid2D_pll_Node,* Array< double, 3 > & *counter_Grid2D_Node*)

A less computationally intensive function to calculate only the longitudinal structure functions as functions of (lx,lz) in addition to function of (l) using 2D velocity field.

The following function computes the only longitudinal structure function using 2D velocity field data. This function is less computationally expensive compared to SFunc2D. The function exploits the fact that $\langle du(l)^p \rangle = \langle du(-l)^p \rangle$, where $du(l)$ is the component parallel to l. Thus, although this function uses six nested loops, the innermost loop starts from z1 instead of 0, where z1 is the iteration number of the third for-loop. The rest of the structure is similar to the function SFunc3D.

Parameters

Ux	is a 2D array representing the x-component of velocity field
Uz	is a 2D array representing the z-component of velocity field
SF_Node	is a 2D array containing the values of the nodal longitudinal structure functions for a range of orders specified by the user.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node so as to get the average.
SF_Grid2D_pll_Node	is a 3D array containing the values of the nodal longitudinal structure functions as function of (lx,lz) for a range of orders specified by the user.
counter_Grid2D_Node	is a 3D array containing the numbers for dividing the values of SF_Grid_pll_Node so as to get the average.

2.1.2.24 void SFunc_long_3D (Array< double, 3 > *Ux,* Array< double, 3 > *Uy,* Array< double, 3 > *Uz,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *counter_Node*)

A less computationally intensive function to calculate only the longitudinal structure functions using 3D velocity field.

The following function computes the only longitudinal structure function using 3D velocity field data. This function is less computationally expensive compared to SFunc3D. The function exploits the fact that $\langle du(l)^p \rangle = \langle du(-l)^p \rangle$, where $du(l)$ is the component parallel to l. Thus, although this function uses six nested loops, the innermost loop starts from z1 instead of 0, where z1 is the iteration number of the third for-loop. The rest of the structure is similar to the function SFunc3D.

Parameters

Ux	is a 3D array representing the x-component of velocity field
Uy	is a 3D array representing the y-component of velocity field
Uz	is a 3D array representing the z-component of velocity field
SF_Node	is a 2D array containing the values of the nodal longitudinal structure functions for a range of orders specified by the user.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node so as to get the average.

2.1.2.25 void SFunc_long_3D (Array< double, 3 > *Ux,* Array< double, 3 > *Uy,* Array< double, 3 > *Uz,* Array< double, 2 > & *SF_Node,* Array< double, 2 > & *counter_Node,* Array< double, 4 > & *SF_Grid_pll_Node,* Array< double, 4 > & *counter_Grid_Node*)

A less computationally intensive function to calculate only the longitudinal structure functions as functions of (lx,ly,lz) in addition to function of (l) using 3D velocity field.

The following function computes the only longitudinal structure function using 3D velocity field data. This function is less computationally expensive compared to SFunc3D. The function exploits the fact that $\langle du(l)^p \rangle = \langle du(-l)^p \rangle$,

where $du(l)$ is the component parallel to l. Thus, although this function uses six nested loops, the innermost loop starts from z1 instead of 0, where z1 is the iteration number of the third for-loop. The rest of the structure is similar to the function SFunc3D.

Parameters

Ux	is a 3D array representing the x-component of velocity field
Uy	is a 3D array representing the y-component of velocity field
Uz	is a 3D array representing the z-component of velocity field
SF_Node	is a 2D array containing the values of the nodal longitudinal structure functions for a range of orders specified by the user.
counter_Node	is a 2D array containing the numbers for dividing the values of SF_Node so as to get the average.
SF_Grid_pll_Node	is a 4D array containing the values of the nodal longitudinal structure functions as function of (lx,ly,lz) for a range of orders specified by the user.
counter_Grid_Node	is a 4D array containing the numbers for dividing the values of SF_Grid_pll_Node and SF_Grid_perp_Node_p so as to get the average.

2.1.2.26 void write_2D (Array< double, 2 > *A*, string *file*)

Function to write the structure functions as function of l as a 2D hdf5 file.

This function reads writes the structure functions as a 2D array of dimensions (N x p). Here, N = $\sqrt{Nx^2+Ny^2+Nz^2}$ is the number of equidistant points from 0 to L, where L is the length of the diagonal of the domain in which the structure functions are calculated. p is the order of the structure functions.

Parameters

A	is the 2D array to store the structure functions.
file	is the name of the hdf5 file and the dataset in which the structure functions are stored.

2.1.2.27 void write_3D (Array< double, 3 > *A*, string *file*, int *q*)

Function to write the structure functions as function of lx,lz as a 2D hdf5 file.

This function reads the structure functions as a 4D array of dimensions (lx x lz x np), where np is the order of the structure function. The structure functions of different orders are then stored as separate 2D hdf5 files.

Parameters

A	is the 3D array representing the structure functions.
file	is the name of the hdf5 file and the dataset in which the structure functions are stored.
q	is the order of the structure function to be stored.

2.1.2.28 void write_4D (Array< double, 4 > *A*, string *file*, int *q*)

Function to write the structure functions as function of lx,ly,lz as a 3D hdf5 file.

This function reads the structure functions as a 4D array of dimensions (lx x ly x lz x np), where np is the order of the structure function. The structure functions of different orders are then stored as separate 3D hdf5 files.

Parameters

A	is the 4D array representing the structure functions.
file	is the name of the hdf5 file and the dataset in which the structure functions are stored.

q	is the order of the structure function to be stored.

2.1.3 Variable Documentation

2.1.3.1 Array<double,4> counter_Grid

4D array storing the counter array for averaging SF_Grid_pll and SF_Grid_perp

2.1.3.2 Array<double,3> counter_Grid2D

3D array storing the counter array for averaging SF_Grid_pll and SF_Grid_perp

2.1.3.3 int field_procedure

This variable decides whether the code needs to generate velocity field data or read velocity field data from HDF5 files.

If the value is 0, the code generates velocity field data. Else, it reads from the HDF5 files in the "in" folder. Entered by the user.

2.1.3.4 bool grid_switch

This variable decides whether the structure functions as function of (lx, ly, lz), or of (lx, lz) in case of 2D, needs to be calculated in addition to the structure functions as function of (l).

If the value is TRUE, then the structure functions as function of (lx, ly, lz) / (lx, lz) will be computed in addtion to the structure functions as function of (l).

2.1.3.5 bool longitudinal

This variable decides whether to calculate both transverse and longitudinal structure functions or only the longitudinal structure functions using a less computationally expensive technique.

If the value is false, the code calculates both transverse and longitudinal structure functions. Else, it calculates only the longitudinal structure functions using less number of iterations.

2.1.3.6 int q1

The first order of the range of orders of the structure functions to be computed.

Entered by the user.

2.1.3.7 int q2

The last order of the range of orders of the structure functions to be computed.

Entered by the user.

2.1.3.8 bool scalar_switch

This variable decides whether the scalar or velocity structure functions are to be evaluated.

If the value is TRUE, then scalar structure functions will be evaluated, else the vector (velocity) structure functions will be evaluated.

2.1.3.9 bool two_dimension_switch

This variable decides whether the structure functions are to be calculated using 2D or 3D velocity field data.

If true, then the code will read 2D velocity fields and calculate the corresponding structure functions. Otherwise, it will read 3D velocity fields. Entered by the user