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_Factory Class Reference

`#include <bc_factory.hpp>`

Public Types

- typedef BC_Particle ∗(∗ **Factory**) (Domain ∗domain, int dim_Index, short isRight, std::string type)

Public Member Functions

- BC_Particle ∗∗ **constructConditions** (Domain ∗domain, const char(∗bound)[32])
- void **declare** (const std::string &type, Factory factory)
- Factory **lookup** (const std::string &type)
- std::vector< const std::string ∗ > **types** () const

Static Public Member Functions

- static BC_Factory & **getInstance** ()

3.1.1 Detailed Description

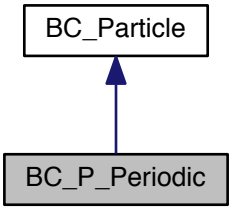
A singleton class to handle registration of particle boundaries

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

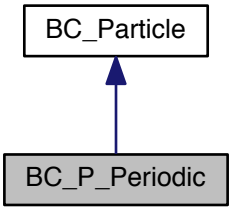
- src/boundaries/bc_factory.hpp
- src/boundaries/bc_factory.cpp

3.2 BC_P_Periodic Class Reference

Inheritance diagram for BC_P_Periodic:



Collaboration diagram for BC_P_Periodic:



Public Member Functions

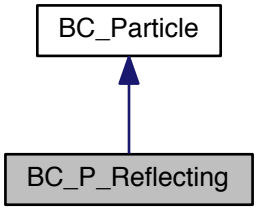
- **BC_P_Periodic** (Domain *domain, int dim_Index, short isRight, std::string type)
- void **computeParticleBCs** (std::vector< Particle > *pl)
- int **completeBC** (std::vector< Particle > *pl)

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

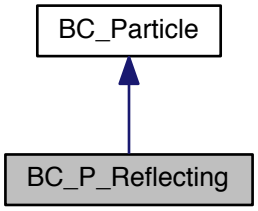
- src/boundaries/b_particles/bc_p_periodic.cpp

3.3 BC_P_Reflecting Class Reference

Inheritance diagram for BC_P_Reflecting:



Collaboration diagram for BC_P_Reflecting:



Public Member Functions

- **BC_P_Reflecting** (Domain *domain, int dim_Index, short isRight, std::string type)
- void **computeParticleBCs** (std::vector< Particle > *pl)
- int **completeBC** (std::vector< Particle > *pl)

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

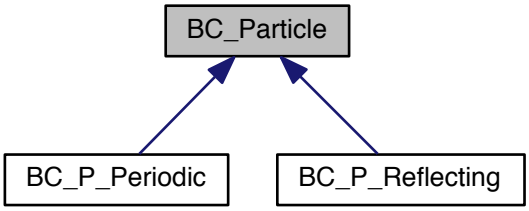
- src/boundaries/b_particles/bc_p_reflecting.cpp

3.4 BC_Particle Class Reference

Class which defines a particle boundary condition.

`#include <boundary_particles.hpp>`

Inheritance diagram for BC_Particle:



Public Member Functions

- int **computeParticleBCs** (std::vector< Particle > *pl)

3.4.1 Detailed Description

Class which defines a particle boundary condition.

Boundary conditions have two stages.

1st stage: Cycling through particle list and determining which particles need to have boundary conditions applied, then applies them.

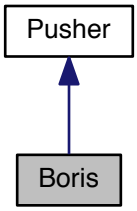
2nd stage: Perform any more auxilliary computations, including MPI calls, creating new ghost particles, shuffling particles ETC...

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

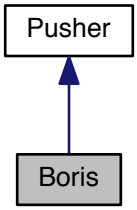
- src/boundaries/boundary_particles.hpp
- src/boundaries/boundary_particles.cpp

3.5 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
- src/pusher/relativisticBoris.cpp

3.6 Depositor Class Reference

Public Member Functions

- void **deposit_particle_J** (Particle ∗part, double ∗lcell, double ∗cellverts, double ∗JObj)
- void **deposit_particle_Rho** (Particle ∗part, double ∗lcell, double ∗cellverts, double ∗RhoObj)

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

- src/particles/deposit.hpp
- src/particles/deposit.cpp

3.7 Domain Class Reference

Public Member Functions

- **Domain** (Input_Info_t *input_info)
- int **getnGhosts** (void)
- int * **getnxyz** (void)
- int * **getn2xyz** (void)
- double * **getxyz0** (void)
- double * **getLxyz** (void)
- double getmindx (void)

  Find minimum grid size.
- void mallocGhosts (int xgsize, int ygsize, int zgsize)

  Allocate ghost buffers for MPI.
- void **freeGhosts** (void)
- void PassFields (Grid *grids, Input_Info_t *input_info, int sendID)

  Pass fields across MPI boundaries, or execute physical boundary conditions.
- int * **getnProcxyz** (void)
- int * **getmyijk** (void)
- int * **getNeighbours** ()
- int **getxl** (void)
- int **getyl** (void)
- int **getzl** (void)
- int **getxr** (void)
- int **getyr** (void)
- int **getzr** (void)
- int ijkToRank (int i, int j, int k)

  return rank for assigned i,j,k
- void RankToijk (int rank, int *myijk)

  assign value to allocated myijk[3]

3.7.1 Member Function Documentation

3.7.1.1 mallocGhosts()

```
void Domain::mallocGhosts (
            int xgsize,
            int ygsize,
            int zgsize )
```

Allocate ghost buffers for MPI.

xgsize : size of ghost buffer in x direction ygsize : size of ghost buffer in y direction zgsize : size of ghost buffer in z direction

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

- src/domain/domain.hpp
- src/domain/domain.cpp
- src/domain/ghosts.cpp
- src/domain/pass_fields.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 FieldBC Class Reference

Class for supplying boundary conditions to field grid.

`#include <fieldBC.hpp>`

Public Member Functions

- **FieldBC** (std::string &fieldStr, int dim, bool edge, double amp, double omega, double phase)
- void applyBCs (double t, Grid &grid)

  Apply boundary condition to grid.

3.9.1 Detailed Description

Class for supplying boundary conditions to field grid.

Boundary conditions are of form:
amp ∗ cos(omega ∗ t + phase)
along plane perpendicular to dimension dim (0 = x, 1 = y, 2 = z) on edge (false = low, true = high)
fieldStr one of Ex, Ey, Ez, Bx, By, Bz

3.9.2 Member Function Documentation

3.9.2.1 applyBCs()

```
void FieldBC::applyBCs (
            double t,
            Grid & grid )
```

Apply boundary condition to grid.

Uses setFieldAlongEdge method in grid to add field to grid.

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

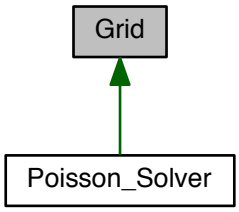
- src/grid/fieldBC.hpp
- src/grid/fieldBC.cpp

3.10 Grid Class Reference

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

`#include <grid.hpp>`

Inheritance diagram for Grid:



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.
- int evolveFieldsES (double dt)
  Evolve Electric Fields Electrostatically.
- void InitializeFields (int restart)
  Initialize E and B fields.
- void zeroJ ()
  sets all of J (Jx,Jy,Jz) to be identically zero
- void zeroRho ()
  sets all of rho to be identically zero
- void zeroE ()
  sets all of E to be identically zero
- void zeroB ()
  sets all of B to be identically zero
- int addJ (int cellID, double ∗Jvec)
  Add currents from particle to grid.
- int addRho (int cellID, double ∗Rhovec)
  Add charge 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.

- int getNumCells3D (double *nvec)

  Get # of cells in each dimension of grid.

- double getStepSize (int dimension)

  Get step size along dimension in grid.

- int setFieldAlongEdge (std::string &fieldStr, int dim, bool edge, double fieldVal)

  Set field along a certain edge.

- int getGhostVecSize ()

  returns size of ghost cell data to send

- void getGhostVec (const int side, double *ghostVec, int sendID)

  bundles the data in the ghost cells to send

- void setGhostVec (const int side, double *ghostVec, int sendID)

  unbundles the data in the ghost cells that have been received

- void updatePeriodicGhostCells ()

  updates J,E,B ghost cells in y/z directions with periodic boundary conditions

- void **setBoundaryVec** (const int side, const double *ghostVec)

Protected Member Functions

- double *** newField_ (int ifield)

  allocates memory for a single field

- void deleteField_ (double ***fieldPt, int ifield)

  frees memory for a single field

- int ** **setFieldSize_** ()
- void **deleteFieldSize_** ()
- int * **setFieldType_** ()
- void **deleteFieldType_** ()
- double **** **setFieldPtr_** ()
- void **deleteFieldPtr_** ()
- int sideToIndex_ (const int side, const int fieldID)

  function to convert -/+ 1 left/right side indicator to index in x direction (description out of date)

- void checkInput_ ()

  checks validity of input parameters for Grid constructor

- void sliceMatToVec_ (const int fieldID, const int side, const int offset, double *vec)

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

- void unsliceMatToVec_ (const int fieldID, const int side, const int offset, double *vec)

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

- int setFieldInPlane_ (int dim, int indx, double ***field, double fieldVal)

  Internal method to set field along a plane.

- **FRIEND_TEST** (oGridInternalTest, EMWave)
- **FRIEND_TEST** (oGridInternalTest, EMWaveLong)

Protected Attributes

- const int **nx_**
- const int **ny_**
- const int **nz_**
- const int **nGhosts_**
- const int **nxTot_**
- const int **nyTot_**
- const int **nzTot_**
- 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 double **dx_**
- const double **dy_**
- const double **dz_**
- const double **idx_**
- const double **idy_**
- const double **idz_**
- const int **maxPointsInPlane_**
- const int **nFieldsToSend_**
- const int **ghostVecSize_**
- const int **nFieldsTotal_**
- const int **ExID_**
- const int **EyID_**
- const int **EzID_**
- const int **BxID_**
- const int **ByID_**
- const int **BzID_**
- const int **JxID_**
- const int **JyID_**
- const int **JzID_**
- const int **Bx_tm1ID_**
- const int **By_tm1ID_**
- const int **Bz_tm1ID_**
- const int **rhoID_**
- const int **nTypes_**
- const int **edgeXID_**
- const int **edgeYID_**
- const int **edgeZID_**
- const int **faceXID_**
- const int **faceYID_**
- const int **faceZID_**
- const int **vertID_**
- 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 *** **rho_**
- int * **fieldType_**
- int ** **fieldSize_**
- double **** **fieldPtr_**
- double * **fieldIsContiguous_**
- double * **sliceTmp_**
- double * **ghostTmp_**

Friends

- class **oGridInternalTest**

3.10.1 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.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 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

3.10.2.2 ~Grid()

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

Grid destructor.

calls deleteField_ on each of the double*** fields

3.10.3 Member Function Documentation

3.10.3.1 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))]

3.10.3.2 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.

3.10.3.3 deleteField_()

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

frees memory for a single field

Uses fieldIsContiguous_ to determine contiguous or noncontiguous deltion method

3.10.3.4 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.

3.10.3.5 evolveFieldsES()

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

Evolve Electric Fields Electrostatically.

Ignores "light wave" contribution (curl terms), effectively only solves poisson equation.

3.10.3.6 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)

3.10.3.7 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)

3.10.3.8 getGhostVec()

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

bundles the data in the ghost cells to send

side = -/+ 1 for left/right x direction, -/+ 2 for y, -/+ 3 for z
ghostVec is the vector to store the data in, which must be of length ghostVecSize_ (can be determined with getGhostVecSize())
Stores the data of the E,B,J fields along the specified boundary plane into a 1D array to be sent with a single MPI call. Stores in order: Ex,Ey,Ez,Bx,By,Bz,Jx,Jy,Jz.
ghostVec can be unpacked with setGhostVec function

3.10.3.9 getGhostVecSize()

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

returns size of ghost cell data to send

this size is stored in the protected int ghostVecSize_

3.10.3.10 getNumberOfCells()

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

Get total number of cells in grid.

Includes ghost cells.

3.10.3.11 getNumCells3D()

```
int Grid::getNumCells3D (
            double * nvec )
```

Get # of cells in each dimension of grid.

Includes ghost cells.

3.10.3.12 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.

3.10.3.13 InitializeFields()

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

Initialize E and B fields.

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

3.10.3.14 newField_()

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

allocates memory for a single field

Returns double*** of size [nx_+1][ny_+1][nz_+1].
First attempts to allocate contiguously. If that fails, issues a warning and attempts to allocate with several calls to new.

3.10.3.15 setFieldAlongEdge()

```
int Grid::setFieldAlongEdge (
            std::string & fieldStr,
            int dim,
            bool edge,
            double fieldVal )
```

Set field along a certain edge.

Inputs:
fieldStr: string of format "Ex", "Bz", etc
dim: dimension along which to apply boundary condition
edge: side along which to apply boundary condition

3.10.3.16 setFieldInPlane_()

```
int Grid::setFieldInPlane_ (
            int dim,
            int indx,
            double *** field,
            double fieldVal )  [protected]
```

Internal method to set field along a plane.

Inputs:
dimension perpendicular to plane.
For example, if dim=0 (x direction), then this program set field in one yz plane.

indx along dimenstion perpendicular to plane.
For example, if dim=0 and indx =14, then set field for the 14th yz plane.

field to set along dimension
value to set field

3.10.3.17 setGhostVec()

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

unbundles the data in the ghost cells that have been received

side = -/+ 1 for left/right x direction, -/+ 2 for y, -/+ 3 for z
ghostVec is the vector to read the data from, which must be of length ghostVecSize_ (can be determined with getGhostVecSize())
Sets the data of the E,B,J fields along the specified boundary plane from the 1D array received from a single MPI call. Sets in order: Ex,Ey,Ez,Bx,By,Bz,Jx,Jy,Jz.
ghostVec can be generated with getGhostVec function

3.10.3.18 sideToIndex_()

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

function to convert -/+ 1 left/right side indicator to index in x direction (description out of date)

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.

3.10.3.19 sliceMatToVec_()

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

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

mat is 3D array whose real (non-ghost) data on one side will be stored in vec as a 1D array. vec must be of size maxPointsInPlane_. side is an integer -/+ 1 to indicate the location on the left/right side in the x direction, -/+ 2 in y, -/+ 3 in z. offset is an integer offset from the first/last physical index determined by side (e.g. side=-1 and offset=0 gives the yz plane of the 1st physical grid points in x direction, whereas offset=-1 would have returned the adjacent ghost cells and offset = 3 would have returned the 4th physical yz plane from the left). unsliceMatToVec_ is the inverse function.

3.10.3.20 unsliceMatToVec_()

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

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

mat is 3D array whose real (non-ghost) data on one side will be replaced by data in the 1D array vec. vec must be of size maxPointsInPlane_. side is an integer -/+ 1 to indicate the location on the left/right side in the x direction, -/+ 2 in y, -/+ 3 in z. offset is an integer offset from the first/last physical index determined by side (e.g. side=-1 and offset=0 gives the yz plane of the 1st physical grid points in x direction, whereas offset=-1 would have returned the adjacent ghost cells and offset = 3 would have returned the 4th physical yz plane from the left). sliceMatToVec_ is the inverse function.

3.10.3.21 zeroJ()

```
void Grid::zeroJ ( )
```

sets all of J (Jx,Jy,Jz) to be identically zero

Used during particle deposition.

3.10.3.22 zeroRho()

```
void Grid::zeroRho ( )
```

sets all of rho to be identically zero

Used during particle deposition.

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

3.11 Input_Info_t Struct Reference

Structure storing info in the input file.

```
#include <IO.hpp>
```

Public Attributes

- int **nCell** [3]
- int **nProc** [3]
- int **nt**
- int **restart**
- int **debug**
- long **np**
- double **t0**
- double **dens**
- double **temp**
- double **massratio**
- double **xyz0** [3]
- double **Lxyz** [3]
- char **distname** [50]
- char **parts_bound** [6][32]
- char **fields_bound** [6][32]

3.11.1 Detailed Description

Structure storing info in the input file.

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

- src/IO/IO.hpp

3.12 Interpolator Class Reference

Public Member Functions

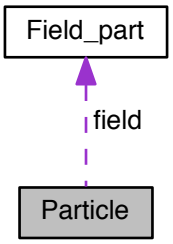
- void **interpolate_fields** (double ∗pos, double ∗lcell, double ∗cellvars, Field_part ∗field)

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

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

3.13 Particle Struct Reference

Collaboration diagram for Particle:



Public Attributes

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

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

- src/particles/particle.hpp

3.14 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

3.15 Particle_Handler Class Reference

Class that handles all particle-relevant operations.

`#include <particle_handler.hpp>`

Public Member Functions

- void **Load** (Input_Info_t *input_info, Domain *domain)
- void **Push** (double dt)
- long **nParticles** ()
- void **incrementNParticles** (int inc)
- void **SortParticles** (Particle_Compare comp)
- void **setPusher** (Pusher *pusher)
- void **clearGhosts** ()
- void **InterpolateEB** (Grid *grid)
- void **depositRhoJ** (Grid *grid, bool depositRho)
- std::vector< Particle > **getParticleVector** ()
- double **computeCFLTimestep** (Domain *domain)
- void **setParticleBoundaries** (BC_Particle **bc)
- void **executeParticleBoundaryConditions** ()

3.15.1 Detailed Description

Class that handles all particle-relevant operations.

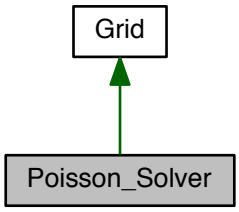
Particle handler handles all the particle operations. This includes deposition, boundary conditions, particle pushing, and communication between MPI nodes if needed

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

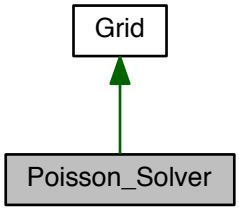
- src/particles/particle_handler.hpp
- src/particles/particle_handler.cpp

3.16 Poisson_Solver Class Reference

Inheritance diagram for Poisson_Solver:



Collaboration diagram for Poisson_Solver:



Public Member Functions

- **Poisson_Solver** (int ∗nxyz, int nGhosts, double ∗xyz0, double ∗Lxyz)
- void **initialize_poisson_fields** ()

Protected Member Functions

- void **run_poisson_solver_** (const int fieldID, double ∗∗∗u0, double ∗∗∗u1, double ∗∗∗R, double convergenceTol, double sourceMult)
- void **setPoissonFieldType_** ()
- void **setPoissonFieldPtr_** ()

Protected Attributes

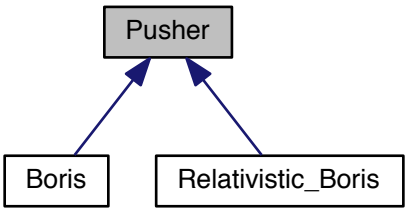
- double *** **phi1_**
- double *** **phi2_**
- double *** **Ax1_**
- double *** **Ay1_**
- double *** **Az1_**
- double *** **Ax2_**
- double *** **Ay2_**
- double *** **Az2_**
- const int **phi1ID_**
- const int **phi2ID_**
- const int **Ax1ID_**
- const int **Ay1ID_**
- const int **Az1ID_**
- const int **Ax2ID_**
- const int **Ay2ID_**
- const int **Az2ID_**

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

- src/poisson/poisson.hpp
- src/poisson/poisson.cpp

3.17 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

3.18 Random_Number_Generator Class Reference

Public Member Functions

- **Random_Number_Generator** (long seed)
- double **getUniform** ()
- double **getStandardNormal** ()
- double **getGaussian** (double mu, double sigma)
- RNG_State * **getRNGState** ()
- void **setRNGState** (RNG_State *state)

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

- src/utils/RNG.hpp
- src/utils/RNG.cpp

3.19 RegisterParticleBoundary Struct Reference

Public Member Functions

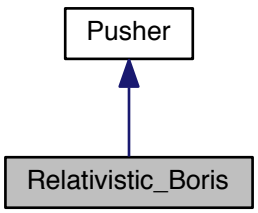
- **RegisterParticleBoundary** (const std::string &type, BC_Factory::Factory factory)

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

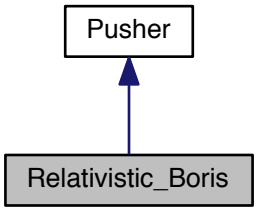
- src/boundaries/bc_factory.hpp

3.20 Relativistic_Boris Class Reference

Inheritance diagram for Relativistic_Boris:



Collaboration diagram for Relativistic_Boris:



Public Member Functions

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

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

- src/pusher/relativisticBoris.cpp

3.21 RNG_State Struct Reference

Public Attributes

- long int **idum**
- long int **idum2**
- long int **iy**
- long int **iv** [RNG_NTAB]
- double **z0**
- double **z1**
- bool **generate**

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

- src/utils/RNG.hpp