APC_524

Generated by Doxygen 1.8.12

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 BC_P_Collection Class Reference

Public Member Functions

- **executeParticleBoundaries** ()

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

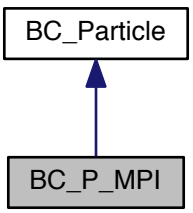
- src/boundaries/boundary_particles.hpp

3.2 BC_P_MPI Class Reference

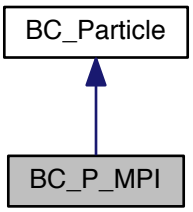
Class representing an MPI transfer condition for particles.

`#include <bc_p_MPI.hpp>`

Inheritance diagram for BC_P_MPI:



Collaboration diagram for BC_P_MPI:



Public Member Functions

- **BC_P_MPI** (Particle_List ∗pl, double xMin, double xMax, int targetRank, double lengthShift)
- void **computeParticleBCs** ()
- void **completeBC** ()

3.2.1 Detailed Description

Class representing an MPI transfer condition for particles.

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

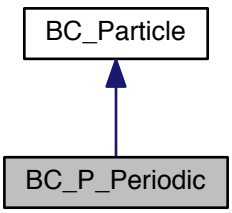
- src/boundaries/b_particles/bc_p_MPI.hpp
- src/boundaries/b_particles/bc_p_MPI.cpp

3.3 BC_P_Periodic Class Reference

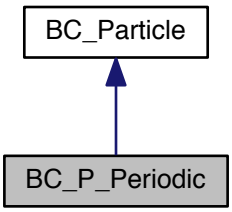
Class representing a periodic boundary condition for particles.

`#include <bc_p_periodic.hpp>`

Inheritance diagram for BC_P_Periodic:



Collaboration diagram for BC_P_Periodic:



Public Member Functions

- **BC_P_Periodic** (Particle_List ∗pl, double xMin, double xMax, int dim_Index)
- void **computeParticleBCs** ()
- void **completeBC** ()

3.3.1 Detailed Description

Class representing a periodic boundary condition for particles.

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

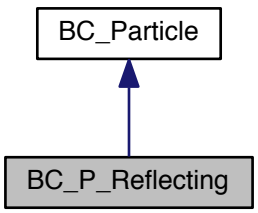
- src/boundaries/b_particles/bc_p_periodic.hpp
- src/boundaries/b_particles/bc_p_periodic.cpp

3.4 BC_P_Reflecting Class Reference

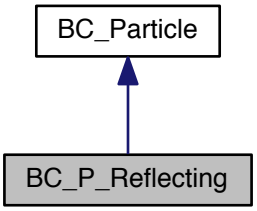
Class representing a reflecting boundary condition for particles.

`#include <bc_p_reflecting.hpp>`

Inheritance diagram for BC_P_Reflecting:



Collaboration diagram for BC_P_Reflecting:



Public Member Functions

- **BC_P_Reflecting** (Particle_List ∗pl, double xMin, double xMax, int dim_Index)
- void **computeParticleBCs** ()
- void **completeBC** ()

3.4.1 Detailed Description

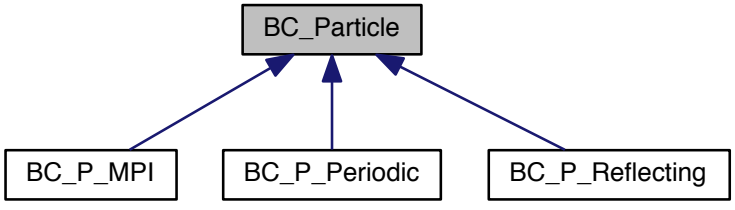
Class representing a reflecting boundary condition for particles.

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

- src/boundaries/b_particles/bc_p_reflecting.hpp
- src/boundaries/b_particles/bc_p_reflecting.cpp

3.5 BC_Particle Class Reference

Inheritance diagram for BC_Particle:



Public Member Functions

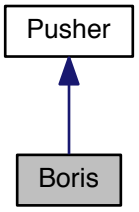
- virtual void **computeParticleBCs** ()=0
- virtual void **completeBC** ()=0

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

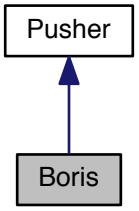
- src/boundaries/boundary_particles.hpp

3.6 Boris Class Reference

Inheritance diagram for Boris:



Collaboration diagram for Boris:



Public Member Functions

- int **Step** (Particle ∗part, Field_part ∗field, double dt)

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

- src/pusher/boris.hpp
- src/pusher/boris.cpp

3.7 Domain Class Reference

Public Member Functions

- **Domain** (int size, int rank, Input_Info_t *input_info)
- int **getnGhosts** (void)
- int * **getnxyz** (void)
- double * **getxyz0** (void)
- double * **getLxyz** (void)
- double getmindx (void)

  Find minimum grid size.

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

- src/domain/domain.hpp
- src/domain/domain.cpp

3.8 Field_part Struct Reference

Public Attributes

- double **e1**
- double **e2**
- double **e3**
- double **b1**
- double **b2**
- double **b3**

The documentation for this struct was generated from the following file:

- src/particles/particle.hpp

3.9 Grid Class Reference

Class representing grid on which E and B fields and currents are defined.

`#include <grid.hpp>`

Public Member Functions

- Grid (int *nxyz, int nGhosts, double *xyz0, double *Lxyz)

  Grid constructor.
- virtual ~Grid ()

  Grid destructor.
- int evolveFields (double dt)

  Evolve Electric and Magnetic fields in time.
- void InitializeFields (int restart)

  Initialize E and B fields.
- void zeroJ ()

  sets all of J (Jx,Jy,Jz) to be identically zero
- int addJ (int cellID, double *Jvec)

  Add currents from particle to grid.
- int getFieldInterpolatorVec (int cellID, double *InterpolatorVec)

  Return vector for field interpolation.
- int getCellID (double x, double y, double z)

  Get cell ID based on particle position.
- int getCellVertex (int cellID, double *xyz)

  Returns vertex corresponding to cell ID.
- int getNumberOfCells ()

  Get total number of cells in grid.
- double getStepSize (int dimension)

  Get step size along dimension in grid.
- void updateGhostCells ()

  updates ghost cells after evolving the field on physical points
- int getGhostVecSize ()

  returns size of ghost cell data to send
- void getGhostVec (const int side, double *ghostVec)

  bundles the data in the ghost cells to send
- void getGhostVecAlt (const int side, double *ghostVec)

  bundles the data in the ghost cells to send
- void setGhostVec (const int side, const double *ghostVec)

  unbundles the data sent from the ghost cells and puts it in the field
- void setGhostVecAlt (const int side, const double *ghostVec)

  unbundles the data sent from the ghost cells and puts it in the field

Protected Member Functions

- double *** newField_ ()

  allocates contiguous block of memory for a single field
- void deleteField_ (double ***fieldPt)

  frees contiguous block of memory for a single field
- int sideToIndex_ (const int side)

  function to convert -/+ 1 left/right side indicator to index in x direction
- void checkInput_ ()

  checks validity of input parameters for Grid constructor
- void sliceMatToVec_ (double ***const mat, const int side)

  slices a physical plane in the x direction (excludes ghosts)
- void unsliceMatToVec_ (double ***mat, const int side)

  unslices a physical plane in the x direction (excludes ghosts)

Protected Attributes

- const int **nx_**
- const int **ny_**
- const int **nz_**
- const int **nGhosts_**
- const double **x0_**
- const double **y0_**
- const double **z0_**
- const double **Lx_**
- const double **Ly_**
- const double **Lz_**
- const int **iBeg_**
- const int **jBeg_**
- const int **kBeg_**
- const int **iEnd_**
- const int **jEnd_**
- const int **kEnd_**
- const double **dx_**
- const double **dy_**
- const double **dz_**
- const double **idx_**
- const double **idy_**
- const double **idz_**
- const int **nRealPtsYZPlane_**
- const int **nFields_**
- const int **ghostVecSize_**
- double *** **Ex_**
- double *** **Ey_**
- double *** **Ez_**
- double *** **Bx_**
- double *** **By_**
- double *** **Bz_**
- double *** **Bx_tm1_**
- double *** **By_tm1_**
- double *** **Bz_tm1_**
- double *** **Jx_**
- double *** **Jy_**
- double *** **Jz_**
- double *** **rhox_**
- double *** **rhoy_**
- double *** **rhoz_**
- double * **sliceTmp_**

3.9.1 Detailed Description

Class representing grid on which E and B fields and currents are defined.

Grid has ghost cells on each face. The ghost cell updating in y and z arises from periodic boundary conditions. x-direction ghost cells allow communication between MPI domains.

Following Yee (1966), electric fields and currents reside on edges, and magnetic fields on faces. Fields are updated using a set of finite-difference equations approximating Ampere's and Faraday's Laws.

A set of getters are available to allow particles to interpolate electric fields based on their position.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 Grid()

```
Grid::Grid (
            int * nxyz,
            int nGhosts,
            double * xyz0,
            double * Lxyz )
```

Grid constructor.

Input arguments:
nxyz: integer array [nx,ny,nz] where nx is the total number of cells (physical + ghost) in the x direction in the simulation, and the same for ny,nz.
nGhosts: integer number of ghost cells on each side of the domain. This should always be at least 1. Currently the code does not support nGhosts>1, though it may in the future (to take advantage of higher order finite difference and interpolation methods, for instance).
xyz0: integer array [x0,y0,z0] where x0 is the initial x position, and the same for y0,z0
Lxyz0: double array [Lx,Ly,Lz] where Lx is the physical length of each cell in the x direction, and the same for Ly,Lz

3.9.2.2 ~Grid()

```
Grid::~Grid ( )  [virtual]
```

Grid destructor.

calls deleteField_ on each of the double*** fields

3.9.3 Member Function Documentation

3.9.3.1 addJ()

```
int Grid::addJ (
            int cellID,
            double * Jvec )
```

Add currents from particle to grid.

Currents added to cell with ID cellID via input vector of form:
[Jx((0,0,0) -> (1,0,0)), Jx((0,1,0) -> (1,1,0)), Jx((0,1,1) -> (1,1,1)), Jx((0,0,1) -> (1,0,1)),...
Jy((0,0,0) -> (0,1,0)), Jy((0,0,1) -> (0,1,1)), Jy((1,0,1) -> (1,1,1)), Jy((1,0,0) -> (1,1,0)),...
Jz((0,0,0) -> (0,0,1)), Jz((1,0,0) -> (1,0,1)), Jz((1,1,0) -> (1,1,1)), Jz((0,1,0) -> (0,1,1))]

3.9.3.2 checkInput_()

```
void Grid::checkInput_ ( )  [protected]
```

checks validity of input parameters for Grid constructor

asserts necessary conditions on each input (mainly positivity of many parameters). Terminates program if inputs are incorrect.

3.9.3.3 deleteField_()

```
void Grid::deleteField_ (
            double *** fieldPt )  [protected]
```

frees contiguous block of memory for a single field

Deletes double*** of size [nx_+1][ny_+1][nz_+1]

3.9.3.4 evolveFields()

```
int Grid::evolveFields (
            double dt )
```

Evolve Electric and Magnetic fields in time.

Uses Yee algorithm to advance E and B fields. Assumes Gaussian-style Maxwell equation, with c = 1.

3.9.3.5 getCellID()

```
int Grid::getCellID (
            double x,
            double y,
            double z )
```

Get cell ID based on particle position.

Cell ID is uniquely given by (ny_*nz_)*ix + nz_*iy + iz.
If particle is in a ghost cell or off the grid entirely, returns
-1 if off (-z), -2 if off (+z)
-3 if off (-y), -4 if off (+y)
-5 if off (-x), -6 if off (+x)

3.9.3.6 getFieldInterpolatorVec()

```
int Grid::getFieldInterpolatorVec (
            int cellID,
            double * InterpolatorVec )
```

Return vector for field interpolation.

Based on cellID, return relevant edge E and face B fields and cell origin, in format:
[x, y, z, ...
Ex(ix, iy, iz), Ex(ix, iy+1,iz), Ex(ix, iy+1, iz+1), Ex(ix, iy, iz+1), ...
Ey(ix, iy, iz), Ey(ix, iy, iz+1), Ey(ix+1, iy, iz+1), Ey(ix+1, iy, iz), ...
Ez(ix, iy, iz), Ez(ix+1, iy, iz), Ez(ix+1, iy+1, iz), Ez(ix, iy+1, iz), ...
Bx(ix, iy, iz), Bx(ix+1, iy, iz), ...
By(ix, iy, iz), By(ix, iy+1, iz), ...
Bz(ix, iy, iz), Bz(ix, iy, iz+1), ...]
where ix, iy, and iz are the row indices for each of the three dimensions (calculated from the cellID)

3.9.3.7 getGhostVec()

```
void Grid::getGhostVec (
            const int side,
            double * ghostVec )
```

bundles the data in the ghost cells to send

stores the data of the E,B,J fields at all of the ghost points along the domain interfaces (yz plane) into a 1D array of doubles to be sent with a single MPI call. ghostVec is an array of length ghostVecSize_ to store the data in. Side is -1 for left side of domain, +1 for right side. Sends (in order): Ex,Ey,Ez,Bx,By,Bz,Jx,Jy,Jz. This is a more elegant implementation than the one in getGhostVec, but may increase cache misses? Requires profiling.

3.9.3.8 getGhostVecAlt()

```
void Grid::getGhostVecAlt (
            const int side,
            double * ghostVec )
```

bundles the data in the ghost cells to send

stores the data of the E,B,J fields at all of the ghost points along the domain interfaces (yz plane) into a 1D array of doubles to be sent with a single MPI call. ghostVec is an array of length ghostVecSize_ to store the data in. Side is -1 for left side of domain, +1 for right side. Sends (in order): Ex,Ey,Ez,Bx,By,Bz,Jx,Jy,Jz. This is an alternative implementation to the one in getGhostVecAlt which is less elegant but might decrease cache misses? Requires profiling

3.9.3.9 getGhostVecSize()

```
int Grid::getGhostVecSize ( )
```

returns size of ghost cell data to send

this size is stored in the protected int ghostVecSize_

3.9.3.10 getNumberOfCells()

```
int Grid::getNumberOfCells ( )
```

Get total number of cells in grid.

Includes ghost cells.

3.9.3.11 getStepSize()

```
double Grid::getStepSize (
            int dimension )
```

Get step size along dimension in grid.

Returns step size along dimension according to; dimension = 0: x dimension = 1: y dimension = 2: z Returns -1 if invalid dimension.

3.9.3.12 InitializeFields()

```
void Grid::InitializeFields (
            int restart )
```

Initialize E and B fields.

Use restart file to set values of initial E,B,J fields

3.9.3.13 newField_()

```
double *** Grid::newField_ ( )  [protected]
```

allocates contiguous block of memory for a single field

Returns double*** of size [nx_+1][ny_+1][nz_+1]

3.9.3.14 setGhostVec()

```
void Grid::setGhostVec (
            const int side,
            const double * ghostVec )
```

unbundles the data sent from the ghost cells and puts it in the field

to be used in conjuction with getGhostVec or getGhostVecAlt. ghostVec is a 1D array storing each of the E,B,J field values at each of the ghost points along the domain interfaces (yz plane) of a single side. Side specifies which side this data should be set to. -1 corresponds to the left side of the domain (receiving data from the right ghost cells of the previous domain) and +1 to the right side (receiving data from the left ghost cells of the next domain). This is an alternate implementation of setGhostVecAlt. setGhostVec is more elegant but may increase cache misses (requires profiling).

3.9.3.15 setGhostVecAlt()

```
void Grid::setGhostVecAlt (
            const int side,
            const double * ghostVec )
```

unbundles the data sent from the ghost cells and puts it in the field

to be used in conjuction with getGhostVec or getGhostVecAlt. ghostVec is a 1D array storing each of the E,B,J field values at each of the ghost points along the domain interfaces (yz plane) of a single side. Side specifies which side this data should be set to. -1 corresponds to the left side of the domain (receiving data from the right ghost cells of the previous domain) and +1 to the right side (receiving data from the left ghost cells of the next domain). This is an alternate implementation of setGhostVec. setGhostVecAlt is less elegant but may reduce cache misses (requires profiling).

3.9.3.16 sideToIndex_()

```
int Grid::sideToIndex_ (
            const int side )  [protected]
```

function to convert -/+ 1 left/right side indicator to index in x direction

For use with ghost cell methods. side=-1 indicates operations on the left side of the domain, side=+1 indicates operations on the right side of the domain. This method converts side into the correct index i to reference ghost cells on that side of the domain. For instance, called by getGhostVec and setGhostVec. Generalizes to any number of ghost cells so long as iBeg_ and iEnd_ are initialized correctly.

3.9.3.17 sliceMatToVec_()

```
void Grid::sliceMatToVec_ (
            double ***const mat,
            const int side )  [protected]
```

slices a physical plane in the x direction (excludes ghosts)

mat is 3D array whose real (non-ghost) data on one side will be stored in sliceTmp_ as a 1D array. side is an integer +1 to indicate storage of the right hand side values and -1 to indicate storage of the left hand side. Complementary function to unsliceMatToVec_.

3.9.3.18 unsliceMatToVec_()

```
void Grid::unsliceMatToVec_ (
            double *** mat,
            const int side )  [protected]
```

unslices a physical plane in the x direction (excludes ghosts)

mat is 3D array whose real (non-ghost) data on one side will be set from the temporary 1D array sliceTmp_. side is an integer +1 to indicate setting of the right hand side values and -1 to indicate setting of the left hand side. Complementary function to sliceMatToVec_.

3.9.3.19 updateGhostCells()

```
void Grid::updateGhostCells ( )
```

updates ghost cells after evolving the field on physical points

For each of Ei_,Bi_,Ji_ (for i=x,y,z), copies the value of the field on the outermost physical grid points onto their adjacent ghost grid points. Currently this method requires nGhosts_=1 and will not perform correctly if nGhosts_ != 1 (it may not crash but will not update the ghost cells as desired).

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

- src/grid/grid.hpp
- src/grid/grid.cpp
- src/grid/oGrid.cpp
- src/grid/spookyGrid.cpp

3.10 Input_Info_t Struct Reference

Structure storing info in the input file.

`#include <IO.hpp>`

Public Attributes

- int **nx**
- int **nt**
- int **restart**
- long **np**
- double **t0**
- double **dens**
- double **temp**
- char **distname** [50]

3.10.1 Detailed Description

Structure storing info in the input file.

The documentation for this struct was generated from the following file:

- src/IO/IO.hpp

3.11 Interpolator Class Reference

Public Member Functions

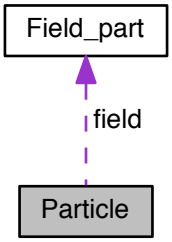
- void **interpolate_fields** (double ∗pos, double ∗lcell, double ∗cellvars, Field_part ∗field)

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

- src/particles/interpolate.hpp
- src/particles/interpolate.cpp

3.12 Particle Struct Reference

Collaboration diagram for Particle:



Public Attributes

- double **x** [3]
- double **v** [3]
- double **xo** [3]
- double **vo** [3]
- double **q**
- double **m**
- int **my_id**
- short **isGhost**
- Field_part **field**

The documentation for this struct was generated from the following file:

- src/particles/particle.hpp

3.13 Particle_Compare Class Reference

Public Member Functions

- **Particle_Compare** (Grid *grid)
- bool **operator()** (Particle const a, Particle const b) const

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

- src/particles/particle_utils.hpp

3.14 Particle_Handler Class Reference

Class that handles all particle-relevant operations.

`#include <particle_handler.hpp>`

Public Member Functions

- **Particle_Handler** (long np)
- void **Load** (int restart)
- void **Push** (double dt)
- void **Pass** ()
- long **nParticles** ()
- void **SortParticles** (Particle_Compare comp)
- void **setPusher** (Pusher *pusher)
- void **InterpolateEB** (Grid *grid)
- void **depositCurrent** (Grid *grids)
- void **depositCharge** (Grid *grids)
- std::vector< Particle > **getParticleVector** ()
- double **maxVelocity** (void)

3.14.1 Detailed Description

Class that handles all particle-relevant operations.

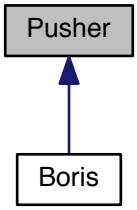
Particle handler handles all the particle operations. This includes deposition, boundary conditions, particle pushing, and communication between MPI nodes if needed

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

- src/particles/particle_handler.hpp
- src/particles/particle_handler.cpp

3.15 Pusher Class Reference

Inheritance diagram for Pusher:



Public Member Functions

- virtual int **Step** (Particle ∗part, Field_part ∗field, double dt)=0

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

- src/pusher/pusher.hpp