UPC++ Spectral NS Solver

Generated by Doxygen 1.8.14

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 FFT1D Class Reference

do FFT in 1 dimension

`#include <FFT1D.H>`

Public Member Functions

- FFT1D ()
  default constructor
- void init (int N, int stride, cdouble *input)
  set stride and size to reuse plan
- ~FFT1D ()
  destructor
- void forward (cdouble *input, int N, int stride=1)
  in place forward transform on data of size N with stride stride
- void inverse (cdouble *input, int N, int stride=1)
  in place inverse transform on data of size N with stride stride
- void transform (cdouble *input, int N, int stride, int dir)
  in place transform
- void **transform** (cdouble *input, int DIR)

Private Attributes

- fftw_plan m_forward
  fftw plan for many FFTs of the same size and stride
- fftw_plan **m_backward**
- int m_N
  size of FFT
- int m_stride
  stride
- int m_dir
  FFT direction.
- int **m_alignment**

2.1.1 Detailed Description

do FFT in 1 dimension

2.1.2 Member Function Documentation

2.1.2.1 transform()

```
void FFT1D::transform (
            cdouble * input,
            int N,
            int stride,
            int dir )
```

in place transform

dir = FFTW_FORWARD is forward
dir = FFTW_BACKWARD is inverse

The documentation for this class was generated from the following files:

- /home/sam/school/cs267/spectral_upcxx/src/FFT1D.H
- /home/sam/school/cs267/spectral_upcxx/src/FFT1D.cpp

2.2 Particle Class Reference

Marker particle class.

```
#include <Particle.H>
```

Public Member Functions

- Particle ()
  default constructor
- void **move** (const Vector &velocity, double K)
- double **energy** ()
- array< double, DIM > **position** ()

Private Attributes

- array< double, DIM > **m_loc**
- array< double, DIM > **m_v**

2.2.1 Detailed Description

Marker particle class.

The documentation for this class was generated from the following files:

- /home/sam/school/cs267/spectral_upcxx/src/Particle.H
- /home/sam/school/cs267/spectral_upcxx/src/Particle.cpp

2.3 Scalar Class Reference

Store a 3D array with UPCXX.

`#include <Scalar.H>`

Public Member Functions

- Scalar ()
  default constructor
- Scalar (array< INT, DIM > N, bool physical=true)
  constructor
- ~Scalar ()
  destructor
- Scalar (const Scalar &scalar)
  copy constructor
- void operator= (const Scalar &scalar)
  copy assignment. Does a deep copy
- void init (array< INT, DIM > N, bool physical=true)
  initialize the array after calling default constructor
- void set (array< INT, DIM > index, cdouble val)
  set a value in the distributed array
- cdouble get (array< INT, DIM > index) const
  get a value from the distributed array
- cdouble & operator[ ] (array< INT, DIM > index)
  get a local element from the distributed array
- cdouble operator[ ] (array< INT, DIM > index) const
  const access to distributed array
- cdouble & operator() (INT a_i, INT a_j, INT a_k)
  get a local element from the distributed array (with 3D indexing)
- cdouble operator() (INT a_i, INT a_j, INT a_k) const
  const access to a local element (with 3D indexing)
- cdouble & operator[ ] (INT index)
  direct access to local pointer
- cdouble operator[ ] (INT index) const
  const access to local pointer
- array< double, DIM > freq (array< INT, DIM > ind) const
  return frequency for grid point ind
- void forward ()

in place forward transform

- void inverse ()

  in place reverse transform
- void forward (Scalar &a_scalar) const

  out of place forward transform
- void inverse (Scalar &a_scalar) const

  out of place reverse transform
- Vector gradient () const

  compute gradient ∇f. Returns in fourier space
- Scalar laplacian () const

  compute laplacian: $\nabla^2 f$. Returns in fourier space
- void laplacian_inverse (double a=0, double b=1)

  invert laplacian
- void zeroHighModes ()

  zero out the highest modes
- INT localSize () const

  get size per processor
- array< INT, DIM > getDims () const

  get dimensions
- array< INT, DIM > getPDims () const

  get parallel truncated dimensions
- array< INT, DIM > getPStart () const

  get beginning of local truncated dimensions
- array< INT, DIM > getPEnd () const

  get end of truncated local dimensions
- INT size () const

  get total size
- double memory () const

  compute allocated memory size per processor
- bool isPhysical () const

  *query if (*this) is in physical space*
- bool isFourier () const

  *query if (*this) is in fourier space*
- void setFourier ()

  *flag (*this) as in fourier space*
- void setPhysical ()

  *flag (*this) as in physical space*
- double average () const

  get the average value of the data

Private Member Functions

- void transform (int dir)

  perform fourier transform in parallel
- void transposeX2Z (cdouble *f)

  transpose so that contiguous dimension is in z
- void transposeZ2X (cdouble *f)

  transpose from contiguous in z to contiguous in x
- void getIndex (array< INT, DIM > index, INT &rank, INT &loc) const

  get the rank and location into the distributed array
- cdouble * getLocal () const

  get pointer to local data

Private Attributes

- INT m_Nz

  number of Nx by Ny slabs in the z direction
- INT m_Ny

  number of Nx by Nz slabs in the y direction (for after the global transpose)
- INT m_N

  total number of values $N_x \times N_y \times N_z$
- INT m_dSize

  size of data owned by a processor
- array< INT, DIM > m_dims

  dimensions of array in x, y, z
- array< INT, DIM > m_pdims

  parallel truncated dimensions
- vector< upcxx::global_ptr< cdouble > > m_ptrs

  global pointers to z slabs
- cdouble * m_local

  local version of m_ptrs[rank_me()]
- FFT1D m_fft_x

  fft for striding in x
- FFT1D m_fft_y

  fft for striding in y
- FFT1D m_fft_z

  fft for striding in z
- int m_rank

  store upcxx rank
- bool m_fourier

  store if in physical/fourier space
- bool m_initialized

  store if memory is allocated with init
- vector< upcxx::global_ptr< cdouble > > tmp

  preallocate tmp for transpose

Static Private Attributes

- static int m_nscalars =0

  total scalars created

2.3.1 Detailed Description

Store a 3D array with UPCXX.

stores x,y locally and distributes in z

2.3.2 Constructor & Destructor Documentation

2.3.2.1 Scalar()

```
Scalar::Scalar (
            array< INT, DIM > N,
            bool physical = true )
```

constructor

set size in DIM directions. Defaults to building in physical space

2.3.3 Member Function Documentation

2.3.3.1 laplacian_inverse()

```
void Scalar::laplacian_inverse (
            double a = 0,
            double b = 1 )
```

invert laplacian

inverts $a + b\nabla^2$ where a, b are scalars
Divides by: $a - b(m^2 + n^2 + p^2)$

2.3.3.2 operator[]() [1/2]

```
cdouble & Scalar::operator[] (
            array< INT, DIM > index )
```

get a local element from the distributed array

can only access local values

2.3.3.3 operator[]() [2/2]

```
cdouble Scalar::operator[] (
            array< INT, DIM > index ) const
```

const access to distributed array

can only access local values

The documentation for this class was generated from the following files:

- /home/sam/school/cs267/spectral_upcxx/src/Scalar.H
- /home/sam/school/cs267/spectral_upcxx/src/Scalar.cpp

2.4 Vector Class Reference

Represent a vector with scalars in DIM directions.

`#include <Vector.H>`

Public Member Functions

- Vector ()
  default constructor
- Vector (array< INT, DIM > dims, bool physical=true)
  constructor
- void init (array< INT, DIM > dims, bool physical=true)
  initialize memory
- Scalar & operator[ ] (int a_i)
  index into the components of the vector
- const Scalar & operator[ ] (int a_i) const
  const access to components of vector
- void forward ()
  component wise transform to fourier space
- void inverse ()
  component wise transform to physical space
- void forward (Vector &a_vector) const
  component wise out of place transform to fourier space
- void inverse (Vector &a_vector) const
  component wise out of place transform to physical space
- Vector cross (const Vector &a_v) const
  cross product of two vectors ($(*this) \times a_v$ *)*
- Vector curl () const
  curl $\nabla \times \vec{f}$
- Scalar divergence () const
  Divergence of (∗*this):* $\nabla \cdot \vec{f}$.
- Vector laplacian () const
  vector laplacian: $\nabla \cdot \nabla \vec{f}$
- bool isFourier () const
  query if (∗*this) is in fourier space*
- bool isPhysical () const
  query if (∗*this) is in physical space*
- INT localSize () const
  number of elements local to this processor
- array< INT, DIM > getDims () const
  get full dimensions
- array< INT, DIM > getPStart () const
  get local parallel truncated start
- array< INT, DIM > getPEnd () const
  get local parallel truncated end
- array< INT, DIM > getPDims () const
  get parallel truncated dimensions

Private Member Functions

- void setFourier ()

  flag (∗this) as in fourier space
- void setPhysical ()

  flag (∗this) as in physical space
- array< cdouble ∗, DIM > getLocal () const

  direct access to local data

Private Attributes

- array< INT, DIM > m_dims

  dimensions
- INT m_N

  total size
- array< Scalar, DIM > m_vector

  store the DIM scalars

2.4.1 Detailed Description

Represent a vector with scalars in DIM directions.

2.4.2 Constructor & Destructor Documentation

2.4.2.1 Vector()

```
Vector::Vector (
            array< INT, DIM > dims,
            bool physical = true )
```

constructor

supply Nx, Ny, Nz. Defaults to building in physical space

2.4.3 Member Function Documentation

2.4.3.1 cross()

```
Vector Vector::cross (
            const Vector & a_v ) const
```

cross product of two vectors ($(*this) \times a_v$)

does cross product in physical space. returns in fourier space

The documentation for this class was generated from the following files:

- /home/sam/school/cs267/spectral_upcxx/src/Vector.H
- /home/sam/school/cs267/spectral_upcxx/src/Vector.cpp

2.5 Writer Class Reference

Parallel VTK IO class.

`#include <Writer.H>`

Public Member Functions

- Writer (string name="solution")
  constructor. provide base name of output VTK file
- ∼Writer ()
  destructor. cleans up fstream
- void add (Scalar &a_scalar, string a_name)
  add a Scalar to the output list
- void add (Vector &a_vector, string a_name)
  add a Vector to the output list
- void write ()
  write all variables to VTK
- void setFreq (int a_f)
  set write frequency to not output at every time step

Private Attributes

- string m_name
  store base name
- int m_count
  number of times write has been called
- int m_writes
  number of files that have been written to file
- vector< Scalar * > m_scalars
  store pointers to scalar variables
- vector< string > m_scalar_names
  names of scalar variables
- vector< Vector * > m_vectors
  pointers to vector variables
- vector< string > m_vector_names
  names of vector variables
- int m_f
  output frequency
- ofstream **m_out**

2.5.1 Detailed Description

Parallel VTK IO class.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 Writer()

```
Writer::Writer (
            string name = "solution" )
```

constructor. provide base name of output VTK file

actual name will be name+upcxx::rank_me()_step#.vtk

2.5.3 Member Function Documentation

2.5.3.1 add()

```
void Writer::add (
            Scalar & a_scalar,
            string a_name )
```

add a Scalar to the output list

stores pointer to memory so output updates with changes

2.5.3.2 setFreq()

```
void Writer::setFreq (
            int a_f )
```

set write frequency to not output at every time step

write every f calls to write

The documentation for this class was generated from the following files:

- /home/sam/school/cs267/spectral_upcxx/src/Writer.H
- /home/sam/school/cs267/spectral_upcxx/src/Writer.cpp

Index