spacex

Generated by Doxygen 1.8.14

Contents

Chapter 1

Template-SpaceX

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

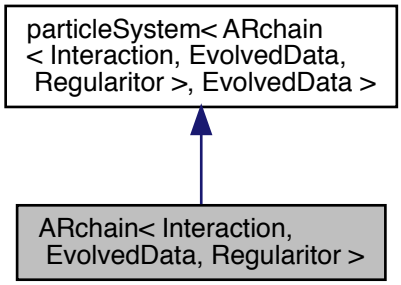
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

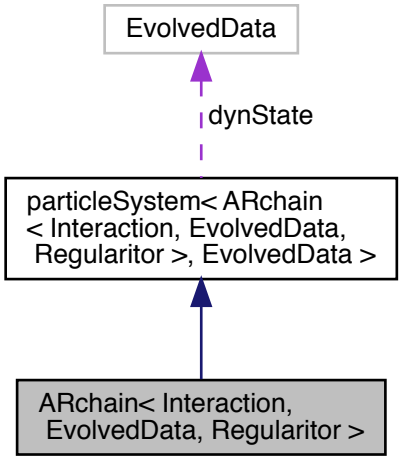
Class Documentation

4.1 ARchain< Interaction, EvolvedData, Regularitor > Class Template Reference

Inheritance diagram for ARchain< Interaction, EvolvedData, Regularitor >:



Collaboration diagram for ARchain< Interaction, EvolvedData, Regularitor >:



Public Types

- typedef EvolvedData::Scalar **Scalar**
- typedef EvolvedData::Vector **Vector**
- typedef EvolvedData::VectorArray **VectorArray**
- typedef EvolvedData::ScalarArray **ScalarArray**
- typedef std::array< size_t, EvolvedData::size()> **IndexArray**
- typedef std::array< Scalar, EvolvedData::volume()> **PlainArray**

Public Member Functions

- void **advancePos** (Scalar timeStepSize)
- void **advanceVel** (Scalar timeStepSize)
- const ARchain & **operator=** (const ARchain &other)
- std::istream & **read** (std::istream &)
- void **load** (PlainArray &data)
- Scalar **timeScale** (Scalar scale)
- PlainArray & **array** ()

Static Public Member Functions

- static constexpr size_t **size** ()

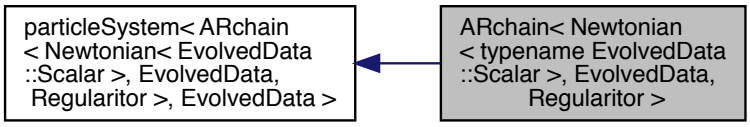
Additional Inherited Members

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

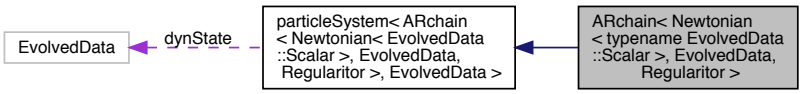
- particleSystem/ARchain.h

4.2 ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor > Class Template Reference

Inheritance diagram for ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >:



Collaboration diagram for ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >:



Public Types

- typedef EvolvedData::Scalar **Scalar**
- typedef EvolvedData::Vector **Vector**
- typedef EvolvedData::VectorArray **VectorArray**
- typedef EvolvedData::ScalarArray **ScalarArray**
- typedef std::array< size_t, EvolvedData::size()> **IndexArray**
- typedef std::array< Scalar, EvolvedData::volume()> **PlainArray**

Public Member Functions

- void **advancePos** (Scalar timeStepSize)
- void **advanceVel** (Scalar timeStepSize)
- const ARchain & **operator=** (const ARchain &other)
- std::istream & **read** (std::istream &)
- void **load** (PlainArray &data)
- Scalar **timeScale** (Scalar scale)
- PlainArray & **array** ()

Static Public Member Functions

- static constexpr size_t **size** ()

Additional Inherited Members

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

- particleSystem/ARchain.h

4.3 BSIterator< ParticSys, Integrator > Class Template Reference

Public Types

- typedef ParticSys::Scalar **Scalar**
- template<typename Scalar , size_t N>
  using **scalarArray** = std::array< Scalar, N >

Public Member Functions

- Scalar **iterate** (ParticSys &particles, Integrator &integrator, Scalar stepLength)
- void **setRelativeError** (Scalar relError)
- void **setAbsoluteError** (Scalar absError)

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

- ODEiterator/BSIterator.h

4.4 constIterator< ParticSys, Integrator > Class Template Reference

Most common iterator.

`#include <constIterator.h>`

Public Types

- typedef ParticSys::Scalar Scalar
  interface to iterate particle system for one step

Public Member Functions

- Scalar **iterate** (ParticSys &particles, Integrator &integrator, Scalar stepLength)

4.4.1 Detailed Description

template<typename ParticSys, typename Integrator>
class constIterator< ParticSys, Integrator >

Most common iterator.

Constant iterator keep the step length constant and integrate the particle system for one step.

4.4.2 Member Typedef Documentation

4.4.2.1 Scalar

```
template<typename ParticSys , typename Integrator >
typedef ParticSys::Scalar constIterator< ParticSys, Integrator >::Scalar
```

interface to iterate particle system for one step

Parameters

particles	Particle system needs evolution.
integrator	Integrator to integrate the particle system.
stepLength	Macro step length for iteration(Here, the step length of the integrator).

Returns

step length for next iteration.

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

- ODEiterator/constIterator.h

4.5 dynamics< DataType, N > Class Template Reference

Class of dynamical variable.

```
#include <dynamicState.h>
```

Public Types

- typedef DataType **Scalar**
- typedef vec3< Scalar > **Vector**
- typedef std::array< vec3< Scalar >, N > **VectorArray**
- typedef std::array< Scalar, N > **ScalarArray**

Public Member Functions

- std::array< Scalar, volume()> & array ()

  Transfer this class to a plain array.
- void initAddiVariable (ScalarArray &mass)

  Initialize extra user defined variables. Interface required for other class.
- void setZero ()

  Set all data to be zero.

Static Public Member Functions

- static constexpr size_t size ()

  Get the number of the particles.
- static constexpr size_t volume ()

  Get the total data number.

Public Attributes

- VectorArray pos

  Array of position of the particles. Element is 3D vector.
- VectorArray vel

  Array of velocity of the particles. Element is 3D vector.
- Scalar time {0.0}

  The physical time of the dynamic system.

4.5.1 Detailed Description

template<typename DataType, size_t N>
class dynamics< DataType, N >

Class of dynamical variable.

All variables in this class are physical quantities need to be evolved. If you want to create you own dynamic state, make sure remove constant quantities away from the class. The interface array() can be used to operate the data in the class as a plain array(for evolution). Extra constant physical quantities waste the calculation.

4.5.2 Member Function Documentation

4.5.2.1 array()

```
template<typename DataType , size_t N>
std::array<Scalar, volume()>& dynamics< DataType, N >::array ( )  [inline]
```

Transfer this class to a plain array.

Returns

The reference of head of this class, reinterpret as a plain array.

4.5.2.2 initAddiVariable()

```
template<typename DataType , size_t N>
void dynamics< DataType, N >::initAddiVariable (
            ScalarArray & mass )  [inline]
```

Initialize extra user defined variables. Interface required for other class.

Parameters

mass	The mass of particles, might be required for initialization.

4.5.2.3 size()

```
template<typename DataType , size_t N>
static constexpr size_t dynamics< DataType, N >::size ( )  [inline], [static]
```

Get the number of the particles.

Returns

The particle number.

4.5.2.4 volume()

```
template<typename DataType , size_t N>
static constexpr size_t dynamics< DataType, N >::volume ( )  [inline], [static]
```

Get the total data number.

Returns

The data number.

Here is the caller graph for this function:

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

- dynamicState.h

4.6 dynamicSystem< ParticSys, Integrator, ODEiterator > Class Template Reference

A wrapper to make particle system, integrator and ODE iterator work together.

`#include <dynamicSystem.h>`

Public Member Functions

- void advanceOneStep ()
  Advance the particle system for one step.
- void loadText (char const *initFilePath)
  Load particle system initial condition from file.
- void setStepLength (double)
  Set the step length.
- virtual ~dynamicSystem ()
  Default destructor, virtualize for inherent class.

Public Attributes

- double stepLength {0.0}
  Macro step size for ODE iterator.
- ParticSys particles
  Particle system.
- Integrator integrator
  Integrator.
- ODEiterator iterator
  ODE Iterator.

4.6.1 Detailed Description

template<typename ParticSys, typename Integrator, typename ODEiterator>
class dynamicSystem< ParticSys, Integrator, ODEiterator >

A wrapper to make particle system, integrator and ODE iterator work together.

4.6.2 Member Function Documentation

4.6.2.1 advanceOneStep()

```
template<typename ParticSys , typename Integrator , typename ODEiterator >
void dynamicSystem< ParticSys, Integrator, ODEiterator >::advanceOneStep ( )  [inline]
```

Advance the particle system for one step.

Advance the particle system with current steplength stepLength. The ODE iterator iterate the integrator to convergence by its own implement. The step length will also be updated by its own implement.

4.6.2.2 loadText()

```
template<typename ParticSys , typename Integrator , typename ODEiterator >
void dynamicSystem< ParticSys, Integrator, ODEiterator >::loadText (
            char const * initFilePath )
```

Load particle system initial condition from file.

This function will read and check the initial file header (begin with '#') and the particle number after the '#'. Pass the rest information to particles by operator '>>'. The way to load the initial condition depend on the implemet of the particles. If the initial condition read successfully. This function will call getInitStepLength() to set the initial step length.

Parameters

initFilePath	The relative path of initial conditions file

Exceptions

If	the partcile number in the header is inconsisitent with the size of particles, this function will throw an exception.

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

- dynamicSystem.h

4.7 errhand Class Reference

Public Member Functions

- **errhand** (std::string err_msg_input, const char *file_input, size_t line_input)
- std::string **get_msg** () const
- std::string **get_file** () const
- size_t **get_line** () const
- std::string **to_string_loc** (const char *obj)
- void **invoke_telegram_bot** ()
- void **print_to_stdout** ()

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

- errhand.h

4.8 GAR< DataType, N > Class Template Reference

Class of velocity dependent dynamical system with regularization variables.

```
#include <GAR.h>
```

Public Types

- typedef DataType **Scalar**
- typedef vec3< Scalar > **Vector**
- typedef std::array< vec3< Scalar >, N > **VectorArray**
- typedef std::array< Scalar, N > **ScalarArray**
- typedef std::array< size_t, N > **IndexArray**

Public Member Functions

- std::array< Scalar, volume()> & array ()

  Transfer this class to a plain array.
- void setZero ()

  Set all data to be zero.
- Scalar getOmega (const ScalarArray &mass)

  Calculate the regularization variable omega.
- void initAddiVariable (ScalarArray &mass)

  Initialize extra user defined variables. Interface required for other class.
- void toChain (GAR &chainData, IndexArray &index)

  Transfer Cartesian coordinate regularization system to chain regularization system.
- void toCartesian (GAR &cartesian, IndexArray &index)

  Transfer chain coordinate regularization system to Cartesian regularization system.
- void moveToCentralMassCoords (ScalarArray &mass)

  Move particles to central mass coordinates.

Static Public Member Functions

- static constexpr size_t size ()

  Get the number of the particles.
- static constexpr size_t volume ()

  Get the total data number.

Public Attributes

- VectorArray pos

  Array of position of the particles. Element is 3D vector.
- VectorArray vel

  Array of velocity of the particles. Element is 3D vector.
- VectorArray auxiVel

  Array of auxiliary velocity of the particles. Element is 3D vector.
- Scalar time {0.0}

  The physical time of the dynamic system.
- Scalar bindE {0.0}

  The binding energy(for regularization) of the dynamic system.
- Scalar omega {0.0}

  The regularization variable of the dynamic system.

4.8.1 Detailed Description

template<typename DataType, size_t N>
class GAR< DataType, N >

Class of velocity dependent dynamical system with regularization variables.

A simple extension of class dynamics in dynamicState.h. Used for regularization system. See detail in https://academic.oup.com/mnras/article/372/1/219/974304 .

4.8.2 Member Function Documentation

4.8.2.1 array()

```
template<typename DataType , size_t N>
std::array<Scalar, volume()>& GAR< DataType, N >::array ( )  [inline]
```

Transfer this class to a plain array.

Returns

The reference of head of this class, reinterpret as a plain array.

4.8.2.2 getOmega()

```
template<typename DataType , size_t N>
Scalar GAR< DataType, N >::getOmega (
            const ScalarArray & mass )  [inline]
```

Calculate the regularization variable omega.

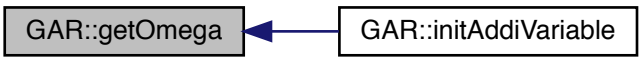
Parameters

mass	The mass of particles, might be required for calculation.

Returns

The calculated value of omega.

Here is the caller graph for this function:



4.8.2.3 initAddiVariable()

```
template<typename DataType , size_t N>
void GAR< DataType, N >::initAddiVariable (
            ScalarArray & mass )  [inline]
```

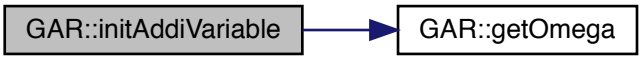
Initialize extra user defined variables. Interface required for other class.

Initialize regularizaiton variable bindE and omega.

Parameters

mass	The mass of particles, might be required for initialization.

Here is the call graph for this function:



4.8.2.4 moveToCentralMassCoords()

```
template<typename DataType , size_t N>
void GAR< DataType, N >::moveToCentralMassCoords (
            ScalarArray & mass )  [inline]
```

Move particles to central mass coordinates.

Move position, velocity and auxiliary velocity to central mass coordinates.

Parameters

mass	Mass of the particles required for moving.

4.8.2.5 size()

```
template<typename DataType , size_t N>
static constexpr size_t GAR< DataType, N >::size ( )  [inline], [static]
```

Get the number of the particles.

Returns

The particle number.

4.8.2.6 toCartesian()

```
template<typename DataType , size_t N>
void GAR< DataType, N >::toCartesian (
            GAR< DataType, N > & cartesian,
            IndexArray & index )  [inline]
```

Transfer chain coordinate regularization system to Cartesian regularization system.

Coordinate transformation. From chain to Cartesian. See details in https://link.springer.↩com/article/10.1007%2FBF00695714 .

Parameters

cartesian	The destination regularization system in Cartesian coordinates.
index	The maping index between Cartesian coordinates and chain coordinates.

4.8.2.7 toChain()

```
template<typename DataType , size_t N>
void GAR< DataType, N >::toChain (
            GAR< DataType, N > & chainData,
            IndexArray & index )  [inline]
```

Transfer Cartesian coordinate regularization system to chain regularization system.

Coordinate transformation. From Cartesian to chain. See details in https://link.springer.↩com/article/10.1007%2FBF00695714 .

Parameters

chainData	The destination regularization system in chain coordinates.
index	The maping index between Cartesian coordinates and chain coordinates.

4.8.2.8 volume()

```
template<typename DataType , size_t N>
static constexpr size_t GAR< DataType, N >::volume ( )  [inline], [static]
```

Get the total data number.

Returns

The data number.

Here is the caller graph for this function:

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

- particleSystem/GAR.h

4.9 logH< DynamicState > Class Template Reference

logH extention algorithmatic regularization interface

```
#include <regularization.h>
```

Public Types

- typedef DynamicState::Scalar **Scalar**

Public Member Functions

- Scalar getPhysicalPosTime (std::array< Scalar, size()> &mass, DynamicState &dyn, Scalar stepSize)

  Calculate the physical time for position advance from integration step size.
- Scalar getPhysicalVelTime (std::array< Scalar, size()> &mass, DynamicState &dyn, Scalar stepSize)

  Calculate the physical time for velocity advance from integration step size.

Static Public Member Functions

- static constexpr size_t **size** ()

4.9.1 Detailed Description

template<typename DynamicState>
class logH< DynamicState >

logH extention algorithmatic regularization interface

See detials in https://link.springer.com/article/10.1023%2FA%3A1008368322547 and http://iopscience.iop.org/article/10.1086/301102/meta .

4.9.2 Member Function Documentation

4.9.2.1 getPhysicalPosTime()

```
template<typename DynamicState >
Scalar logH< DynamicState >::getPhysicalPosTime (
            std::array< Scalar, size()> & mass,
            DynamicState & dyn,
            Scalar stepSize )  [inline]
```

Calculate the physical time for position advance from integration step size.

Parameters

mass	Array of particle mass.
dyn	Dynamic system contains position, velocity and regularization variables. See example class in dynamicState.h.
stepSize	Integration step size. This could not be the physical time. Look references for details in class despriction.

4.9.2.2 getPhysicalVelTime()

```
template<typename DynamicState >
Scalar logH< DynamicState >::getPhysicalVelTime (
            std::array< Scalar, size()> & mass,
            DynamicState & dyn,
            Scalar stepSize )  [inline]
```

Calculate the physical time for velocity advance from integration step size.

Parameters

mass	Array of particle mass.
dyn	Dynamic system contains position, velocity and regularization variables. See example class in dynamicState.h.
stepSize	Integration step size. This could not be the physical time. Look references for details in class despriction.

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

- particleSystem/regularization.h

4.10 Newtonian< Scalar > Class Template Reference

Marker of None velocity dependent force functor(c++ std11)

`#include <interaction.h>`

Public Member Functions

- void **operator()** ()

4.10.1 Detailed Description

template<typename Scalar>
class Newtonian< Scalar >

Marker of None velocity dependent force functor(c++ std11)

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

- interaction/interaction.h

4.11 chain::Node< Scalar > Struct Template Reference

Struture to store the relative distance and index of two particles.

`#include <chain.h>`

Public Attributes

- Scalar Rij
- size_t i
- size_t j
- bool available

4.11.1 Detailed Description

template<typename Scalar>
struct chain::Node< Scalar >

Struture to store the relative distance and index of two particles.

4.11.2 Member Data Documentation

4.11.2.1 available

```
template<typename Scalar>
bool chain::Node< Scalar >::available
```

State of node. If this node can be chained.

4.11.2.2 i

```
template<typename Scalar>
size_t chain::Node< Scalar >::i
```

Particle index.

4.11.2.3 j

```
template<typename Scalar>
size_t chain::Node< Scalar >::j
```

Particle index.

4.11.2.4 Rij

```
template<typename Scalar>
Scalar chain::Node< Scalar >::Rij
```

Relative distance of two particles.

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

- particleSystem/chain.h

4.12 NoRegu< DynamicState > Class Template Reference

Ordinary algorithmatic regularization interface.

`#include <regularization.h>`

Public Types

- typedef DynamicState::Scalar **Scalar**

Public Member Functions

- Scalar getPhysicalPosTime (std::array< Scalar, size()> &mass, DynamicState &dyn, Scalar stepSize)
  Calculate the physical time for position advance from integration step size.
- Scalar getPhysicalVelTime (std::array< Scalar, size()> &mass, DynamicState &dyn, Scalar stepSize)
  Calculate the physical time for velocity advance from integration step size.

Static Public Member Functions

- static constexpr size_t **size** ()

4.12.1 Detailed Description

template<typename DynamicState>
class NoRegu< DynamicState >

Ordinary algorithmatic regularization interface.

No regularization.

4.12.2 Member Function Documentation

4.12.2.1 getPhysicalPosTime()

```
template<typename DynamicState >
Scalar NoRegu< DynamicState >::getPhysicalPosTime (
            std::array< Scalar, size()> & mass,
            DynamicState & dyn,
            Scalar stepSize )  [inline]
```

Calculate the physical time for position advance from integration step size.

Parameters

mass	Array of particle mass.
dyn	Dynamic system contains position, velocity and regularization variables. See example class in dynamicState.h.
stepSize	Integration step size.

4.12.2.2 getPhysicalVelTime()

```
template<typename DynamicState >
Scalar NoRegu< DynamicState >::getPhysicalVelTime (
            std::array< Scalar, size()> & mass,
            DynamicState & dyn,
            Scalar stepSize )  [inline]
```

Calculate the physical time for velocity advance from integration step size.

Parameters

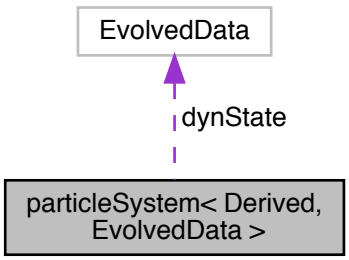
mass	Array of particle mass.
dyn	Dynamic system contains position, velocity and regularization variables. See example class in dynamicState.h.
stepSize	Integration step size.

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

- particleSystem/regularization.h

4.13 particleSystem< Derived, EvolvedData > Class Template Reference

Collaboration diagram for particleSystem< Derived, EvolvedData >:



Public Types

- typedef EvolvedData::Scalar **Scalar**
- typedef EvolvedData::Vector **Vector**
- typedef EvolvedData::VectorArray **VectorArray**
- typedef EvolvedData::ScalarArray **ScalarArray**
- typedef std::array< size_t, EvolvedData::size()> **IntArray**
- typedef std::array< Scalar, EvolvedData::volume()> **PlainArray**

Public Member Functions

- std::ostream & **write** (std::ostream &) const
- std::istream & **read** (std::istream &)
- PlainArray & **array** ()
- Scalar **timeScale** (Scalar scale)
- void **load** (PlainArray &data)
- const particleSystem & **operator=** (const particleSystem &other)

Static Public Member Functions

- static constexpr size_t **size** ()
- static constexpr size_t **volume** ()

Public Attributes

- EvolvedData **dynState**
- VectorArray & **pos**
- VectorArray & **vel**
- Scalar & **time**
- ScalarArray **mass**
- ScalarArray **radius**
- IntArray **type**

Protected Attributes

- VectorArray **acc**

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

- particleSystem.h

4.14 PN1th< Scalar > Class Template Reference

Post newtonian pair interaction functor(c++ std11)

`#include <interaction.h>`

Public Member Functions

- void operator() (Scalar m1, Scalar m2, Vector &dr, Vector &dv, Vector &v1, Vector &v2, Vector &acc1, Vector &acc2)

  Update the velocity dependent acceleration of particle 1 and 2.

4.14.1 Detailed Description

template<typename Scalar>
class PN1th< Scalar >

Post newtonian pair interaction functor(c++ std11)

4.14.2 Member Function Documentation

4.14.2.1 operator()()

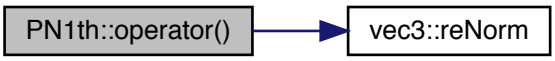
```
template<typename Scalar >
void PN1th< Scalar >::operator() (
            Scalar m1,
            Scalar m2,
            Vector & dr,
            Vector & dv,
            Vector & v1,
            Vector & v2,
            Vector & acc1,
            Vector & acc2 )  [inline]
```

Update the velocity dependent acceleration of particle 1 and 2.

Parameters

m1	Mass of particle 1.
m2	Mass of particle 2.
dr	Relative position pos1 - pos2.
dv	Relative velocity vel1 - vel2.
v1	Velocity of particle 1.
v2	Velocity of particle 2.
acc1	Velocity dependent acceleration of particle 1 as return value.
acc2	Velocity dependent acceleration of particle 1 as return value.

Here is the call graph for this function:



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

- interaction/interaction.h

4.15 reguDynamics< DataType, N > Class Template Reference

Class of dynamical system with regularization variables.

`#include <regularState.h>`

Public Types

- typedef DataType **Scalar**
- typedef vec3< Scalar > **Vector**
- typedef std::array< vec3< Scalar >, N > **VectorArray**
- typedef std::array< Scalar, N > **ScalarArray**
- typedef std::array< size_t, N > **IndexArray**

Public Member Functions

- std::array< Scalar, volume()> & array ()

  Transfer this class to a plain array.
- void setZero ()

  Set all data to be zero.
- Scalar getOmega (ScalarArray &mass)

  Calculate the regularization variable omega.
- void initAddiVariable (ScalarArray &mass)

  Initialize extra user defined variables. Interface required for other class.
- void toChain (reguDynamics &chainData, IndexArray &index)

  Transfer Cartesian coordinate regularization system to chain regularization system.
- void toCartesian (reguDynamics &cartesian, IndexArray &index)

  Transfer chain coordinate regularization system to Cartesian regularization system.
- void moveToCentralMassCoords (ScalarArray &mass)

  Move particles to central mass coordinates.

Static Public Member Functions

- static constexpr size_t size ()

  Get the number of the particles.
- static constexpr size_t volume ()

  Get the total data number.

Public Attributes

- VectorArray pos

  Array of position of the particles. Element is 3D vector.
- VectorArray vel

  Array of velocity of the particles. Element is 3D vector.
- Scalar time {0.0}

  The physical time of the dynamic system.
- Scalar bindE {0.0}

  The binding energy(for regularization) of the dynamic system.
- Scalar omega {0.0}

  The regularization variable of the dynamic system.

4.15.1 Detailed Description

template<typename DataType, size_t N>
class reguDynamics< DataType, N >

Class of dynamical system with regularization variables.

A simple extension of class dynamics in dynamicState.h. Used for regularization system. See detail in https://academic.oup.com/mnras/article/372/1/219/974304 .

4.15.2 Member Function Documentation

4.15.2.1 array()

```
template<typename DataType , size_t N>
std::array<Scalar, volume()>& reguDynamics< DataType, N >::array ( )  [inline]
```

Transfer this class to a plain array.

Returns

The reference of head of this class, reinterpret as a plain array.

4.15.2.2 getOmega()

```
template<typename DataType , size_t N>
Scalar reguDynamics< DataType, N >::getOmega (
            ScalarArray & mass )  [inline]
```

Calculate the regularization variable omega.

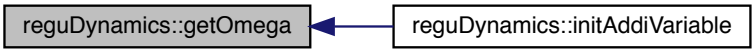
Parameters

mass	The mass of particles, might be required for calculation.

Returns

The calculated value of omega.

Here is the caller graph for this function:



4.15.2.3 initAddiVariable()

```
template<typename DataType , size_t N>
void reguDynamics< DataType, N >::initAddiVariable (
            ScalarArray & mass )  [inline]
```

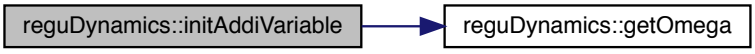
Initialize extra user defined variables. Interface required for other class.

Initialize regularizaiton variable bindE and omega.

Parameters

mass	The mass of particles, might be required for initialization.

Here is the call graph for this function:



4.15.2.4 moveToCentralMassCoords()

```
template<typename DataType , size_t N>
void reguDynamics< DataType, N >::moveToCentralMassCoords (
            ScalarArray & mass )  [inline]
```

Move particles to central mass coordinates.

Move position and velocity to central mass coordinates.

Parameters

mass	Mass of the particles required for moving.

4.15.2.5 size()

```
template<typename DataType , size_t N>
static constexpr size_t reguDynamics< DataType, N >::size ( )  [inline], [static]
```

Get the number of the particles.

Returns

The particle number.

4.15.2.6 toCartesian()

```
template<typename DataType , size_t N>
void reguDynamics< DataType, N >::toCartesian (
            reguDynamics< DataType, N > & cartesian,
            IndexArray & index )  [inline]
```

Transfer chain coordinate regularization system to Cartesian regularization system.

Coordinate transformation. From chain to Cartesian. See details in https://link.springer.↩com/article/10.1007%2FBF00695714 .

Parameters

cartesian	The destination regularization system in Cartesian coordinates.
index	The maping index between Cartesian coordinates and chain coordinates.

4.15.2.7 toChain()

```
template<typename DataType , size_t N>
void reguDynamics< DataType, N >::toChain (
            reguDynamics< DataType, N > & chainData,
            IndexArray & index )  [inline]
```

Transfer Cartesian coordinate regularization system to chain regularization system.

Coordinate transformation. From Cartesian to chain. See details in https://link.springer.↩com/article/10.1007%2FBF00695714 .

Parameters

chainData	The destination regularization system in chain coordinates.
index	The maping index between Cartesian coordinates and chain coordinates.

4.15.2.8 volume()

```
template<typename DataType , size_t N>
static constexpr size_t reguDynamics< DataType, N >::volume ( )  [inline], [static]
```

Get the total data number.

Returns

The data number.

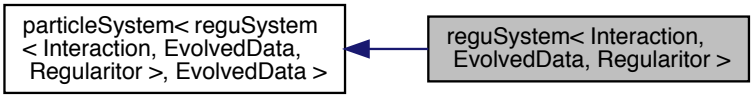
Here is the caller graph for this function:

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

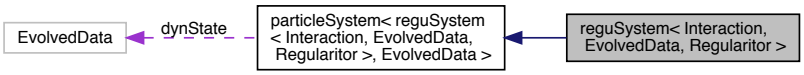
- particleSystem/regularState.h

4.16 reguSystem< Interaction, EvolvedData, Regularitor > Class Template Reference

Inheritance diagram for reguSystem< Interaction, EvolvedData, Regularitor >:



Collaboration diagram for reguSystem< Interaction, EvolvedData, Regularitor >:



Public Types

- typedef EvolvedData::Scalar **Scalar**
- typedef EvolvedData::Vector **Vector**
- typedef EvolvedData::VectorArray **VectorArray**
- typedef EvolvedData::ScalarArray **ScalarArray**
- typedef std::array< Scalar, EvolvedData::volume()> **PlainArray**

Public Member Functions

- void **advancePos** (Scalar timeStepSize)
- void **advanceVel** (Scalar timeStepSize)
- const reguSystem & **operator=** (const reguSystem &other)
- std::istream & **read** (std::istream &)
- void **load** (PlainArray &data)
- Scalar **timeScale** (Scalar scale)

Static Public Member Functions

- static constexpr size_t **size** ()
- static constexpr size_t **volume** ()

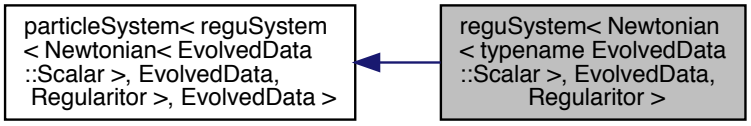
Additional Inherited Members

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

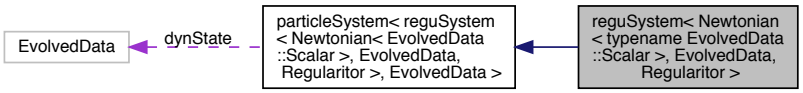
- particleSystem/reguSystem.h

4.17 reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor > Class Template Reference

Inheritance diagram for reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >:



Collaboration diagram for reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >:



Public Types

- typedef EvolvedData::Scalar **Scalar**
- typedef EvolvedData::Vector **Vector**
- typedef EvolvedData::VectorArray **VectorArray**
- typedef EvolvedData::ScalarArray **ScalarArray**
- typedef std::array< Scalar, EvolvedData::volume()> **PlainArray**

Public Member Functions

- void **advancePos** (Scalar timeStepSize)
- void **advanceVel** (Scalar timeStepSize)
- std::istream & **read** (std::istream &)
- void **load** (PlainArray &data)
- Scalar **timeScale** (Scalar scale)

Static Public Member Functions

- static constexpr size_t **size** ()
- static constexpr size_t **volume** ()

Additional Inherited Members

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

- particleSystem/reguSystem.h

4.18 symplectic10th< ParticSys > Class Template Reference

Tenth order symplectic integrator.

`#include <symplectic10th.h>`

Public Member Functions

- void integrate (ParticSys &particles, double stepLength)
  Interface to integrate particle system.

Static Public Attributes

- static const int order {10}
  Order of the integrator.

4.18.1 Detailed Description

template<typename ParticSys>
class symplectic10th< ParticSys >

Tenth order symplectic integrator.

4.18.2 Member Function Documentation

4.18.2.1 integrate()

```
template<typename ParticSys >
void symplectic10th< ParticSys >::integrate (
            ParticSys & particles,
            double stepLength )
```

Interface to integrate particle system.

This function integrate the particle system for one step with DKD leapfrog second order symplectic algorithm.

Parameters

particles	Particle system need to be integrated.
stepLength	Step size for integration.

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

- integrator/symplectic/symplectic10th.h

4.19 symplectic2th< ParticSys > Class Template Reference

Second order symplectic integrator.

`#include <symplectic2th.h>`

Public Member Functions

- void integrate (ParticSys &particles, Scalar stepLength)
  Interface to integrate particle system.

Static Public Attributes

- static const int order {2}
  Order of the integrator.

4.19.1 Detailed Description

template<typename ParticSys>
class symplectic2th< ParticSys >

Second order symplectic integrator.

4.19.2 Member Function Documentation

4.19.2.1 integrate()

```
template<typename ParticSys >
void symplectic2th< ParticSys >::integrate (
            ParticSys & particles,
            Scalar stepLength )
```

Interface to integrate particle system.

This function integrate the particle system for one step with DKD leapfrog second order symplectic algorithm.

Parameters

particles	Particle system need to be integrated.
stepLength	Step size for integration.

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

- integrator/symplectic/symplectic2th.h

4.20 symplectic4th< ParticSys > Class Template Reference

Fourth order symplectic integrator.

`#include <symplectic4th.h>`

Public Member Functions

- void integrate (ParticSys &particles, double stepLength)
  Interface to integrate particle system.

Static Public Attributes

- static const int order {4}
  Order of the integrator.

4.20.1 Detailed Description

template<typename ParticSys>
class symplectic4th< ParticSys >

Fourth order symplectic integrator.

4.20.2 Member Function Documentation

4.20.2.1 integrate()

```
template<typename ParticSys >
void symplectic4th< ParticSys >::integrate (
            ParticSys & particles,
            double stepLength )
```

Interface to integrate particle system.

This function integrate the particle system for one step with DKD leapfrog second order symplectic algorithm.

Parameters

particles	Particle system need to be integrated.
stepLength	Step size for integration.

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

- integrator/symplectic/symplectic4th.h

4.21 symplectic6th< ParticSys > Class Template Reference

Sixth order symplectic integrator.

`#include <symplectic6th.h>`

Static Public Attributes

- static const int order {6}

  Order of the integrator.

4.21.1 Detailed Description

template<typename ParticSys>
class symplectic6th< ParticSys >

Sixth order symplectic integrator.

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

- integrator/symplectic/symplectic6th.h

4.22 symplectic8th< ParticSys > Class Template Reference

Eighth order symplectic integrator.

`#include <symplectic8th.h>`

Public Member Functions

- void integrate (ParticSys &particles, double stepLength)

  Interface to integrate particle system.

Static Public Attributes

- static const int order {8}
  Order of the integrator.

4.22.1 Detailed Description

template<typename ParticSys>
class symplectic8th< ParticSys >

Eighth order symplectic integrator.

4.22.2 Member Function Documentation

4.22.2.1 integrate()

```
template<typename ParticSys >
void symplectic8th< ParticSys >::integrate (
            ParticSys & particles,
            double stepLength )
```

Interface to integrate particle system.

This function integrate the particle system for one step with DKD leapfrog second order symplectic algorithm.

Parameters

particles	Particle system need to be integrated.
stepLength	Step size for integration.

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

- integrator/symplectic/symplectic8th.h

4.23 TTL< DynamicState > Class Template Reference

Time Transform Leapfrog algorithmatic regularization interface.

```
#include <regularization.h>
```

Public Types

- typedef DynamicState::Scalar **Scalar**

Public Member Functions

- Scalar getPhysicalPosTime (std::array< Scalar, size()> &mass, DynamicState &dyn, Scalar stepSize)
  Calculate the physical time for position advance from integration step size.
- Scalar getPhysicalVelTime (std::array< Scalar, size()> &mass, DynamicState &dyn, Scalar stepSize)
  Calculate the physical time for velocity advance from integration step size.

Static Public Member Functions

- static constexpr size_t **size** ()

4.23.1 Detailed Description

template<typename DynamicState>
class TTL< DynamicState >

Time Transform Leapfrog algorithmatic regularization interface.

See detials in https://link.springer.com/article/10.1023%2FA%3A1021149313347 .

4.23.2 Member Function Documentation

4.23.2.1 getPhysicalPosTime()

```
template<typename DynamicState >
Scalar TTL< DynamicState >::getPhysicalPosTime (
            std::array< Scalar, size()> & mass,
            DynamicState & dyn,
            Scalar stepSize )  [inline]
```

Calculate the physical time for position advance from integration step size.

Parameters

mass	Array of particle mass.
dyn	Dynamic system contains position, velocity and regularization variables. See example class in dynamicState.h.
stepSize	Integration step size. This could not be the physical time. Look references for details in class despriction.

4.23.2.2 getPhysicalVelTime()

```
template<typename DynamicState >
```

```
Scalar TTL< DynamicState >::getPhysicalVelTime (
            std::array< Scalar, size()> & mass,
            DynamicState & dyn,
            Scalar stepSize )  [inline]
```

Calculate the physical time for velocity advance from integration step size.

Parameters

mass	Array of particle mass.
dyn	Dynamic system contains position, velocity and regularization variables. See example class in dynamicState.h.
stepSize	Integration step size. This could not be the physical time. Look references for details in class despriction.

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

- particleSystem/regularization.h

4.24 vec3< T > Struct Template Reference

Self 3D vector class.

`#include <vector3.h>`

Public Member Functions

- **vec3** (T vx, T vy, T vz)
- **vec3** (const vec3 &v)
- vec3 operator+ (const vec3 &v) const
  Addition by wise.
- vec3 operator- (const vec3 &v) const
  Subtraction by wise.
- vec3 operator/ (const vec3 &v) const
  Divition by wise.
- vec3 operator+ (const double c) const
  Add scalar by wise.
- vec3 operator- (const double c) const
  Subtract scalar by wise.
- vec3 operator∗ (const double c) const
  Multiply scalar by wise.
- vec3 operator/ (const double c) const
  Divide scalar by wise.
- vec3 operator- () const
  Opposite vector.
- vec3 operator^ (const vec3 &v) const
  Cross product.
- vec3 abs () const
  Absolute value by wise.

- const vec3 & **operator+=** (const vec3 &v)
- const vec3 & **operator-=** (const vec3 &v)
- const vec3 & **operator/=** (const vec3 &v)
- const vec3 & **operator+=** (const double c)
- const vec3 & **operator-=** (const double c)
- const vec3 & **operator*=** (const double c)
- const vec3 & **operator/=** (const double c)
- const vec3 & **operator=** (const vec3 &v)
- double operator* (const vec3 &v) const

  Inner product.
- double norm () const

  Calculate the norm.
- double normSquare () const

  Calcualte the square of the norm.
- double reNorm () const

  Calculate the inverse of the norm.
- void **setZero** ()

Public Attributes

- T **x**
- T **y**
- T **z**

Friends

- vec3 **operator+** (const double c, const vec3 &v)
- vec3 **operator-** (const double c, const vec3 &v)
- vec3 **operator*** (const double c, const vec3 &v)
- std::ostream & operator<< (std::ostream &output, const vec3 &v)

  Output to ostream.
- std::istream & operator>> (std::istream &input, vec3 &v)

  Input from istream.

4.24.1 Detailed Description

template<typename T>
struct vec3< T >

Self 3D vector class.

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

- vector3.h