(Dust) Particles in Athena++

Chao-Chin Yang
Zhaohuan Zhu
University of Nevada, Las Vegas

James Stone
Princeton University

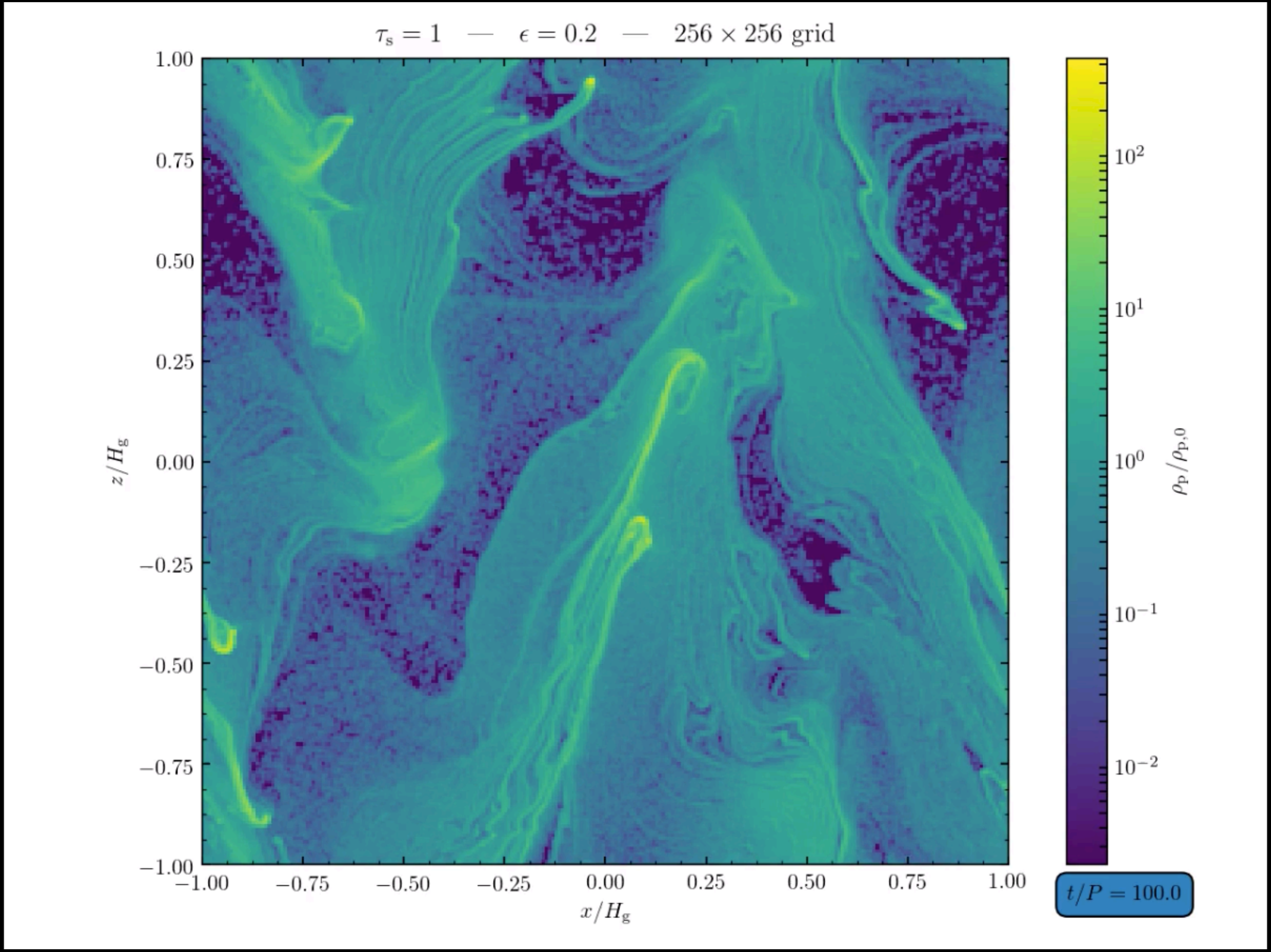
Dust-Gas Dynamics

- Lagrangian formulation
  - Lagrangian vs. two fluids
- Particle-Mesh method
  - Gas properties on each particle
  - Averaged particle properties in each gas cell
    - Triangle-Shaped Cloud (TSC)
- Stopping time $t_\mathrm{s} = 0$: tracer particles

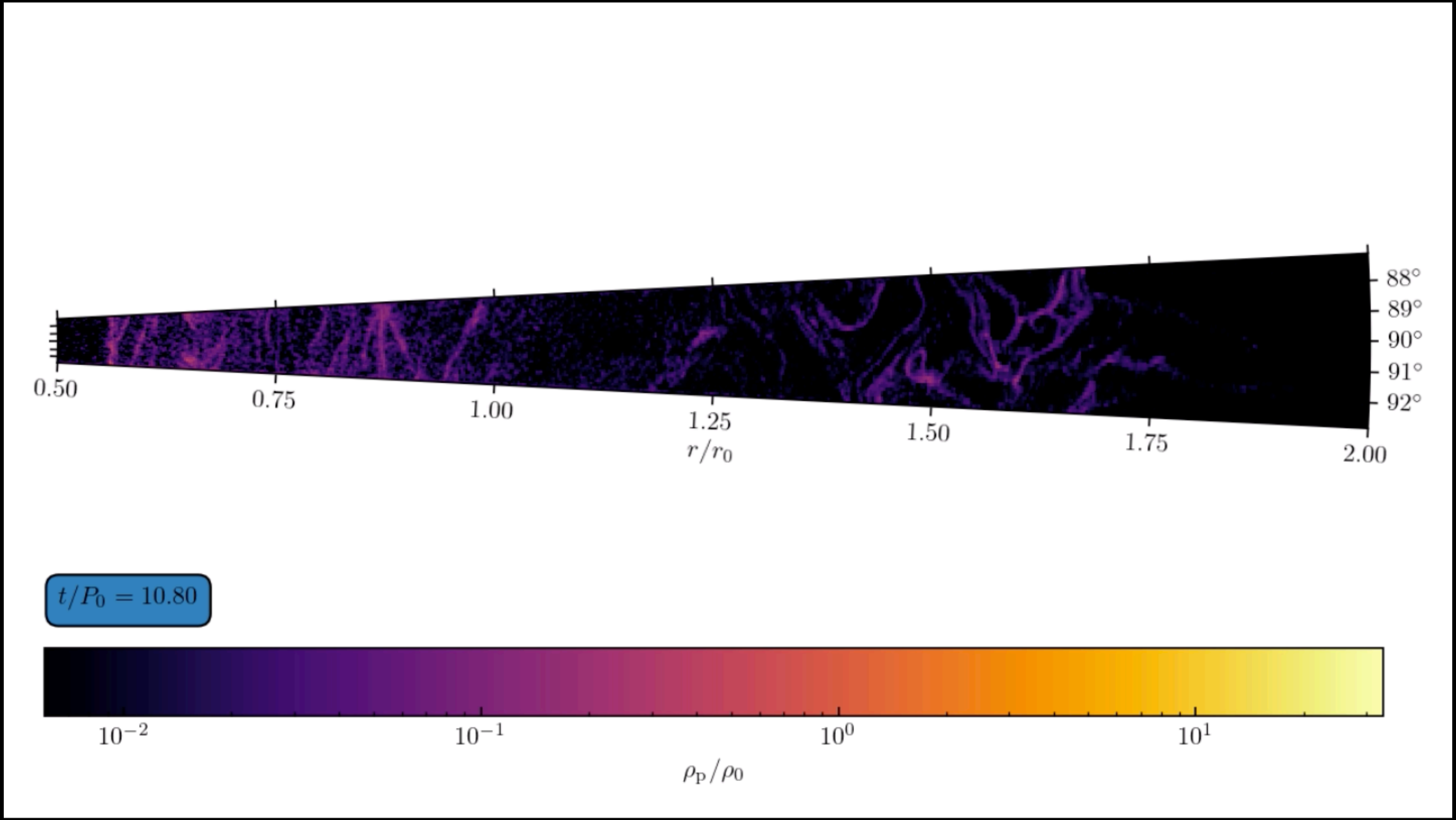
$$\frac{\mathrm{d}\overrightarrow{v}_\mathrm{p}}{\mathrm{d}t} = \frac{\overrightarrow{u}_\mathrm{g} - \overrightarrow{v}_\mathrm{p}}{t_\mathrm{s}}$$

$$\frac{\mathrm{d}\overrightarrow{u}_\mathrm{g}}{\mathrm{d}t} = \left(\frac{\rho_\mathrm{p}}{\rho_\mathrm{g}}\right)\frac{\overrightarrow{v}_\mathrm{p} - \overrightarrow{u}_\mathrm{g}}{t_\mathrm{s}}$$

Nonlinear Streaming Instability



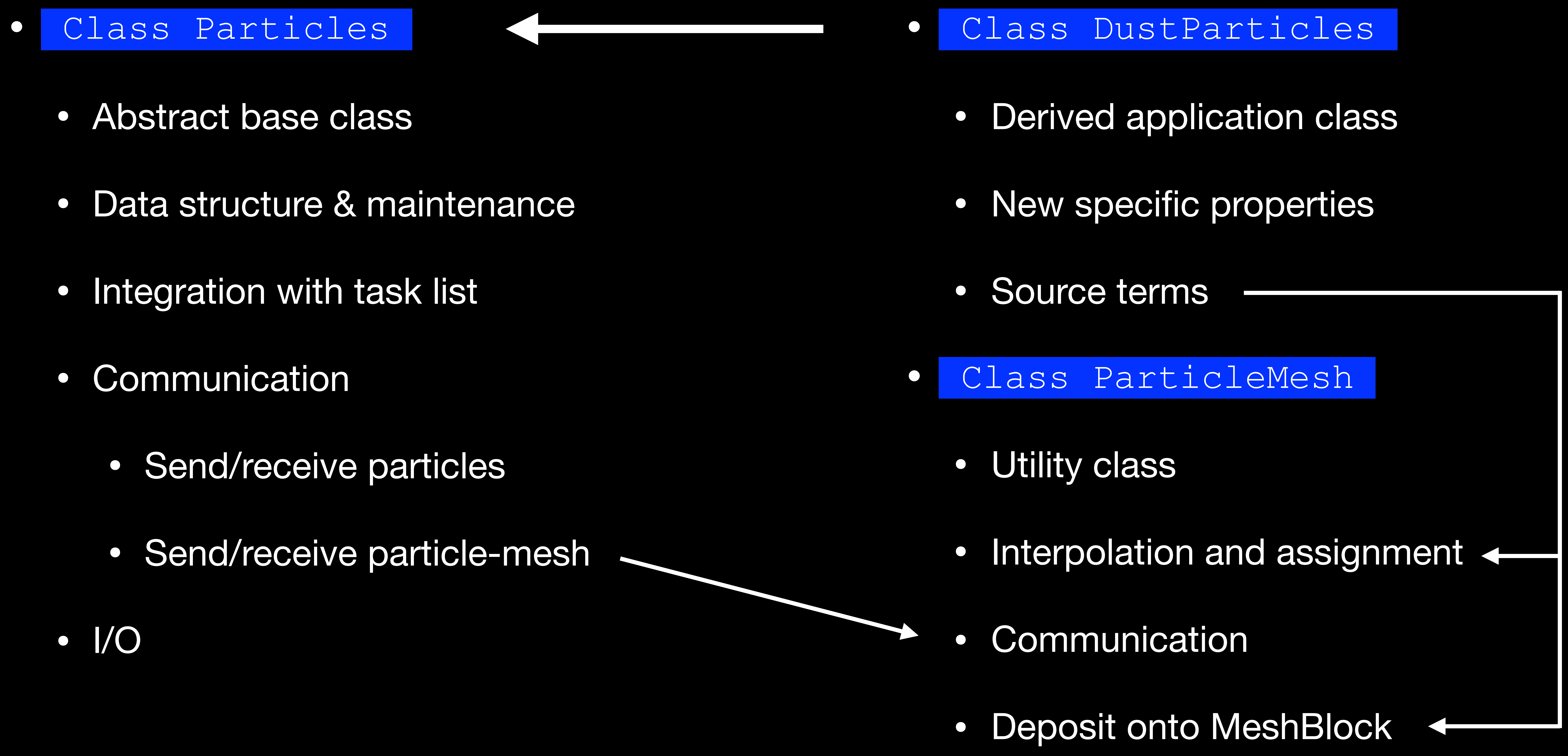
Global Spherical-polar Disk



Current Status

- Van Leer integrator with any reconstruction
- Particle-Mesh: both ways
- Static mesh refinement
- Message Passing Interface (MPI)
- Coordinates: Cartesian ready; spherical-polar under testing
- Useful: OpenMP threads block by block, and hence particles inside

Hierarchical Design



- `Class Particles`
  - Abstract base class
  - Data structure & maintenance
  - Integration with task list
  - Communication
    - Send/receive particles
    - Send/receive particle-mesh
  - I/O

- `Class DustParticles`
  - Derived application class
  - New specific properties
  - Source terms
- `Class ParticleMesh`
  - Utility class
  - Interpolation and assignment
  - Communication
  - Deposit onto MeshBlock

Data Structure

- Number of properties requested before MeshBlock construction
  - Integer properties: Particle ID, …
  - Real properties
    - Dynamical variables: Position `(xp, yp, zp)`, Velocity `(vpx, vpy, vpz)`, etc.
    - Auxiliary variables: always existent and follows each particle
    - Working arrays: garbage after each stage
  - Shorthands (shallow copies)

```
AthenaArray<> prop(nvar, nparmax);
```

- Number of particles in each MeshBlock dynamically maintained
  - `npar`: current number of particles
  - `nparmax`: capacity of the particle arrays

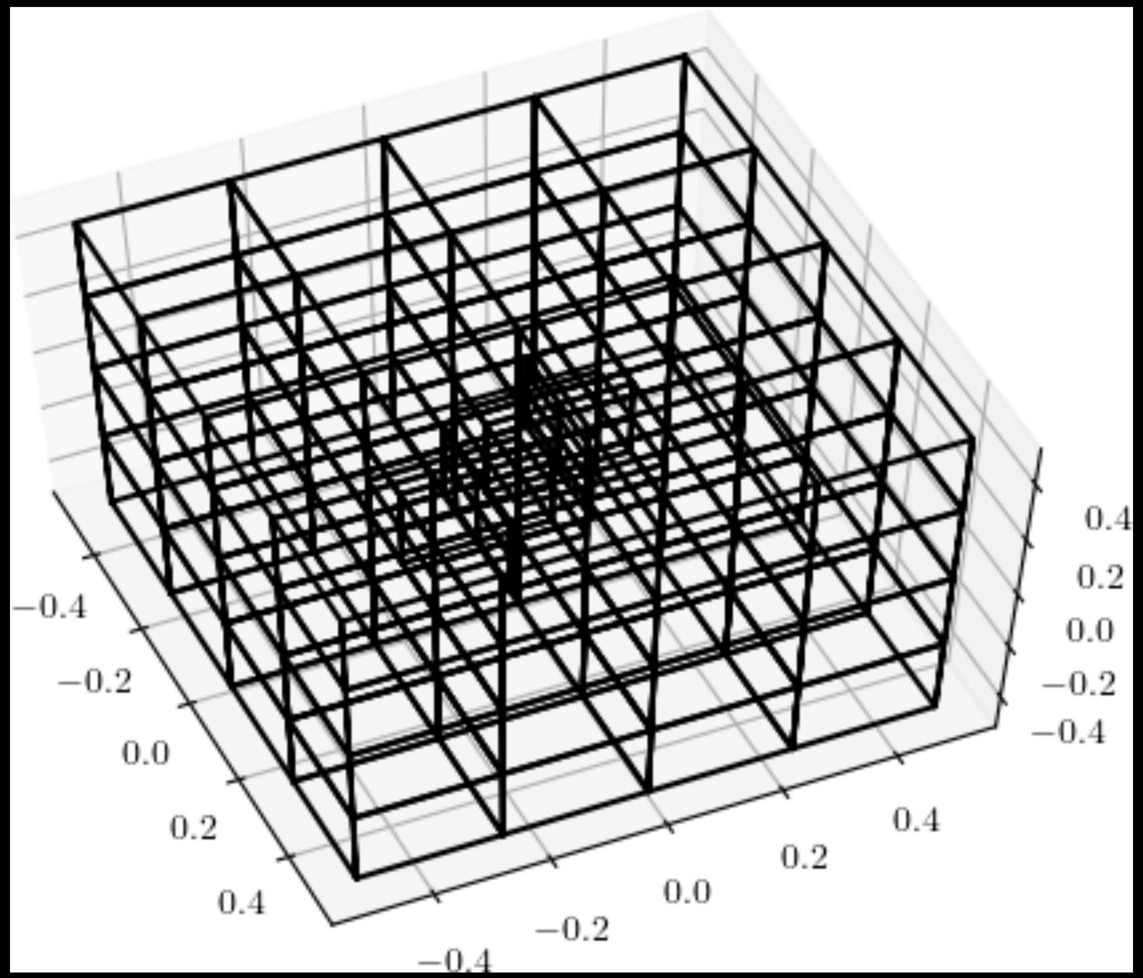
```
Void Particles::UpdateCapacity(int new_nparmax);
```

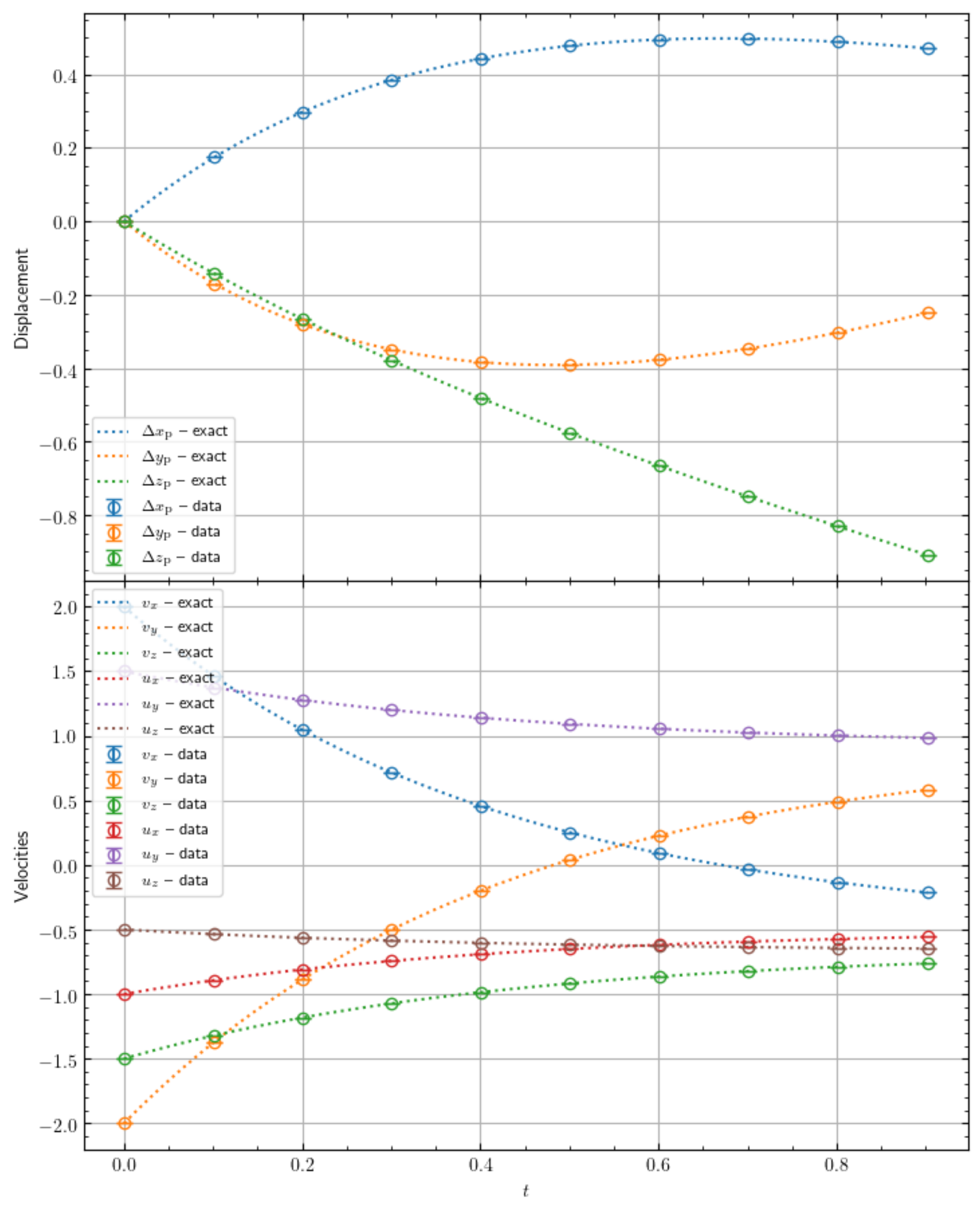
Outputs

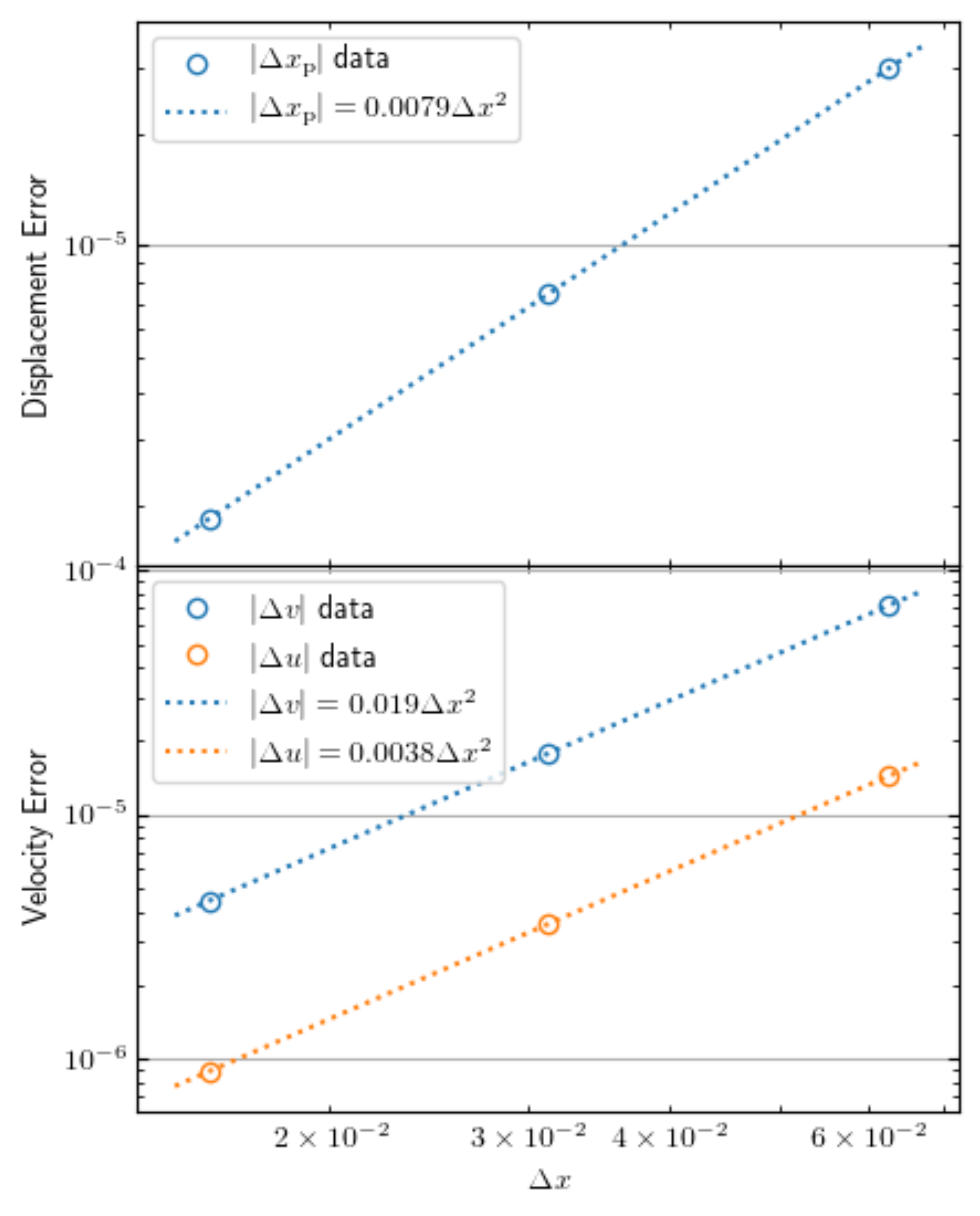
- Particle properties on mesh
  - Can be output in any format available to MeshBlock
  - Number density “`np`”
  - Velocity “`vp`” with respect to Coordinates.
  - Mass density “`rhop`”
  - `“prim” = (“rhop”, “vp1”, “vp2”, “vp3”)`

- Data of individual particles
  - Only formatted table
- Restart files
  - Not ready
  - Inhomogeneous data size between MeshBlocks

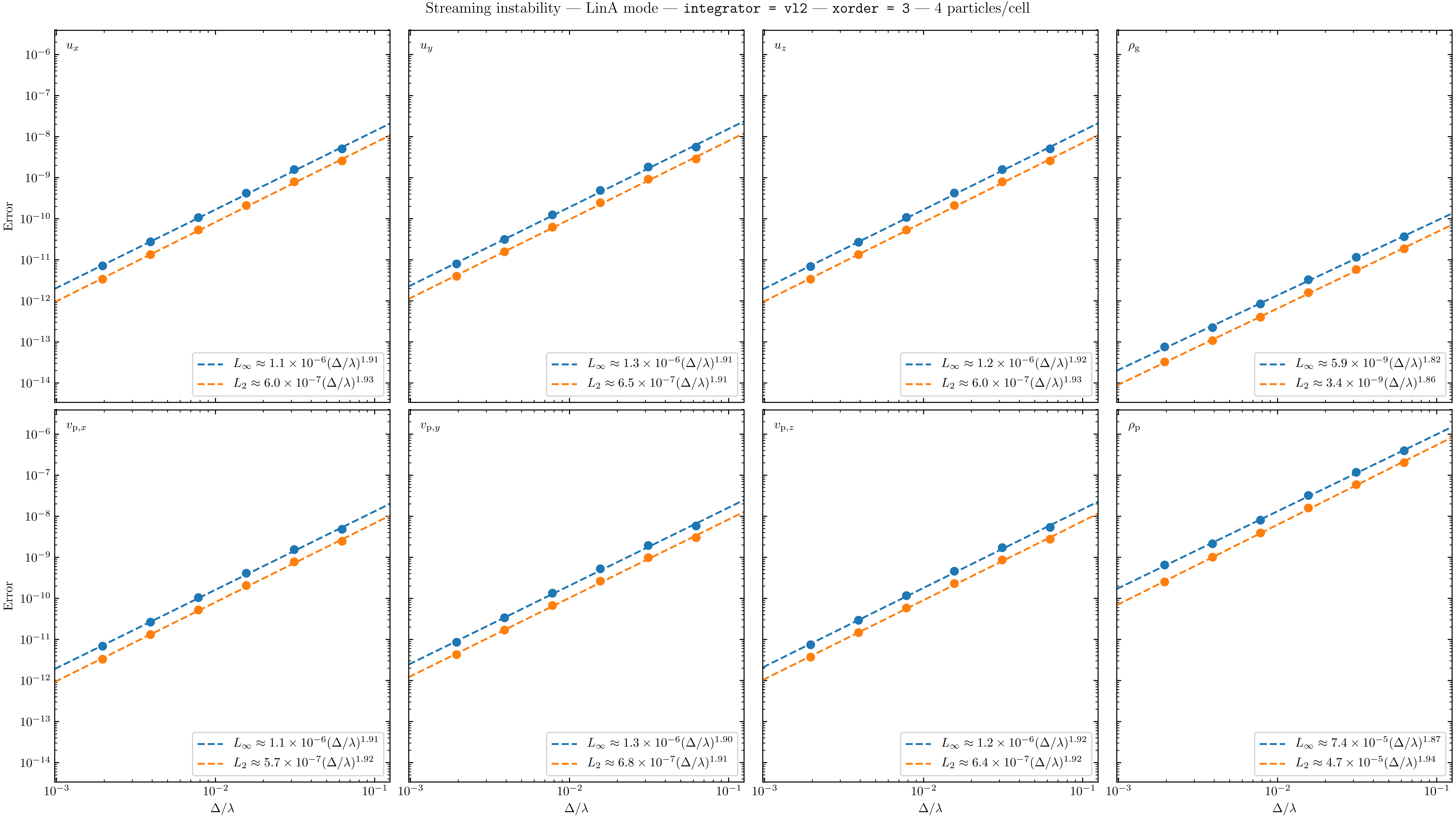
Uniform (Oblique) Streaming





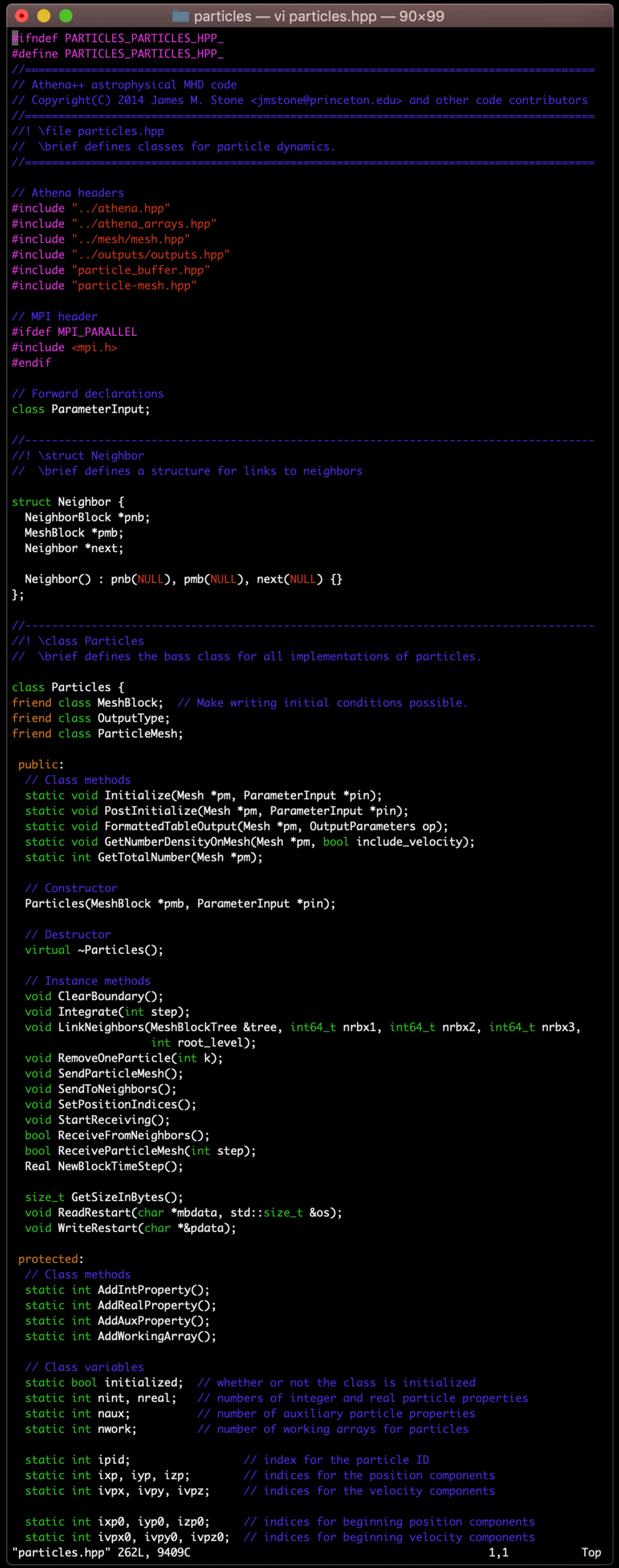


Linear Mode of the Streaming Instability

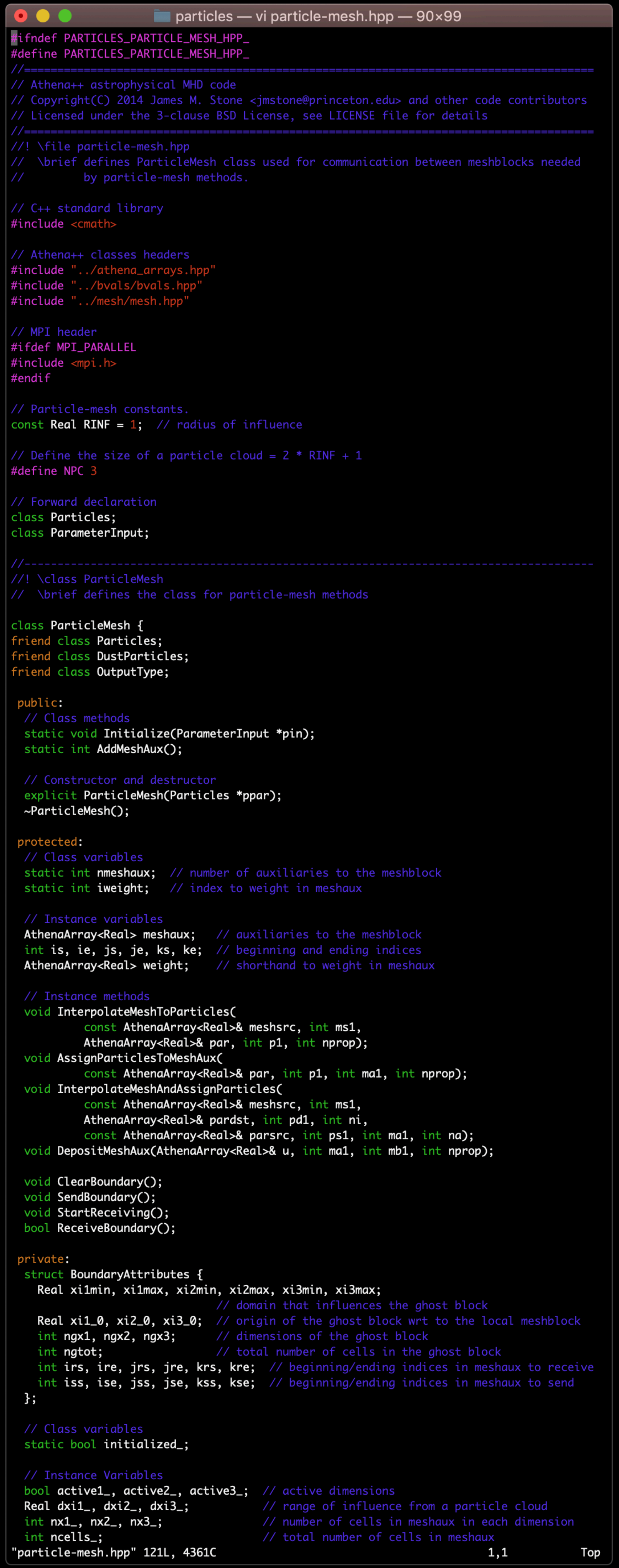


TODOs

- Spherical-polar and cylindrical coordinates
- Particle integrator in sync with any time integrator
- Non-uniform grid
- Restart files
- Boundary conditions for particles
    - Currently, only periodic or removed
- Adaptive mesh refinement
- More Particles outputs
- Optimization and documentation



```
#ifndef PARTICLES_PARTICLES_HPP_
#define PARTICLES_PARTICLES_HPP_
//========================================================================================
// Athena++ astrophysical MHD code
// Copyright(C) 2014 James M. Stone <jmstone@princeton.edu> and other code contributors
//========================================================================================
//! \file particles.hpp
//  \brief defines classes for particle dynamics.
//========================================================================================

// Athena headers
#include "../athena.hpp"
#include "../athena_arrays.hpp"
#include "../mesh/mesh.hpp"
#include "../outputs/outputs.hpp"
#include "particle_buffer.hpp"
#include "particle-mesh.hpp"

// MPI header
#ifdef MPI_PARALLEL
#include <mpi.h>
#endif

// Forward declarations
class ParameterInput;

//----------------------------------------------------------------------------------------
//! \struct Neighbor
//  \brief defines a structure for links to neighbors

struct Neighbor {
  NeighborBlock *pnb;
  MeshBlock *pmb;
  Neighbor *next;

  Neighbor() : pnb(NULL), pmb(NULL), next(NULL) {}
};

//----------------------------------------------------------------------------------------
//! \class Particles
//  \brief defines the bass class for all implementations of particles.

class Particles {
friend class MeshBlock;  // Make writing initial conditions possible.
friend class OutputType;
friend class ParticleMesh;

 public:
  // Class methods
  static void Initialize(Mesh *pm, ParameterInput *pin);
  static void PostInitialize(Mesh *pm, ParameterInput *pin);
  static void FormattedTableOutput(Mesh *pm, OutputParameters op);
  static void GetNumberDensityOnMesh(Mesh *pm, bool include_velocity);
  static int GetTotalNumber(Mesh *pm);

  // Constructor
  Particles(MeshBlock *pmb, ParameterInput *pin);

  // Destructor
  virtual ~Particles();

  // Instance methods
  void ClearBoundary();
  void Integrate(int step);
  void LinkNeighbors(MeshBlockTree &tree, int64_t nrbx1, int64_t nrbx2, int64_t nrbx3,
                     int root_level);
  void RemoveOneParticle(int k);
  void SendParticleMesh();
  void SendToNeighbors();
  void SetPositionIndices();
  void StartReceiving();
  bool ReceiveFromNeighbors();
  bool ReceiveParticleMesh(int step);
  Real NewBlockTimeStep();

  size_t GetSizeInBytes();
  void ReadRestart(char *mbdata, std::size_t &os);
  void WriteRestart(char *&pdata);

 protected:
  // Class methods
  static int AddIntProperty();
  static int AddRealProperty();
  static int AddAuxProperty();
  static int AddWorkingArray();

  // Class variables
  static bool initialized;  // whether or not the class is initialized
  static int nint, nreal;   // numbers of integer and real particle properties
  static int naux;          // number of auxiliary particle properties
  static int nwork;         // number of working arrays for particles

  static int ipid;                 // index for the particle ID
  static int ixp, iyp, izp;        // indices for the position components
  static int ivpx, ivpy, ivpz;     // indices for the velocity components

  static int ixp0, iyp0, izp0;     // indices for beginning position components
  static int ivpx0, ivpy0, ivpz0;  // indices for beginning velocity components
"particles.hpp" 262L, 9409C                                         1,1           Top
```



```
#ifndef PARTICLES_PARTICLE_MESH_HPP_
#define PARTICLES_PARTICLE_MESH_HPP_
//========================================================================================
// Athena++ astrophysical MHD code
// Copyright(C) 2014 James M. Stone <jmstone@princeton.edu> and other code contributors
// Licensed under the 3-clause BSD License, see LICENSE file for details
//========================================================================================
//! \file particle-mesh.hpp
//  \brief defines ParticleMesh class used for communication between meshblocks needed
//         by particle-mesh methods.

// C++ standard library
#include <cmath>

// Athena++ classes headers
#include "../athena_arrays.hpp"
#include "../bvals/bvals.hpp"
#include "../mesh/mesh.hpp"

// MPI header
#ifdef MPI_PARALLEL
#include <mpi.h>
#endif

// Particle-mesh constants.
const Real RINF = 1;  // radius of influence

// Define the size of a particle cloud = 2 * RINF + 1
#define NPC 3

// Forward declaration
class Particles;
class ParameterInput;

//----------------------------------------------------------------------------------------
//! \class ParticleMesh
//  \brief defines the class for particle-mesh methods

class ParticleMesh {
friend class Particles;
friend class DustParticles;
friend class OutputType;

 public:
  // Class methods
  static void Initialize(ParameterInput *pin);
  static int AddMeshAux();

  // Constructor and destructor
  explicit ParticleMesh(Particles *ppar);
  ~ParticleMesh();

 protected:
  // Class variables
  static int nmeshaux;  // number of auxiliaries to the meshblock
  static int iweight;   // index to weight in meshaux

  // Instance variables
  AthenaArray<Real> meshaux;   // auxiliaries to the meshblock
  int is, ie, js, je, ks, ke;  // beginning and ending indices
  AthenaArray<Real> weight;    // shorthand to weight in meshaux

  // Instance methods
  void InterpolateMeshToParticles(
           const AthenaArray<Real>& meshsrc, int ms1,
           AthenaArray<Real>& par, int p1, int nprop);
  void AssignParticlesToMeshAux(
           const AthenaArray<Real>& par, int p1, int ma1, int nprop);
  void InterpolateMeshAndAssignParticles(
           const AthenaArray<Real>& meshsrc, int ms1,
           AthenaArray<Real>& pardst, int pd1, int ni,
           const AthenaArray<Real>& parsrc, int ps1, int ma1, int na);
  void DepositMeshAux(AthenaArray<Real>& u, int ma1, int mb1, int nprop);

  void ClearBoundary();
  void SendBoundary();
  void StartReceiving();
  bool ReceiveBoundary();

 private:
  struct BoundaryAttributes {
    Real xi1min, xi1max, xi2min, xi2max, xi3min, xi3max;
                               // domain that influences the ghost block
    Real xi1_0, xi2_0, xi3_0;  // origin of the ghost block wrt to the local meshblock
    int ngx1, ngx2, ngx3;      // dimensions of the ghost block
    int ngtot;                 // total number of cells in the ghost block
    int irs, ire, jrs, jre, krs, kre;  // beginning/ending indices in meshaux to receive
    int iss, ise, jss, jse, kss, kse;  // beginning/ending indices in meshaux to send
  };

  // Class variables
  static bool initialized_;

  // Instance Variables
  bool active1_, active2_, active3_;  // active dimensions
  Real dxi1_, dxi2_, dxi3_;           // range of influence from a particle cloud
  int nx1_, nx2_, nx3_;               // number of cells in meshaux in each dimension
  int ncells_;                        // total number of cells in meshaux
"particle-mesh.hpp" 121L, 4361C                                     1,1           Top
```