spacehub

Generated by Doxygen 1.8.14

Contents

Chapter 1

Template-SpaceX

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 NOTICE Namespace Reference

Functions

- void Telegram (const char *host, const char *msg)
- void Title (const char *T)
- void SubTitle (const char *T)
- void EraseLine ()
- void Line ()
- void SubLine ()
- void RunInfo (double timeLimit, double outputsize_terval, double tolerance)

Variables

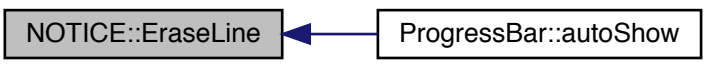
- constexpr size_t WIDTH = 80
- bool Message

6.1.1 Function Documentation

6.1.1.1 EraseLine()

```
void NOTICE::EraseLine ( )  [inline]
```

Here is the caller graph for this function:



6.1.1.2 Line()

```
void NOTICE::Line ( )  [inline]
```

6.1.1.3 RunInfo()

```
void NOTICE::RunInfo (
            double timeLimit,
            double outputsize_terval,
            double tolerance )  [inline]
```

6.1.1.4 SubLine()

```
void NOTICE::SubLine ( )  [inline]
```

6.1.1.5 SubTitle()

```
void NOTICE::SubTitle (
            const char * T )  [inline]
```

6.1.1.6 Telegram()

```
void NOTICE::Telegram (
            const char * host,
            const char * msg )  [inline]
```

6.1.1.7 Title()

```
void NOTICE::Title (
            const char * T )  [inline]
```

6.1.2 Variable Documentation

6.1.2.1 Message

```
bool NOTICE::Message
```

6.1.2.2 WIDTH

```
constexpr size_t NOTICE::WIDTH = 80
```

6.2 SpaceH Namespace Reference

Namespaces

- chain

Classes

- struct ArrayWrapper
- struct ArrayWrapper< T, DYNAMICAL >
- struct EmptyForce
- struct ExtVelDepForce
- struct ExtVelIndepForce
- struct Force
- struct get_value_type
- class Interaction
- struct kahan

  Kahan number.
- struct NewtonForce
- class PostNewtonian

  Post newtonian pair interaction functor(c++ std11)
- struct ProtoType
- struct VelDepForce
- struct VelIndepForce

Enumerations

- enum PARTICTYPE {
  NEUTRONSTAR, STAR, BLACKHOLE, POINT,
  NONE = 0 }
- enum EVENTTYPE {
  TDE, MERGE, ESCAPE, DISRUPTED,
  UNEVENTFUL, HVS }
- enum INTEGRATORTYPE {
  INTEGRATORTYPE::DKDLEAPFROG, INTEGRATORTYPE::KDKLEAPFROG, INTEGRATORTYPE::SYM4,
  INTEGRATORTYPE::PEFRL,
  INTEGRATORTYPE::SYM6, INTEGRATORTYPE::SYM8, INTEGRATORTYPE::SYM10 }
- enum SYSTEMTYPE { SYSTEMTYPE::PLAIN, SYSTEMTYPE::CHAIN }
- enum REGUTYPE { REGUTYPE::LOGH, REGUTYPE::TTL, REGUTYPE::NONE }
- enum ITERTYPE { ITERTYPE::BSITER, ITERTYPE::SEQITER }
- enum DATASTRUCT { DATASTRUCT::PLAIN =0, DATASTRUCT::CHAIN }

Functions

- template<typename T1 , typename T2 >
  const T2 min (const T1 &x, const T2 &y)

  Self min()
- template<typename T1 , typename T2 >
  const T2 max (const T1 &x, const T2 &y)

  Self max()
- template<class T >
  const T abs (const T &x)

  Self abs()
- template<typename Scalar1 , typename Scalar2 >
  void advanceScalar (Scalar1 &var, Scalar2 increase)

  Self swap()
- template<typename Scalar , typename Vector1 , typename Vector2 >
  void advanceVector (Vector1 &var, const Vector2 &increase, Scalar stepSize)
- template<typename ScalarArray , typename VectorArray >
  void moveToCMCoord (const ScalarArray &mass, VectorArray &phyVar)

  Move variables to central mass coordinates.
- template<typename Scalar , size_t N>
  double getKineticEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &vel)

  Calculate the kinetic energy of particles.
- template<typename Scalar , size_t N>
  double getPotentialEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &pos)

  Calculate the potential energy of particles.
- template<typename Scalar , size_t N>
  double getTotalEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &pos, const std::array< vec3< Scalar >, N > &vel)

  Calculate the total(potential + kinetic) energy of particles.
- template<typename T >
  void print (T &var)

  print an array. Used for debug

Variables

- constexpr double INV_C = 1 / C
- constexpr double INV_C2 = INV_C * INV_C
- constexpr double INV_C3 = INV_C2 * INV_C
- constexpr double INV_C4 = INV_C3 * INV_C
- constexpr double INV_C5 = INV_C4 * INV_C
- constexpr size_t DYNAMICAL = 0

6.2.1 Enumeration Type Documentation

6.2.1.1 DATASTRUCT

```
enum SpaceH::DATASTRUCT  [strong]
```

Enumerator

PLAIN	
CHAIN	

6.2.1.2 EVENTTYPE

```
enum SpaceH::EVENTTYPE
```

Enumerator

TDE	
MERGE	
ESCAPE	
DISRUPTED	
UNEVENTFUL	
HVS	

6.2.1.3 INTEGRATORTYPE

```
enum SpaceH::INTEGRATORTYPE  [strong]
```

Enumerator

DKDLEAPFROG	
KDKLEAPFROG	
SYM4	
PEFRL	
SYM6	
SYM8	
SYM10	

6.2.1.4 ITERTYPE

```
enum SpaceH::ITERTYPE  [strong]
```

Enumerator

BSITER	
SEQITER	

6.2.1.5 PARTICTYPE

```
enum SpaceH::PARTICTYPE
```

Enumerator

NEUTRONSTAR	
STAR	
BLACKHOLE	
POINT	
NONE	

6.2.1.6 REGUTYPE

```
enum SpaceH::REGUTYPE  [strong]
```

Enumerator

LOGH	
TTL	
NONE	

6.2.1.7 SYSTEMTYPE

```
enum SpaceH::SYSTEMTYPE  [strong]
```

Enumerator

PLAIN	
CHAIN	

6.2.2 Function Documentation

6.2.2.1 abs()

```
template<class T >
const T SpaceH::abs (
            const T & x )  [inline]
```

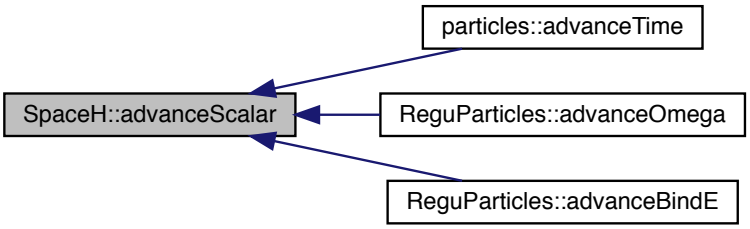
Self abs()

6.2.2.2 advanceScalar()

```
template<typename Scalar1 , typename Scalar2 >
void SpaceH::advanceScalar (
            Scalar1 & var,
            Scalar2 increase )  [inline]
```

Self swap()

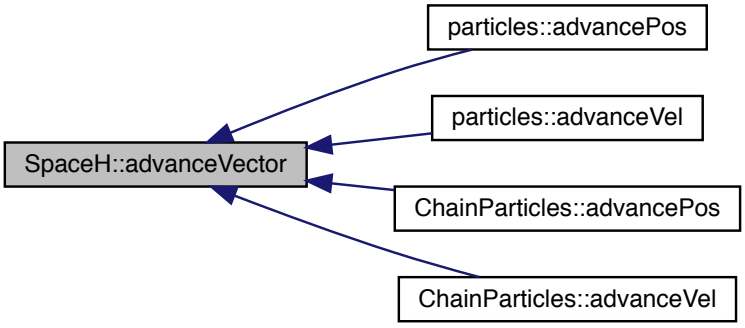
Here is the caller graph for this function:



6.2.2.3 advanceVector()

```
template<typename Scalar , typename Vector1 , typename Vector2 >
void SpaceH::advanceVector (
            Vector1 & var,
            const Vector2 & increase,
            Scalar stepSize )  [inline]
```

Here is the caller graph for this function:



6.2.2.4 getKineticEnergy()

```
template<typename Scalar , size_t N>
double SpaceH::getKineticEnergy (
            const std::array< Scalar, N > & mass,
            const std::array< vec3< Scalar >, N > & vel )
```

Calculate the kinetic energy of particles.

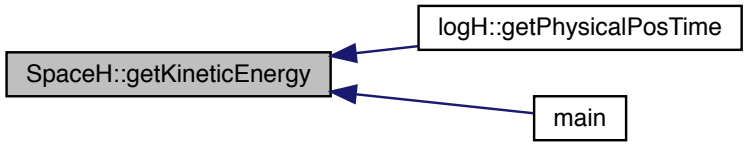
Parameters

mass	Array of mass.
vel	Array of velocity.

Returns

The kinetic energy.

Here is the caller graph for this function:



6.2.2.5 getPotentialEnergy()

```
template<typename Scalar , size_t N>
double SpaceH::getPotentialEnergy (
            const std::array< Scalar, N > & mass,
            const std::array< vec3< Scalar >, N > & pos )
```

Calculate the potential energy of particles.

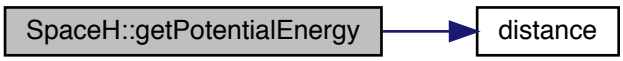
Parameters

mass	Array of mass.
pos	Array of position.

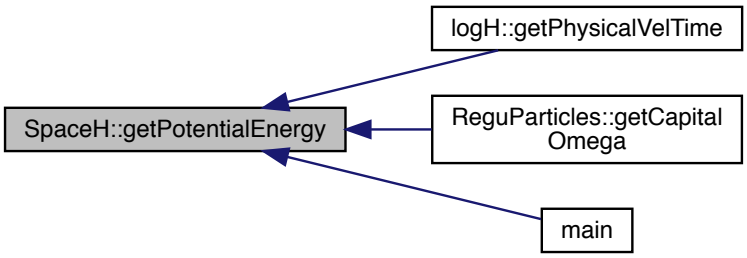
Returns

The potential energy.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2.6 getTotalEnergy()

```
template<typename Scalar , size_t N>
double SpaceH::getTotalEnergy (
            const std::array< Scalar, N > & mass,
            const std::array< vec3< Scalar >, N > & pos,
            const std::array< vec3< Scalar >, N > & vel )  [inline]
```

Calculate the total(potential + kinetic) energy of particles.

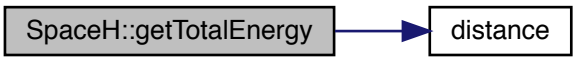
Parameters

mass	Array of mass.
pos	Array of position.
vel	Array of velocity.

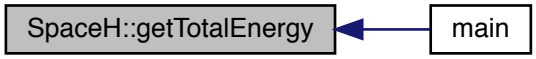
Returns

The total energy.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2.7 max()

```
template<typename T1 , typename T2 >
const T2 SpaceH::max (
            const T1 & x,
            const T2 & y )  [inline]
```

Self max()

6.2.2.8 min()

```
template<typename T1 , typename T2 >
const T2 SpaceH::min (
            const T1 & x,
            const T2 & y )  [inline]
```

Self min()

6.2.2.9 moveToCMCoord()

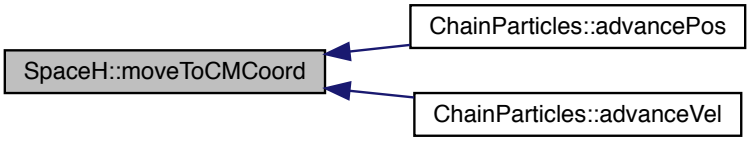
```
template<typename ScalarArray , typename VectorArray >
void SpaceH::moveToCMCoord (
            const ScalarArray & mass,
            VectorArray & phyVar )
```

Move variables to central mass coordinates.

Parameters

mass	Array of mass.
phyVar	Array of variables need to be moved.

Here is the caller graph for this function:



6.2.2.10 print()

```
template<typename T >
void SpaceH::print (
             T & var )
```

print an array. Used for debug

6.2.3 Variable Documentation

6.2.3.1 DYNAMICAL

```
constexpr size_t SpaceH::DYNAMICAL = 0
```

6.2.3.2 INV_C

```
constexpr double SpaceH::INV_C = 1 / C
```

6.2.3.3 INV_C2

```
constexpr double SpaceH::INV_C2 = INV_C * INV_C
```

6.2.3.4 INV_C3

```
constexpr double SpaceH::INV_C3 = INV_C2 * INV_C
```

6.2.3.5 INV_C4

```
constexpr double SpaceH::INV_C4 = INV_C3 * INV_C
```

6.2.3.6 INV_C5

```
constexpr double SpaceH::INV_C5 = INV_C4 * INV_C
```

6.3 SpaceH::chain Namespace Reference

Classes

- struct Node

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

Functions

- template<typename VectorArray , typename IndexArray >
  void getChainIndex (const VectorArray &pos, IndexArray &chainIndex)

  Calculate the mapping index from Cartesian coordinate to chain coordinate.
- template<typename VectorArray , typename NodeArray >
  void createAdjMartix (const VectorArray &pos, NodeArray &AdjMatrix)

  Create the adjoint matrix for particle pairs.
- template<typename NodeArray , typename IndexArray >
  void createChainIndex (NodeArray &AdjMatrix, IndexArray &chainIndex)

  Create mapping index from adjoint matrix.
- template<typename IndexArray >
  bool IsDiff (const IndexArray &Index1, const IndexArray &Index2)

  Check if two mapping indexes are the same.
- template<typename VectorArray , typename IndexArray >
  void updateChain (VectorArray &pos, IndexArray &chainIndex, IndexArray &newIndex)

  Update the position chain.
- template<typename VectorArray , typename IndexArray >
  void synChain (VectorArray &data, VectorArray &chainData, IndexArray &chainIndex)

  Calulate the chain data from Cartesian data and chain index mapping.
- template<typename VectorArray , typename IndexArray >
  void synCartesian (VectorArray &chainData, VectorArray &data, IndexArray &chainIndex)

  Calulate the Cartesian data from chain data and chain index mapping.

6.3.1 Function Documentation

6.3.1.1 createAdjMartix()

```
template<typename VectorArray , typename NodeArray >
void SpaceH::chain::createAdjMartix (
             const VectorArray & pos,
             NodeArray & AdjMatrix )
```

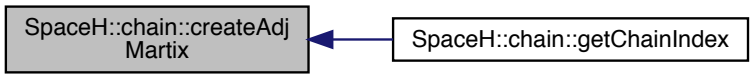
Create the adjoint matrix for particle pairs.

Create the adjoint matrix(distance of particle pairs organized by index-index matrix).

Parameters

pos	The array of particle position, used to calculate the distance of particle pairs.
AdjMatrix	The adjoint matrix needs to be calculated as a return value.

Here is the caller graph for this function:



6.3.1.2 createChainIndex()

```
template<typename NodeArray , typename IndexArray >
void SpaceH::chain::createChainIndex (
             NodeArray & AdjMatrix,
             IndexArray & chainIndex )
```

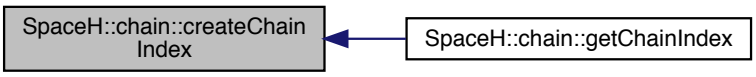
Create mapping index from adjoint matrix.

Create mapping index from sorted elements of adjoint matrix and connect them to a chain consequently.

Parameters

AdjMatrix	The adjoint matrix.
chainIndex	The maping index needs to be calculated as a return value.

Here is the caller graph for this function:



6.3.1.3 getChainIndex()

```
template<typename VectorArray , typename IndexArray >
void SpaceH::chain::getChainIndex (
            const VectorArray & pos,
            IndexArray & chainIndex )
```

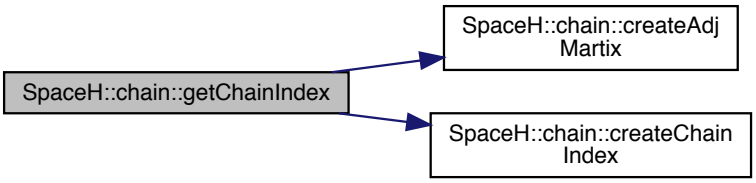
Calculate the mapping index from Cartesian coordinate to chain coordinate.

Find the mapping index from Cartesian coordinate to chain coordinate. The chain is formed by connecting the nearest particle pairs consequently.

Parameters

pos	The array of particle position, used to calculate the distance of particle pairs.
chainIndex	The maping index needs to be calculated as a return value.

Here is the call graph for this function:



6.3.1.4 IsDiff()

```
template<typename IndexArray >
bool SpaceH::chain::IsDiff (
```

```
            const IndexArray & Index1,
            const IndexArray & Index2 )
```

Check if two mapping indexes are the same.

Checking the identity of two chain index mappings.

Parameters

Index1	The first index array.
Index2	The second index array.

Returns

boolean

Note

[2,4,5,3,1] is identical to [1,3,5,4,2]

6.3.1.5 synCartesian()

```
template<typename VectorArray , typename IndexArray >
void SpaceH::chain::synCartesian (
            VectorArray & chainData,
            VectorArray & data,
            IndexArray & chainIndex )
```

Calulate the Cartesian data from chain data and chain index mapping.

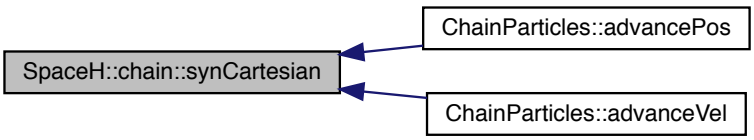
Parameters

chainData	Data in chain coordinates.
data	Data need to be calculated in Cartesian coordinates.
chainIndex	Chain index mapping.

Note

This function should be a inverse transformation of synChain().

Here is the caller graph for this function:



6.3.1.6 synChain()

```
template<typename VectorArray , typename IndexArray >
void SpaceH::chain::synChain (
             VectorArray & data,
             VectorArray & chainData,
             IndexArray & chainIndex )
```

Calulate the chain data from Cartesian data and chain index mapping.

Parameters

data	Data in Cartesian coordinates.
chainData	Data need to be calculated in chain coordinates.
chainIndex	Chain index mapping.

Note

This function should be a inverse transformation of synCartesian().

6.3.1.7 updateChain()

```
template<typename VectorArray , typename IndexArray >
void SpaceH::chain::updateChain (
             VectorArray & pos,
             IndexArray & chainIndex,
             IndexArray & newIndex )
```

Update the position chain.

Update the position chain. Due to the evolution, the chain index mapping could change with time, this function is used to update the position chain with old chain data.

Parameters

pos	The old chain position array needs update.
chainIndex	The old chain index mapping.
newIndex	The new chain index mapping.

Chapter 7

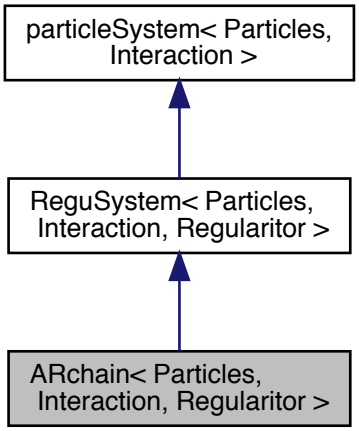
Class Documentation

7.1 ARchain< Particles, Interaction, Regularitor > Class Template Reference

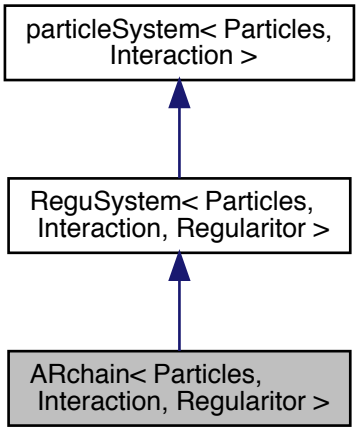
Algorithmatic Regularization chain System.

`#include <ARchain.h>`

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



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



Public Types

- using Base = particleSystem< Particles, Interaction >
- using Scalar = typename type::Scalar
- using Vector = typename type::Vector
- using VectorArray = typename type::VectorArray
- using type = typename Particles::type
- using ActiveScalarArray = typename Particles::ActiveScalarArray

Public Member Functions

- void kick (Scalar stepSize)

  Advance velocity one step with current acceleration.

Public Attributes

- Interaction act

  Interaction class.
- Particles partc

  Particle class.

Additional Inherited Members

7.1.1 Detailed Description

template<typename Particles, typename Interaction, typename Regularitor>
class ARchain< Particles, Interaction, Regularitor >

Algorithmatic Regularization chain System.

See details in https://arxiv.org/abs/0709.3367 .

7.1.2 Member Typedef Documentation

7.1.2.1 ActiveScalarArray

```
template<typename Particles , typename Interaction , typename Regularitor >
using particleSystem< Particles, Interaction >::ActiveScalarArray = typename Particles::↩
ActiveScalarArray
```

7.1.2.2 Base

```
template<typename Particles , typename Interaction , typename Regularitor >
using ARchain< Particles, Interaction, Regularitor >::Base = particleSystem<Particles, Interaction>
```

7.1.2.3 Scalar

```
template<typename Particles , typename Interaction , typename Regularitor >
using ARchain< Particles, Interaction, Regularitor >::Scalar = typename type::Scalar
```

7.1.2.4 type

```
template<typename Particles , typename Interaction , typename Regularitor >
using particleSystem< Particles, Interaction >::type = typename Particles::type
```

7.1.2.5 Vector

```
template<typename Particles , typename Interaction , typename Regularitor >
using ARchain< Particles, Interaction, Regularitor >::Vector = typename type::Vector
```

7.1.2.6 VectorArray

```
template<typename Particles , typename Interaction , typename Regularitor >
using ARchain< Particles, Interaction, Regularitor >::VectorArray = typename type::VectorArray
```

7.1.3 Member Function Documentation

7.1.3.1 kick()

```
template<typename Particles , typename Interaction , typename Regularitor >
void ARchain< Particles, Interaction, Regularitor >::kick (
            Scalar stepSize ) [inline]
```

Advance velocity one step with current acceleration.

Advance velocity array one step with current integration step size and accelerations.

Parameters

stepSize	Integration step size, will be transfered to physical time in the function.

7.1.4 Member Data Documentation

7.1.4.1 act

```
template<typename Particles , typename Interaction , typename Regularitor >
Interaction particleSystem< Particles, Interaction >::act
```

Interaction class.

7.1.4.2 partc

```
template<typename Particles , typename Interaction , typename Regularitor >
Particles particleSystem< Particles, Interaction >::partc
```

Particle class.

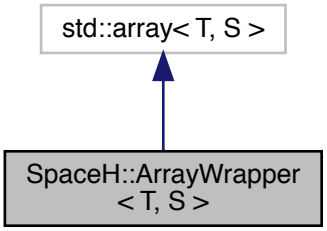
The documentation for this class was generated from the following file:

- particleSystem/ARChain/ARchain.h

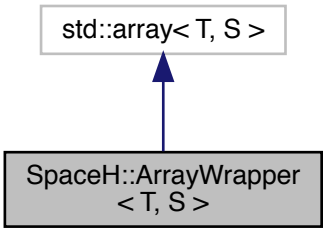
7.2 SpaceH::ArrayWrapper< T, S > Struct Template Reference

```
#include <protoType.h>
```

Inheritance diagram for SpaceH::ArrayWrapper< T, S >:



Collaboration diagram for SpaceH::ArrayWrapper< T, S >:



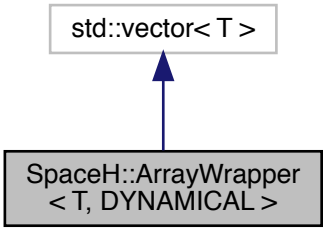
The documentation for this struct was generated from the following file:

- protoType.h

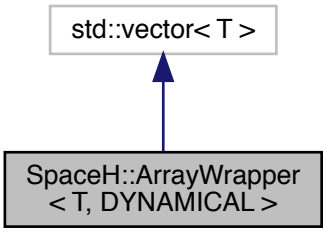
7.3 SpaceH::ArrayWrapper< T, DYNAMICAL > Struct Template Reference

```
#include <protoType.h>
```

Inheritance diagram for SpaceH::ArrayWrapper< T, DYNAMICAL >:



Collaboration diagram for SpaceH::ArrayWrapper< T, DYNAMICAL >:



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

- protoType.h

7.4 BSIterator< ParticSys, Integrator > Class Template Reference

Bulirsch-Stoer extrapolation algorithm.

`#include <BSIterator.h>`

Public Types

- using type = typename ParticSys::type
- using Scalar = typename type::Scalar
- using ActiveScalarArray = typename ParticSys::ActiveScalarArray
- template<typename T , size_t S>
  using Container = typename type::template Container< T, S >

Public Member Functions

- BSIterator ()

  Constructor for initializing cost, nSteps, fmin and CC.
- Scalar iterate (ParticSys &particles, Scalar stepLength)

  Interface of ODE iterator.
- void setRelativeError (Scalar relError)

  Set the local relative error.
- void setAbsoluteError (Scalar absError)

  Set the local absolute error.

7.4.1 Detailed Description

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

Bulirsch-Stoer extrapolation algorithm.

7.4.2 Member Typedef Documentation

7.4.2.1 ActiveScalarArray

```
template<typename ParticSys , typename Integrator >
using BSIterator< ParticSys, Integrator >::ActiveScalarArray = typename ParticSys::Active↩
ScalarArray
```

7.4.2.2 Container

```
template<typename ParticSys , typename Integrator >
template<typename T , size_t S>
using BSIterator< ParticSys, Integrator >::Container = typename type::template Container<T,
S>
```

7.4.2.3 Scalar

```
template<typename ParticSys , typename Integrator >
using BSIterator< ParticSys, Integrator >::Scalar = typename type::Scalar
```

7.4.2.4 type

```
template<typename ParticSys , typename Integrator >
using BSIterator< ParticSys, Integrator >::type = typename ParticSys::type
```

7.4.3 Constructor & Destructor Documentation

7.4.3.1 BSIterator()

```
template<typename ParticSys , typename Integrator >
BSIterator< ParticSys, Integrator >::BSIterator ( )
```

Constructor for initializing cost, nSteps, fmin and CC.

7.4.4 Member Function Documentation

7.4.4.1 iterate()

```
template<typename ParticSys , typename Integrator >
ParticSys::type::Scalar BSIterator< ParticSys, Integrator >::iterate (
            ParticSys & particles,
            Scalar stepLength )
```

Interface of ODE iterator.

Note

BSiterator will force use the internal mid-point integrator as the basic integrator.

Parameters

particles	Particle system need iteration.
integrator	Basica integrator used to evolve, but here BS iterator will force use internal mid-point integrator.
stepLength	Macro integration step length.

Returns

The next macro integration step length.

7.4.4.2 setAbsoluteError()

```
template<typename ParticSys , typename Integrator >
void BSIterator< ParticSys, Integrator >::setAbsoluteError (
            Scalar absError )  [inline]
```

Set the local absolute error.

7.4.4.3 setRelativeError()

```
template<typename ParticSys , typename Integrator >
void BSIterator< ParticSys, Integrator >::setRelativeError (
            Scalar relError )  [inline]
```

Set the local relative error.

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

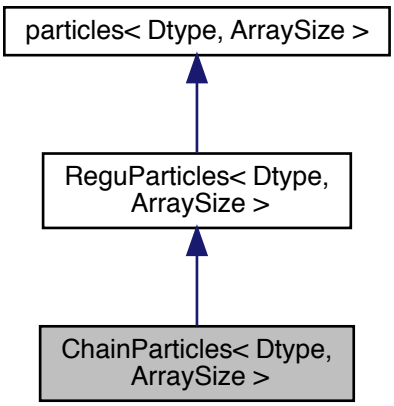
- ODEiterator/BSIterator.h

7.5 ChainParticles< Dtype, ArraySize > Class Template Reference

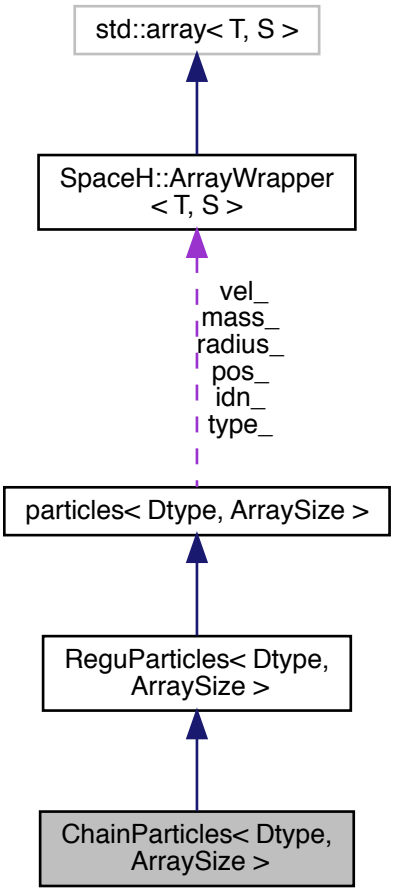
Class of dynamical variable.

```
#include <dynamicChain.h>
```

Inheritance diagram for ChainParticles< Dtype, ArraySize >:



Collaboration diagram for ChainParticles< Dtype, ArraySize >:



Public Types

- using Base = ReguParticles< Dtype, ArraySize >
- using Scalar = typename type::Scalar
- using Vector = typename type::Vector
- using VectorArray = typename type::VectorArray
- using IndexArray = typename type::IndexArray
- using ActiveScalarArray = typename Base::ActiveScalarArray

Public Member Functions

- const VectorArray & chainPos () const

  Position array const interface. Reference to pos_.
- const VectorArray & chainVel () const

  Velocity array const interface. Reference to vel_.
- const IndexArray & chainIndex () const

Index array const interface. Reference to ch_index_.

- const Vector & chainPos (size_t i) const

  Position vector const interface. Reference to pos_[i].
- const Vector & chainVel (size_t i) const

  Velocity vecotr const interface. Reference to vel_[i].
- const size_t chainIndex (size_t i) const

  Index const interface. Reference to ch_index_[i].
- void advancePos (Scalar stepSize)

  Advance the position array with internal velocity array.
- void advanceVel (const VectorArray &acc, Scalar stepSize)

  Advance the velocity array with given acceleration array.

Static Public Attributes

- static constexpr SpaceH::DATASTRUCT dataStruct {SpaceH::DATASTRUCT::CHAIN}

Friends

- std::istream & operator>> (std::istream &is, ChainParticles &partc)

  Input(Initialize) variables with istream.
- ActiveScalarArray & operator>> (ActiveScalarArray &data, ChainParticles &partc)

  Input variables with plain scalar array.
- ActiveScalarArray & operator<< (ActiveScalarArray &data, const ChainParticles &partc)

  Output variables to plain scalar array.

Additional Inherited Members

7.5.1 Detailed Description

template<typename Dtype, size_t ArraySize>
class ChainParticles< Dtype, ArraySize >

Class of dynamical variable.

7.5.2 Member Typedef Documentation

7.5.2.1 ActiveScalarArray

```
template<typename Dtype , size_t ArraySize>
using ChainParticles< Dtype, ArraySize >::ActiveScalarArray = typename Base::ActiveScalarArray
```

7.5.2.2 Base

```
template<typename Dtype , size_t ArraySize>
using ChainParticles< Dtype, ArraySize >::Base = ReguParticles<Dtype, ArraySize>
```

7.5.2.3 IndexArray

```
template<typename Dtype , size_t ArraySize>
using ChainParticles< Dtype, ArraySize >::IndexArray = typename type::IndexArray
```

7.5.2.4 Scalar

```
template<typename Dtype , size_t ArraySize>
using ChainParticles< Dtype, ArraySize >::Scalar = typename type::Scalar
```

7.5.2.5 Vector

```
template<typename Dtype , size_t ArraySize>
using ChainParticles< Dtype, ArraySize >::Vector = typename type::Vector
```

7.5.2.6 VectorArray

```
template<typename Dtype , size_t ArraySize>
using ChainParticles< Dtype, ArraySize >::VectorArray = typename type::VectorArray
```

7.5.3 Member Function Documentation

7.5.3.1 advancePos()

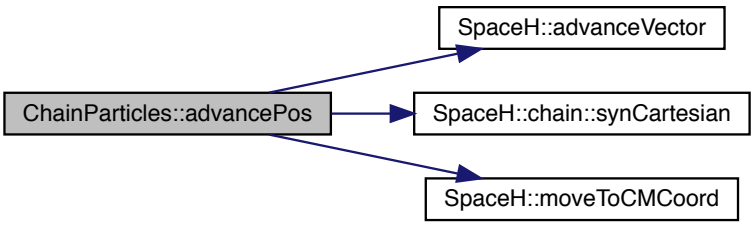
```
template<typename Dtype , size_t ArraySize>
void ChainParticles< Dtype, ArraySize >::advancePos (
            Scalar stepSize )  [inline]
```

Advance the position array with internal velocity array.

Parameters

stepSize	The advance step size.

Here is the call graph for this function:



7.5.3.2 advanceVel()

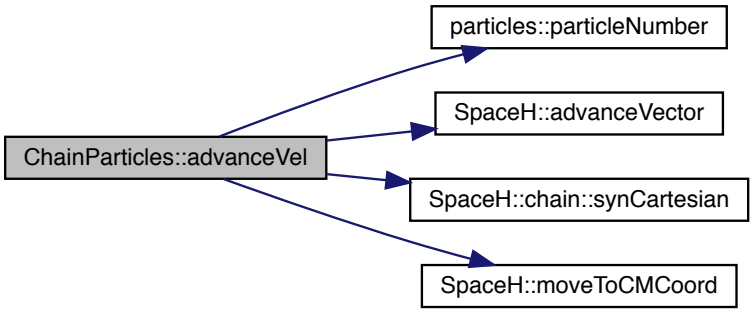
```
template<typename Dtype , size_t ArraySize>
void ChainParticles< Dtype, ArraySize >::advanceVel (
            const VectorArray & acc,
            Scalar stepSize )  [inline]
```

Advance the velocity array with given acceleration array.

Parameters

stepSize	The advance step size.
acc	The acceleration array.

Here is the call graph for this function:



7.5.3.3 chainIndex() [1/2]

```
template<typename Dtype , size_t ArraySize>
const IndexArray& ChainParticles< Dtype, ArraySize >::chainIndex ( ) const  [inline]
```

Index array const interface. Reference to ch_index_.

7.5.3.4 chainIndex() [2/2]

```
template<typename Dtype , size_t ArraySize>
const size_t ChainParticles< Dtype, ArraySize >::chainIndex (
            size_t i ) const  [inline]
```

Index const interface. Reference to ch_index_[i].

7.5.3.5 chainPos() [1/2]

```
template<typename Dtype , size_t ArraySize>
const VectorArray& ChainParticles< Dtype, ArraySize >::chainPos ( ) const  [inline]
```

Position array const interface. Reference to pos_.

7.5.3.6 chainPos() [2/2]

```
template<typename Dtype , size_t ArraySize>
const Vector& ChainParticles< Dtype, ArraySize >::chainPos (
            size_t i ) const  [inline]
```

Position vector const interface. Reference to pos_[i].

7.5.3.7 chainVel() [1/2]

```
template<typename Dtype , size_t ArraySize>
const VectorArray& ChainParticles< Dtype, ArraySize >::chainVel ( ) const  [inline]
```

Velocity array const interface. Reference to vel_.

7.5.3.8 chainVel() [2/2]

```
template<typename Dtype , size_t ArraySize>
const Vector& ChainParticles< Dtype, ArraySize >::chainVel (
            size_t i ) const  [inline]
```

Velocity vecotr const interface. Reference to vel_[i].

7.5.4 Friends And Related Function Documentation

7.5.4.1 operator<<

```
template<typename Dtype , size_t ArraySize>
ActiveScalarArray& operator<< (
            ActiveScalarArray & data,
            const ChainParticles< Dtype, ArraySize > & partc )  [friend]
```

Output variables to plain scalar array.

7.5.4.2 operator>> [1/2]

```
template<typename Dtype , size_t ArraySize>
std::istream& operator>> (
            std::istream & is,
            ChainParticles< Dtype, ArraySize > & partc )  [friend]
```

Input(Initialize) variables with istream.

7.5.4.3 operator>> [2/2]

```
template<typename Dtype , size_t ArraySize>
ActiveScalarArray& operator>> (
            ActiveScalarArray & data,
            ChainParticles< Dtype, ArraySize > & partc )  [friend]
```

Input variables with plain scalar array.

7.5.5 Member Data Documentation

7.5.5.1 dataStruct

```
template<typename Dtype , size_t ArraySize>
constexpr SpaceH::DATASTRUCT ChainParticles< Dtype, ArraySize >::dataStruct {SpaceH::DATASTRUCT::CHAIN}
[static]
```

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

- particleSystem/ARChain/dynamicChain.h

7.6 constIterator< ParticSys, Integrator > Class Template Reference

Most common iterator.

`#include <constIterator.h>`

Public Types

- using type = typename ParticSys::type
- using Scalar = typename type::Scalar

Public Member Functions

- Scalar iterate (ParticSys &particles, Scalar stepLength)

  interface to iterate particle system for one step

7.6.1 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.

7.6.2 Member Typedef Documentation

7.6.2.1 Scalar

```
template<typename ParticSys , typename Integrator >
using constIterator< ParticSys, Integrator >::Scalar = typename type::Scalar
```

7.6.2.2 type

```
template<typename ParticSys , typename Integrator >
using constIterator< ParticSys, Integrator >::type = typename ParticSys::type
```

7.6.3 Member Function Documentation

7.6.3.1 iterate()

```
template<typename ParticSys , typename Integrator >
Scalar constIterator< ParticSys, Integrator >::iterate (
            ParticSys & particles,
            Scalar stepLength )  [inline]
```

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

7.7 dicholterator< ParticSys, Integrator > Class Template Reference

Dichotomy iterator.

```
#include <dichotomy.h>
```

Public Types

- typedef ParticSys::Scalar Scalar
- typedef ParticSys::PlainArray PlainArray

Public Member Functions

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

  interface to iterate particle system for one step
- void setRelativeError (Scalar relError)

  Set the local relative error.
- void setAbsoluteError (Scalar absError)

  Set the local absolute error.

7.7.1 Detailed Description

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

Dichotomy iterator.

Dichotomy iterator, use dichotomy to adjust the step size.

7.7.2 Member Typedef Documentation

7.7.2.1 PlainArray

```
template<typename ParticSys , typename Integrator >
typedef ParticSys::PlainArray dichoIterator< ParticSys, Integrator >::PlainArray
```

7.7.2.2 Scalar

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

7.7.3 Member Function Documentation

7.7.3.1 iterate()

```
template<typename ParticSys , typename Integrator >
ParticSys::Scalar dichoIterator< ParticSys, Integrator >::iterate (
            ParticSys & particles,
            Integrator & integrator,
            Scalar stepLength )
```

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.

7.7.3.2 setAbsoluteError()

```
template<typename ParticSys , typename Integrator >
void dichoIterator< ParticSys, Integrator >::setAbsoluteError (
            Scalar absError )  [inline]
```

Set the local absolute error.

7.7.3.3 setRelativeError()

```
template<typename ParticSys , typename Integrator >
void dichoIterator< ParticSys, Integrator >::setRelativeError (
            Scalar relError )  [inline]
```

Set the local relative error.

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

- ODEiterator/dichotomy.h

7.8 dtype< T > Struct Template Reference

Public Types

- using type = decltype(check< T >(0))

Static Public Member Functions

- template<typename U >
  static U::value_type check (typename U::value_type)
- template<typename U >
  static U check (U)

7.8.1 Member Typedef Documentation

7.8.1.1 type

```
template<typename T >
using dtype< T >::type = decltype(check<T>(0))
```

7.8.2 Member Function Documentation

7.8.2.1 check() [1/2]

```
template<typename T >
template<typename U >
static U::value_type dtype< T >::check (
            typename U::value_type  )  [static]
```

7.8.2.2 check() [2/2]

```
template<typename T >
template<typename U >
static U dtype< T >::check (
            U  )  [static]
```

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

- test/typetest.cpp

7.9 dynamicSystem< ParticSys, ODEiterator > Class Template Reference

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

```
#include <dynamicSystem.h>
```

Public Types

- using type = typename ParticSys::type
- using Scalar = typename type::Scalar

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 (Scalar)

  Set the step length.
- virtual ~dynamicSystem ()

  Default destructor, virtualize for inherent class.

Public Attributes

- Scalar stepLength {0.0}

  Macro step size for ODE iterator.
- int steps {0}

  Steps.
- ParticSys particles

  Particle system.
- ODEiterator iterator

  ODE Iterator.

7.9.1 Detailed Description

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

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

7.9.2 Member Typedef Documentation

7.9.2.1 Scalar

```
template<typename ParticSys , typename ODEiterator >
using dynamicSystem< ParticSys, ODEiterator >::Scalar = typename type::Scalar
```

7.9.2.2 type

```
template<typename ParticSys , typename ODEiterator >
using dynamicSystem< ParticSys, ODEiterator >::type = typename ParticSys::type
```

7.9.3 Constructor & Destructor Documentation

7.9.3.1 ~dynamicSystem()

```
template<typename ParticSys , typename ODEiterator >
virtual dynamicSystem< ParticSys, ODEiterator >::~dynamicSystem ( )  [inline], [virtual]
```

Default destructor, virtualize for inherent class.

7.9.4 Member Function Documentation

7.9.4.1 advanceOneStep()

```
template<typename ParticSys , typename ODEiterator >
void dynamicSystem< ParticSys, 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.

7.9.4.2 loadText()

```
template<typename ParticSys , typename ODEiterator >
void dynamicSystem< ParticSys, 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.

Here is the call graph for this function:

Here is the caller graph for this function:

7.9.4.3 setStepLength()

```
template<typename ParticSys , typename ODEiterator >
void dynamicSystem< ParticSys, ODEiterator >::setStepLength (
            Scalar stepSize )
```

Set the step length.

7.9.5 Member Data Documentation

7.9.5.1 iterator

```
template<typename ParticSys , typename ODEiterator >
ODEiterator dynamicSystem< ParticSys, ODEiterator >::iterator
```

ODE Iterator.

7.9.5.2 particles

```
template<typename ParticSys , typename ODEiterator >
ParticSys dynamicSystem< ParticSys, ODEiterator >::particles
```

Particle system.

7.9.5.3 stepLength

```
template<typename ParticSys , typename ODEiterator >
Scalar dynamicSystem< ParticSys, ODEiterator >::stepLength {0.0}
```

Macro step size for ODE iterator.

7.9.5.4 steps

```
template<typename ParticSys , typename ODEiterator >
int dynamicSystem< ParticSys, ODEiterator >::steps {0}
```

Steps.

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

- dynamicSystem.h

7.10 SpaceH::EmptyForce< Dtype, ArraySize > Struct Template Reference

```
#include <forces.h>
```

Public Types

- using type = SpaceH::ProtoType< Dtype, ArraySize >
- using Scalar = typename type::Scalar
- using Vector = typename type::Vector
- using VectorArray = typename type::VectorArray
- using ScalarArray = typename type::ScalarArray
- using IndexArray = typename type::IndexArray

Public Member Functions

- void calcuAcc (const ScalarArray &mass, const VectorArray &pos)
- void calcuAcc (const ScalarArray &mass, const VectorArray &pos, const VectorArray &vel)
- void calcuAcc (const ScalarArray &mass, const VectorArray &pos, const VectorArray &chainPos, const IndexArray &chainIndex)
- void calcuAcc (const ScalarArray &mass, const VectorArray &pos, const VectorArray &vel, const VectorArray &chainPos, const VectorArray &chainVel, const IndexArray &chainIndex)
- void addTotal (VectorArray &acc)
- const VectorArray & acc ()
- const Vector & acc (size_t i)

Static Public Attributes

- static constexpr bool isVelDep {false}

7.10.1 Member Typedef Documentation

7.10.1.1 IndexArray

```
template<typename Dtype , size_t ArraySize>
using SpaceH::EmptyForce< Dtype, ArraySize >::IndexArray = typename type::IndexArray
```

7.10.1.2 Scalar

```
template<typename Dtype , size_t ArraySize>
using SpaceH::EmptyForce< Dtype, ArraySize >::Scalar = typename type::Scalar
```

7.10.1.3 ScalarArray

```
template<typename Dtype , size_t ArraySize>
using SpaceH::EmptyForce< Dtype, ArraySize >::ScalarArray = typename type::ScalarArray
```

7.10.1.4 type

```
template<typename Dtype , size_t ArraySize>
using SpaceH::EmptyForce< Dtype, ArraySize >::type = SpaceH::ProtoType<Dtype, ArraySize>
```

7.10.1.5 Vector

```
template<typename Dtype , size_t ArraySize>
using SpaceH::EmptyForce< Dtype, ArraySize >::Vector = typename type::Vector
```

7.10.1.6 VectorArray

```
template<typename Dtype , size_t ArraySize>
using SpaceH::EmptyForce< Dtype, ArraySize >::VectorArray = typename type::VectorArray
```

7.10.2 Member Function Documentation

7.10.2.1 acc() [1/2]

```
template<typename Dtype , size_t ArraySize>
const VectorArray& SpaceH::EmptyForce< Dtype, ArraySize >::acc ( )  [inline]
```

7.10.2.2 acc() [2/2]

```
template<typename Dtype , size_t ArraySize>
const Vector& SpaceH::EmptyForce< Dtype, ArraySize >::acc (
            size_t i )  [inline]
```

7.10.2.3 addTotal()

```
template<typename Dtype , size_t ArraySize>
void SpaceH::EmptyForce< Dtype, ArraySize >::addTotal (
            VectorArray & acc )  [inline]
```

7.10.2.4 calcuAcc() [1/4]

```
template<typename Dtype , size_t ArraySize>
void SpaceH::EmptyForce< Dtype, ArraySize >::calcuAcc (
            const ScalarArray & mass,
            const VectorArray & pos )  [inline]
```

7.10.2.5 calcuAcc() [2/4]

```
template<typename Dtype , size_t ArraySize>
void SpaceH::EmptyForce< Dtype, ArraySize >::calcuAcc (
             const ScalarArray & mass,
             const VectorArray & pos,
             const VectorArray & vel )  [inline]
```

7.10.2.6 calcuAcc() [3/4]

```
template<typename Dtype , size_t ArraySize>
void SpaceH::EmptyForce< Dtype, ArraySize >::calcuAcc (
             const ScalarArray & mass,
             const VectorArray & pos,
             const VectorArray & chainPos,
             const IndexArray & chainIndex )  [inline]
```

7.10.2.7 calcuAcc() [4/4]

```
template<typename Dtype , size_t ArraySize>
void SpaceH::EmptyForce< Dtype, ArraySize >::calcuAcc (
             const ScalarArray & mass,
             const VectorArray & pos,
             const VectorArray & vel,
             const VectorArray & chainPos,
             const VectorArray & chainVel,
             const IndexArray & chainIndex )  [inline]
```

7.10.3 Member Data Documentation

7.10.3.1 isVelDep

```
template<typename Dtype , size_t ArraySize>
constexpr bool SpaceH::EmptyForce< Dtype, ArraySize >::isVelDep {false}  [static]
```

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

- interaction/forces.h

7.11 errhand Class Reference

```
#include <errhand.h>
```

Public Member Functions

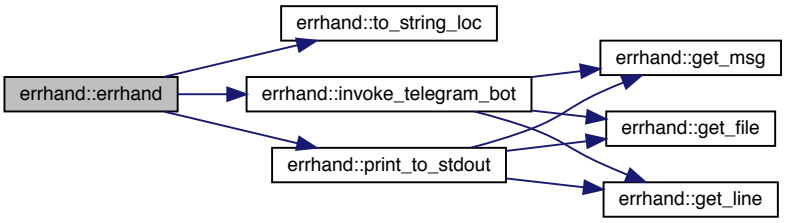
- errhand (std::string err_msg_input, const char *file_input, size_t line_input)
- std::string get_msg () const
- std::string get_file () const
- size_t get_line () const
- std::string to_string_loc (const char *obj)
- void invoke_telegram_bot ()
- void print_to_stdout ()

7.11.1 Constructor & Destructor Documentation

7.11.1.1 errhand()

```
errhand::errhand (
            std::string err_msg_input,
            const char * file_input,
            size_t line_input )  [inline]
```

Here is the call graph for this function:

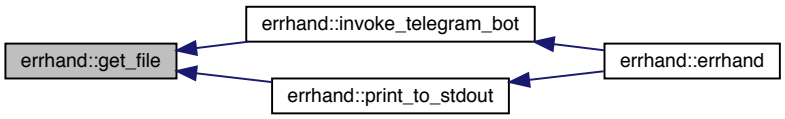


7.11.2 Member Function Documentation

7.11.2.1 get_file()

```
std::string errhand::get_file ( ) const  [inline]
```

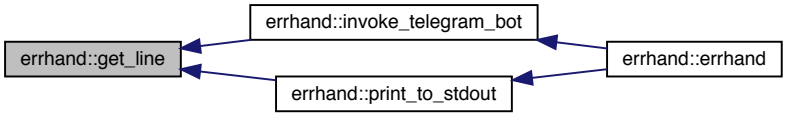
Here is the caller graph for this function:



7.11.2.2 get_line()

```
size_t errhand::get_line ( ) const  [inline]
```

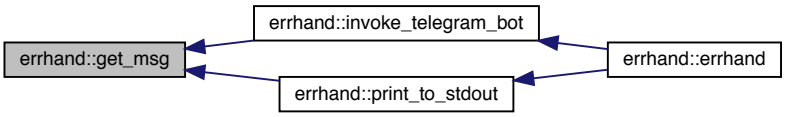
Here is the caller graph for this function:



7.11.2.3 get_msg()

```
std::string errhand::get_msg ( ) const  [inline]
```

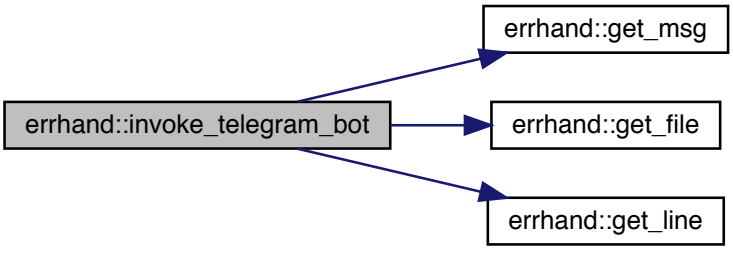
Here is the caller graph for this function:



7.11.2.4 invoke_telegram_bot()

```
void errhand::invoke_telegram_bot ( )  [inline]
```

Here is the call graph for this function:

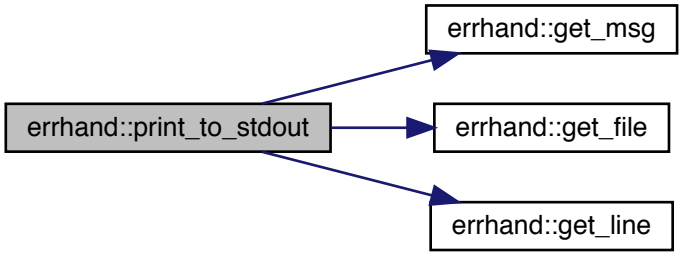


Here is the caller graph for this function:

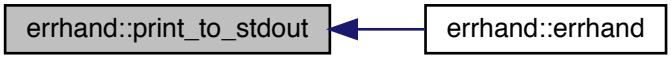
7.11.2.5 print_to_stdout()

```
void errhand::print_to_stdout ( )  [inline]
```

Here is the call graph for this function:



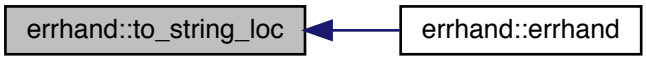
Here is the caller graph for this function:



7.11.2.6 to_string_loc()

```
std::string errhand::to_string_loc (
            const char * obj )  [inline]
```

Here is the caller graph for this function:



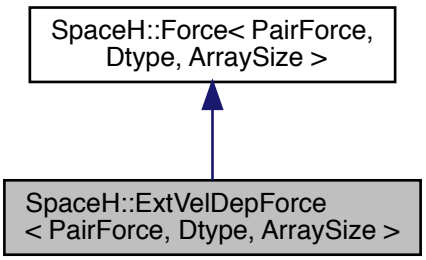
The documentation for this class was generated from the following file:

- errhand.h

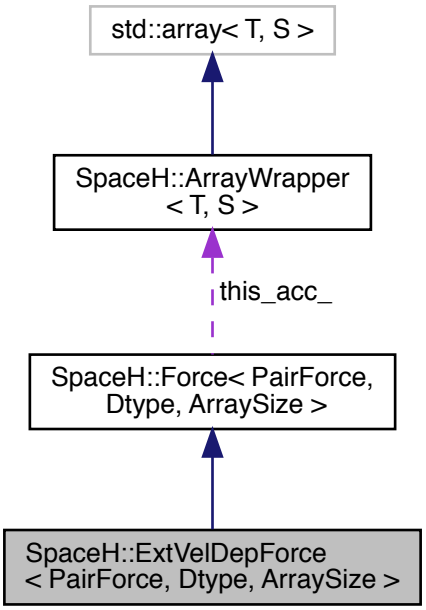
7.12 SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize > Struct Template Reference

```
#include <forces.h>
```

Inheritance diagram for SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize >:



Collaboration diagram for SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize >:



Public Types

- using Base = Force< PairForce, Dtype, ArraySize >
- using type = typename Base::type
- using ScalarArray = typename type::ScalarArray
- using VectorArray = typename type::VectorArray

Public Member Functions

- void calcuAcc (const ScalarArray &mass, const VectorArray &pos, const VectorArray &vel)

Public Attributes

- PairForce force_
- VectorArray this_acc_

Static Public Attributes

- static constexpr bool isVelDep {true}

Additional Inherited Members

7.12.1 Member Typedef Documentation

7.12.1.1 Base

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize >::Base = Force<PairForce, Dtype,
ArraySize>
```

7.12.1.2 ScalarArray

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize >::ScalarArray = typename type::ScalarArray
```

7.12.1.3 type

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize >::type = typename Base::type
```

7.12.1.4 VectorArray

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize >::VectorArray = typename type::VectorArray
```

7.12.2 Member Function Documentation

7.12.2.1 calcuAcc()

```
template<typename PairForce , typename Dtype , size_t ArraySize>
void SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize >::calcuAcc (
            const ScalarArray & mass,
            const VectorArray & pos,
            const VectorArray & vel )  [inline]
```

7.12.3 Member Data Documentation

7.12.3.1 force_

```
template<typename PairForce , typename Dtype , size_t ArraySize>
PairForce SpaceH::Force< PairForce, Dtype, ArraySize >::force_
```

7.12.3.2 isVelDep

```
template<typename PairForce , typename Dtype , size_t ArraySize>
constexpr bool SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize >::isVelDep {true}  [static]
```

7.12.3.3 this_acc_

```
template<typename PairForce , typename Dtype , size_t ArraySize>
VectorArray SpaceH::Force< PairForce, Dtype, ArraySize >::this_acc_
```

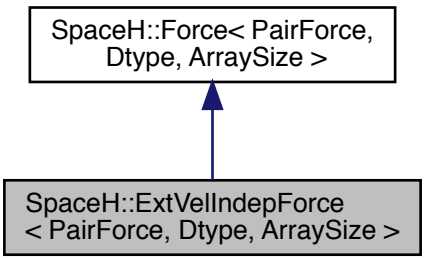
The documentation for this struct was generated from the following file:

- interaction/forces.h

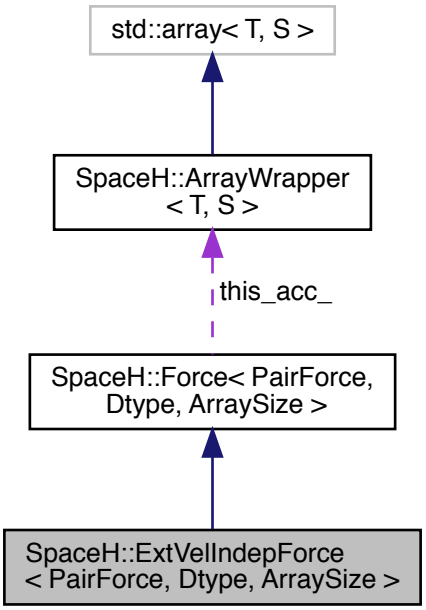
7.13 SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize > Struct Template Reference

```
#include <forces.h>
```

Inheritance diagram for SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize >:



Collaboration diagram for SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize >:



Public Types

- using Base = Force< PairForce, Dtype, ArraySize >
- using type = typename Base::type
- using ScalarArray = typename type::ScalarArray
- using VectorArray = typename type::VectorArray

Public Member Functions

- void calcuAcc (const ScalarArray &mass, const VectorArray &pos)

Public Attributes

- PairForce force_
- VectorArray this_acc_

Static Public Attributes

- static constexpr bool isVelDep {false}

Additional Inherited Members

7.13.1 Member Typedef Documentation

7.13.1.1 Base

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize >::Base = Force<PairForce, Dtype,
ArraySize>
```

7.13.1.2 ScalarArray

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize >::ScalarArray = typename type::ScalarArray
```

7.13.1.3 type

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize >::type = typename Base::type
```

7.13.1.4 VectorArray

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize >::VectorArray = typename type::VectorArray
```

7.13.2 Member Function Documentation

7.13.2.1 calcuAcc()

```
template<typename PairForce , typename Dtype , size_t ArraySize>
void SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize >::calcuAcc (
            const ScalarArray & mass,
            const VectorArray & pos )  [inline]
```

7.13.3 Member Data Documentation

7.13.3.1 force_

```
template<typename PairForce , typename Dtype , size_t ArraySize>
PairForce SpaceH::Force< PairForce, Dtype, ArraySize >::force_
```

7.13.3.2 isVelDep

```
template<typename PairForce , typename Dtype , size_t ArraySize>
constexpr bool SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize >::isVelDep {false}
[static]
```

7.13.3.3 this_acc_

```
template<typename PairForce , typename Dtype , size_t ArraySize>
VectorArray SpaceH::Force< PairForce, Dtype, ArraySize >::this_acc_
```

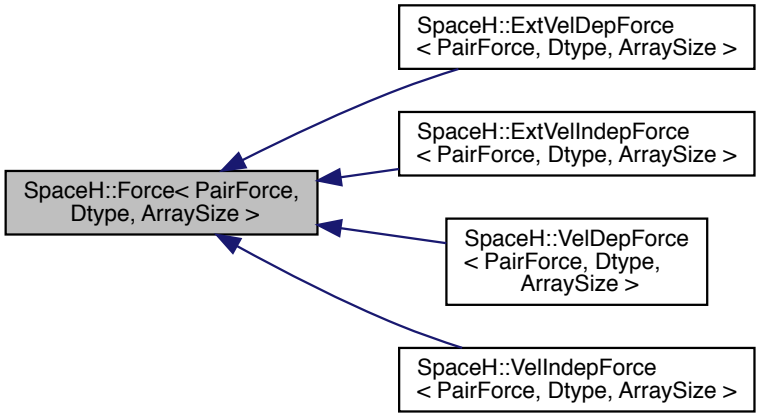
The documentation for this struct was generated from the following file:

- interaction/forces.h

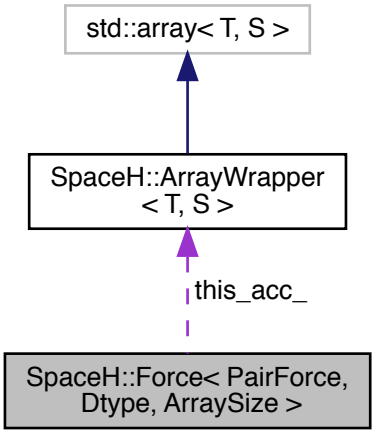
7.14 SpaceH::Force< PairForce, Dtype, ArraySize > Struct Template Reference

```
#include <forces.h>
```

Inheritance diagram for SpaceH::Force< PairForce, Dtype, ArraySize >:



Collaboration diagram for SpaceH::Force< PairForce, Dtype, ArraySize >:



Public Types

- using type = SpaceH::ProtoType< Dtype, ArraySize >
- using Vector = typename type::Vector
- using VectorArray = typename type::VectorArray

Public Member Functions

- void addTotal (VectorArray &acc)
- const VectorArray & acc ()
- const Vector & acc (size_t i)

Protected Attributes

- VectorArray this_acc_
- PairForce force_

7.14.1 Member Typedef Documentation

7.14.1.1 type

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::Force< PairForce, Dtype, ArraySize >::type = SpaceH::ProtoType<Dtype, Array↩
Size>
```

7.14.1.2 Vector

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::Force< PairForce, Dtype, ArraySize >::Vector = typename type::Vector
```

7.14.1.3 VectorArray

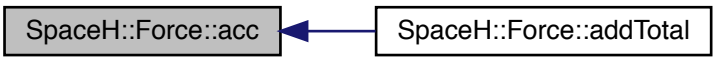
```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::Force< PairForce, Dtype, ArraySize >::VectorArray = typename type::VectorArray
```

7.14.2 Member Function Documentation

7.14.2.1 acc() [1/2]

```
template<typename PairForce , typename Dtype , size_t ArraySize>
const VectorArray& SpaceH::Force< PairForce, Dtype, ArraySize >::acc ( )  [inline]
```

Here is the caller graph for this function:



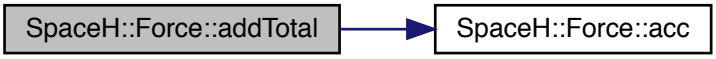
7.14.2.2 acc() [2/2]

```
template<typename PairForce , typename Dtype , size_t ArraySize>
const Vector& SpaceH::Force< PairForce, Dtype, ArraySize >::acc (
            size_t i )  [inline]
```

7.14.2.3 addTotal()

```
template<typename PairForce , typename Dtype , size_t ArraySize>
void SpaceH::Force< PairForce, Dtype, ArraySize >::addTotal (
            VectorArray & acc )  [inline]
```

Here is the call graph for this function:



7.14.3 Member Data Documentation

7.14.3.1 force_

```
template<typename PairForce , typename Dtype , size_t ArraySize>
PairForce SpaceH::Force< PairForce, Dtype, ArraySize >::force_  [protected]
```

7.14.3.2 this_acc_

```
template<typename PairForce , typename Dtype , size_t ArraySize>
VectorArray SpaceH::Force< PairForce, Dtype, ArraySize >::this_acc_  [protected]
```

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

- interaction/forces.h

7.15 SpaceH::get_value_type< T > Struct Template Reference

```
#include <protoType.h>
```

Public Types

- using type = decltype(check< T >(0))

7.15.1 Member Typedef Documentation

7.15.1.1 type

```
template<typename T >
using SpaceH::get_value_type< T >::type = decltype(check<T>(0))
```

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

- protoType.h

7.16 SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep > Class Template Reference

```
#include <interaction.h>
```

Public Types

- using type = typename VelIndep::type
- using Scalar = typename type::Scalar
- using Vector = typename type::Vector
- using VectorArray = typename type::VectorArray
- using ScalarArray = typename type::ScalarArray

Public Member Functions

- const VectorArray & totalAcc ()
- const VectorArray & velIndepAcc ()
- const VectorArray & velDepAcc ()
- const VectorArray & extVelIndepAcc ()
- const VectorArray & extVelDepAcc ()
- const Vector & totalAcc (size_t i)
- const Vector & velIndepAcc (size_t i)
- const Vector & velDepAcc (size_t i)
- const Vector & extVelIndepAcc (size_t i)
- const Vector & extVelDepAcc (size_t i)
- template<typename Particles >
  std::enable_if< Particles::dataStruct==SpaceH::DATASTRUCT::PLAIN >::type calcuVelIndepAcc (const Particles &partc)
- template<typename Particles >
  std::enable_if< Particles::dataStruct==SpaceH::DATASTRUCT::CHAIN >::type calcuVelIndepAcc (const Particles &partc)
- template<typename Particles >
  std::enable_if< Particles::dataStruct==SpaceH::DATASTRUCT::PLAIN >::type calcuVelDepAcc (const Particles &partc)

- template<typename Particles >
  std::enable_if< Particles::dataStruct==SpaceH::DATASTRUCT::CHAIN >::type calcuVelDepAcc (const Particles &partc)
- template<typename Particles >
  std::enable_if< Particles::dataStruct==SpaceH::DATASTRUCT::PLAIN >::type calcuAuxiVelDepAcc (const Particles &partc)
- template<typename Particles >
  std::enable_if< Particles::dataStruct==SpaceH::DATASTRUCT::CHAIN >::type calcuAuxiVelDepAcc (const Particles &partc)
- template<typename Particles >
  void calcuExtVelIndepAcc (const Particles &partc)
- template<typename Particles >
  void calcuExtVelDepAcc (const Particles &partc)
- template<typename Particles >
  void calcuExtAuxiVelDepAcc (const Particles &partc)
- void zeroTotalAcc ()
- void calcuTotalAcc ()

Static Public Attributes

- static constexpr bool isVelDep { VelDep::isVelDep || ExtVelDep::isVelDep }

7.16.1 Member Typedef Documentation

7.16.1.1 Scalar

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
using SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::Scalar = typename
type::Scalar
```

7.16.1.2 ScalarArray

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
using SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::ScalarArray = typename
type::ScalarArray
```

7.16.1.3 type

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
using SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::type = typename Vel↩
Indep::type
```

7.16.1.4 Vector

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
using SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::Vector = typename
type::Vector
```

7.16.1.5 VectorArray

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
using SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::VectorArray = typename
type::VectorArray
```

7.16.2 Member Function Documentation

7.16.2.1 calcuAuxiVelDepAcc() [1/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
template<typename Particles >
std::enable_if<Particles::dataStruct == SpaceH::DATASTRUCT::PLAIN>::type SpaceH::Interaction<
VelIndep, VelDep, ExtVelIndep, ExtVelDep >::calcuAuxiVelDepAcc (
            const Particles & partc )  [inline]
```

7.16.2.2 calcuAuxiVelDepAcc() [2/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
template<typename Particles >
std::enable_if<Particles::dataStruct == SpaceH::DATASTRUCT::CHAIN>::type SpaceH::Interaction<
VelIndep, VelDep, ExtVelIndep, ExtVelDep >::calcuAuxiVelDepAcc (
            const Particles & partc )  [inline]
```

7.16.2.3 calcuExtAuxiVelDepAcc()

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
template<typename Particles >
void SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::calcuExtAuxiVelDepAcc (
            const Particles & partc )  [inline]
```

7.16.2.4 calcuExtVelDepAcc()

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
template<typename Particles >
void SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::calcuExtVelDepAcc (
            const Particles & partc )  [inline]
```

7.16.2.5 calcuExtVelIndepAcc()

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
template<typename Particles >
void SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::calcuExtVelIndepAcc (
            const Particles & partc )  [inline]
```

7.16.2.6 calcuTotalAcc()

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
void SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::calcuTotalAcc ( )  [inline]
```

7.16.2.7 calcuVelDepAcc() [1/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
template<typename Particles >
std::enable_if<Particles::dataStruct == SpaceH::DATASTRUCT::PLAIN>::type SpaceH::Interaction<
VelIndep, VelDep, ExtVelIndep, ExtVelDep >::calcuVelDepAcc (
            const Particles & partc )  [inline]
```

7.16.2.8 calcuVelDepAcc() [2/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
template<typename Particles >
std::enable_if<Particles::dataStruct == SpaceH::DATASTRUCT::CHAIN>::type SpaceH::Interaction<
VelIndep, VelDep, ExtVelIndep, ExtVelDep >::calcuVelDepAcc (
            const Particles & partc )  [inline]
```

7.16.2.9 calcuVelIndepAcc() [1/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
template<typename Particles >
std::enable_if<Particles::dataStruct == SpaceH::DATASTRUCT::PLAIN>::type SpaceH::Interaction<
VelIndep, VelDep, ExtVelIndep, ExtVelDep >::calcuVelIndepAcc (
            const Particles & partc )  [inline]
```

7.16.2.10 calcuVelIndepAcc() [2/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
template<typename Particles >
std::enable_if<Particles::dataStruct == SpaceH::DATASTRUCT::CHAIN>::type SpaceH::Interaction<
VelIndep, VelDep, ExtVelIndep, ExtVelDep >::calcuVelIndepAcc (
            const Particles & partc )  [inline]
```

7.16.2.11 extVelDepAcc() [1/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
const VectorArray& SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::extVel↩
DepAcc ( )  [inline]
```

7.16.2.12 extVelDepAcc() [2/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
const Vector& SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::extVelDepAcc (
            size_t i )  [inline]
```

7.16.2.13 extVelIndepAcc() [1/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
const VectorArray& SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::extVel↩
IndepAcc ( )  [inline]
```

7.16.2.14 extVelIndepAcc() [2/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
const Vector& SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::extVelIndepAcc
(
            size_t i )  [inline]
```

7.16.2.15 totalAcc() [1/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
const VectorArray& SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::totalAcc (
) [inline]
```

7.16.2.16 totalAcc() [2/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
const Vector& SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::totalAcc (
            size_t i ) [inline]
```

7.16.2.17 velDepAcc() [1/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
const VectorArray& SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::velDepAcc
( ) [inline]
```

7.16.2.18 velDepAcc() [2/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
const Vector& SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::velDepAcc (
            size_t i ) [inline]
```

7.16.2.19 velIndepAcc() [1/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
const VectorArray& SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::velIndep↩
Acc ( ) [inline]
```

7.16.2.20 velIndepAcc() [2/2]

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
const Vector& SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::velIndepAcc (
            size_t i ) [inline]
```

7.16.2.21 zeroTotalAcc()

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
void SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::zeroTotalAcc ( )  [inline]
```

7.16.3 Member Data Documentation

7.16.3.1 isVelDep

```
template<typename VelIndep , typename VelDep , typename ExtVelIndep , typename ExtVelDep >
constexpr bool SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >::isVelDep {
VelDep::isVelDep || ExtVelDep::isVelDep }  [static]
```

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

- interaction/interaction.h

7.17 SpaceH::kahan< T > Struct Template Reference

Kahan number.

```
#include <kahanNumber.h>
```

Public Types

- using value_type = T

Public Member Functions

- kahan ()
- kahan (T r)
- kahan (const kahan &k)
- const kahan & operator= (const kahan &hs)
- operator T ()
- operator T () const
- void zeroErr ()

Public Attributes

- T real
- T err

Friends

- kahan operator- (const kahan &hs)
- const kahan & operator+= (kahan &lhs, const kahan &rhs)
- const kahan & operator-= (kahan &lhs, const kahan &rhs)
- const kahan & operator/= (kahan &lhs, const kahan &rhs)
- const kahan & operator*= (kahan &lhs, const kahan &rhs)
- std::ostream & operator<< (std::ostream &output, const kahan &v)

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

  Input from istream.

7.17.1 Detailed Description

template<typename T>
struct SpaceH::kahan< T >

Kahan number.

A way to reduce the round off error when adding a small number to a big one. See details in https://en.wikipedia.org/wiki/Kahan_summation_algorithm

7.17.2 Member Typedef Documentation

7.17.2.1 value_type

```
template<typename T >
using SpaceH::kahan< T >::value_type = T
```

7.17.3 Constructor & Destructor Documentation

7.17.3.1 kahan() [1/3]

```
template<typename T >
SpaceH::kahan< T >::kahan ( )  [inline]
```

7.17.3.2 kahan() [2/3]

```
template<typename T >
SpaceH::kahan< T >::kahan (
            T r )  [inline]
```

7.17.3.3 kahan() [3/3]

```
template<typename T >
SpaceH::kahan< T >::kahan (
            const kahan< T > & k )  [inline]
```

7.17.4 Member Function Documentation

7.17.4.1 operator T() [1/2]

```
template<typename T >
SpaceH::kahan< T >::operator T ( )  [inline]
```

7.17.4.2 operator T() [2/2]

```
template<typename T >
SpaceH::kahan< T >::operator T ( ) const  [inline]
```

7.17.4.3 operator=()

```
template<typename T >
const kahan& SpaceH::kahan< T >::operator= (
            const kahan< T > & hs )  [inline]
```

7.17.4.4 zeroErr()

```
template<typename T >
void SpaceH::kahan< T >::zeroErr ( )  [inline]
```

7.17.5 Friends And Related Function Documentation

7.17.5.1 operator*=

```
template<typename T >
const kahan& operator*= (
            kahan< T > & lhs,
            const kahan< T > & rhs )  [friend]
```

7.17.5.2 operator+=

```
template<typename T >
const kahan& operator+= (
            kahan< T > & lhs,
            const kahan< T > & rhs )  [friend]
```

7.17.5.3 operator-

```
template<typename T >
kahan operator- (
            const kahan< T > & hs )  [friend]
```

7.17.5.4 operator-=

```
template<typename T >
const kahan& operator-= (
            kahan< T > & lhs,
            const kahan< T > & rhs )  [friend]
```

7.17.5.5 operator/=

```
template<typename T >
const kahan& operator/= (
            kahan< T > & lhs,
            const kahan< T > & rhs )  [friend]
```

7.17.5.6 operator<<

```
template<typename T >
std::ostream& operator<< (
            std::ostream & output,
            const kahan< T > & v )  [friend]
```

Output to ostream.

7.17.5.7 operator>>

```
template<typename T >
std::istream& operator>> (
             std::istream & input,
             kahan< T > & v )  [friend]
```

Input from istream.

7.17.6 Member Data Documentation

7.17.6.1 err

```
template<typename T >
T SpaceH::kahan< T >::err
```

7.17.6.2 real

```
template<typename T >
T SpaceH::kahan< T >::real
```

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

- kahanNumber.h

7.18 logH< Particles > Class Template Reference

logH extention algorithmatic regularization interface

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

Public Types

- using type = typename Particles::type
- using Scalar = typename type::Scalar

Public Member Functions

- Scalar getPhysicalPosTime (Particles &partc, Scalar stepSize)

  Calculate the physical time for position advance from integration step size.
- Scalar getPhysicalVelTime (Particles &partc, Scalar stepSize)

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

7.18.1 Detailed Description

template<typename Particles>
class logH< Particles >

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 .

7.18.2 Member Typedef Documentation

7.18.2.1 Scalar

```
template<typename Particles >
using logH< Particles >::Scalar = typename type::Scalar
```

7.18.2.2 type

```
template<typename Particles >
using logH< Particles >::type = typename Particles::type
```

7.18.3 Member Function Documentation

7.18.3.1 getPhysicalPosTime()

```
template<typename Particles >
Scalar logH< Particles >::getPhysicalPosTime (
            Particles & partc,
            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.

Here is the call graph for this function:

7.18.3.2 getPhysicalVelTime()

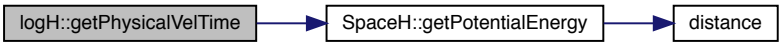
```
template<typename Particles >
Scalar logH< Particles >::getPhysicalVelTime (
            Particles & partc,
            Scalar stepSize )  [inline]
```

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.

Here is the call graph for this function:



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

- particleSystem/reguSystem/regularization.h

7.19 SpaceH::NewtonForce< Dtype, ArraySize > Struct Template Reference

```
#include <forces.h>
```

Public Types

- using type = SpaceH::ProtoType< Dtype, ArraySize >
- using Scalar = typename type::Scalar
- using Vector = typename type::Vector
- using VectorArray = typename type::VectorArray
- using ScalarArray = typename type::ScalarArray

Public Member Functions

- void operator() (Vector &acc1, Vector &acc2, const Scalar m1, const Scalar m2, const Vector &pos1, const Vector &pos2, const Vector &dr)

7.19.1 Member Typedef Documentation

7.19.1.1 Scalar

```
template<typename Dtype , size_t ArraySize>
using SpaceH::NewtonForce< Dtype, ArraySize >::Scalar = typename type::Scalar
```

7.19.1.2 ScalarArray

```
template<typename Dtype , size_t ArraySize>
using SpaceH::NewtonForce< Dtype, ArraySize >::ScalarArray = typename type::ScalarArray
```

7.19.1.3 type

```
template<typename Dtype , size_t ArraySize>
using SpaceH::NewtonForce< Dtype, ArraySize >::type = SpaceH::ProtoType<Dtype, ArraySize>
```

7.19.1.4 Vector

```
template<typename Dtype , size_t ArraySize>
using SpaceH::NewtonForce< Dtype, ArraySize >::Vector = typename type::Vector
```

7.19.1.5 VectorArray

```
template<typename Dtype , size_t ArraySize>
using SpaceH::NewtonForce< Dtype, ArraySize >::VectorArray = typename type::VectorArray
```

7.19.2 Member Function Documentation

7.19.2.1 operator()()

```
template<typename Dtype , size_t ArraySize>
void SpaceH::NewtonForce< Dtype, ArraySize >::operator() (
            Vector & acc1,
            Vector & acc2,
            const Scalar m1,
            const Scalar m2,
            const Vector & pos1,
            const Vector & pos2,
            const Vector & dr )  [inline]
```

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

- interaction/forces.h

7.20 SpaceH::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

7.20.1 Detailed Description

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

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

7.20.2 Member Data Documentation

7.20.2.1 available

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

State of node. If this node can be chained.

7.20.2.2 i

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

Particle index.

7.20.2.3 j

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

Particle index.

7.20.2.4 Rij

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

Relative distance of two particles.

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

- particleSystem/ARChain/chain.h

7.21 NoRegu< Particles > Class Template Reference

Ordinary algorithmatic regularization interface.

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

Public Types

- using type = typename Particles::type
- using Scalar = typename type::Scalar

Public Member Functions

- Scalar getPhysicalPosTime (Particles &partc, Scalar stepSize)

  Calculate the physical time for position advance from integration step size.
- Scalar getPhysicalVelTime (Particles &partc, Scalar stepSize)

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

7.21.1 Detailed Description

template<typename Particles>
class NoRegu< Particles >

Ordinary algorithmatic regularization interface.

No regularization.

7.21.2 Member Typedef Documentation

7.21.2.1 Scalar

```
template<typename Particles >
using NoRegu< Particles >::Scalar = typename type::Scalar
```

7.21.2.2 type

```
template<typename Particles >
using NoRegu< Particles >::type = typename Particles::type
```

7.21.3 Member Function Documentation

7.21.3.1 getPhysicalPosTime()

```
template<typename Particles >
Scalar NoRegu< Particles >::getPhysicalPosTime (
            Particles & partc,
            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.

7.21.3.2 getPhysicalVelTime()

```
template<typename Particles >
Scalar NoRegu< Particles >::getPhysicalVelTime (
            Particles & partc,
            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:

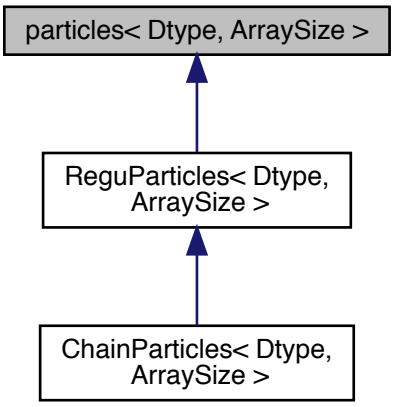
- particleSystem/reguSystem/regularization.h

7.22 particles< Dtype, ArraySize > Class Template Reference

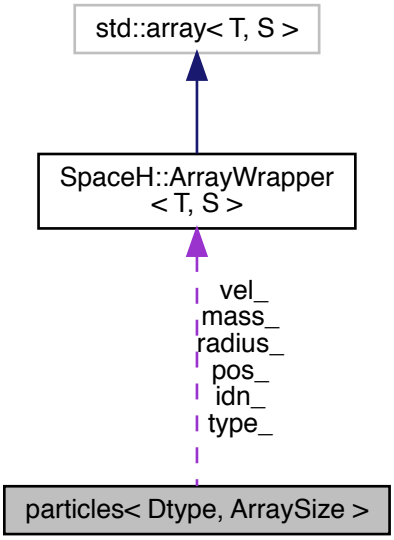
Class of dynamical variable.

```
#include <particles.h>
```

Inheritance diagram for particles< Dtype, ArraySize >:



Collaboration diagram for particles< Dtype, ArraySize >:



Public Types

- using type = SpaceH::ProtoType< Dtype, ArraySize >
- template<typename T , size_t S>
  using Container = typename type::template Container< T, S >

- using Scalar = typename type::Scalar
- using Vector = typename type::Vector
- using VectorArray = typename type::VectorArray
- using ScalarArray = typename type::ScalarArray
- using IntArray = typename type::IntArray
- using ActiveScalarArray = Container< Scalar, activeScalar >

Public Member Functions

- size_t particleNumber () const

  Get the number of the particles.
- const Scalar & time () const

  Physical time scalar const interface. Reference to state.time.
- const VectorArray & pos () const

  Position array const interface. Reference to state.pos.
- const VectorArray & vel () const

  Velocity array const interface. Reference to state.vel.
- const ScalarArray & mass () const

  Mass array const interface. Reference to attribute.mass.
- const ScalarArray & radius () const

  Radius array const interface. Reference to attribute.radius.
- const IntArray & kind () const

  Particle type array const interface. Reference to attribute.type.
- const IntArray & idn () const

  Particle id array const interface. Reference to attribute.type.
- const Vector & pos (size_t i) const

  Position vector const interface. Reference to state.pos[i].
- const Vector & vel (size_t i) const

  Velocity vecotr const interface. Reference to state.vel[i].
- const Scalar & mass (size_t i) const

  Mass const interface. Reference to attribute.mass[i].
- const Scalar & radius (size_t i) const

  Radius const interface. Reference to attribute.radius[i].
- const int & kind (size_t i) const

  Particle type const interface. Reference to attribute.type[i].
- const int & idn (size_t i) const

  Particle id const interface. Reference to attribute.type[i].
- void advanceTime (Scalar dt)

  Advance the time.
- void advancePos (Scalar stepSize)

  Advance the position array with internal velocity array.
- void advanceVel (const VectorArray &acc, Scalar stepSize)

  Advance the velocity array with given acceleration array.

Static Public Attributes

- static constexpr SpaceH::DATASTRUCT dataStruct {SpaceH::DATASTRUCT::PLAIN}
- static constexpr size_t activeScalar {6∗type::arraySize + 1}

Protected Attributes

- VectorArray pos_

  Position array of the particles. Element is 3D vector.
- VectorArray vel_

  Velocity array of the particles. Element is 3D vector.
- ScalarArray mass_

  Mass array of the particles. Element is Scalar.
- ScalarArray radius_

  Radius array of the particles. Element is Scalar.
- IntArray type_

  Type Array of the particles. Element is int.
- IntArray idn_

  Id Array of the particles. Element is int.
- Scalar time_

  The physical time of the dynamic system.
- Scalar totalMass_

  The total mass of the system.

Friends

- std::istream & operator>> (std::istream &is, particles &partc)

  Input(Initialize) variables with istream.
- std::ostream & operator<< (std::ostream &os, const particles &partc)

  Output variables to ostream.
- ActiveScalarArray & operator>> (ActiveScalarArray &data, particles &partc)

  Input variables with plain scalar array.
- ActiveScalarArray & operator<< (ActiveScalarArray &data, const particles &partc)

  Output variables to plain scalar array.

7.22.1 Detailed Description

template<typename Dtype, size_t ArraySize>
class particles< Dtype, ArraySize >

Class of dynamical variable.

7.22.2 Member Typedef Documentation

7.22.2.1 ActiveScalarArray

```
template<typename Dtype , size_t ArraySize>
using particles< Dtype, ArraySize >::ActiveScalarArray = Container<Scalar, activeScalar>
```

7.22.2.2 Container

```
template<typename Dtype , size_t ArraySize>
template<typename T , size_t S>
using particles< Dtype, ArraySize >::Container = typename type::template Container<T, S>
```

7.22.2.3 IntArray

```
template<typename Dtype , size_t ArraySize>
using particles< Dtype, ArraySize >::IntArray = typename type::IntArray
```

7.22.2.4 Scalar

```
template<typename Dtype , size_t ArraySize>
using particles< Dtype, ArraySize >::Scalar = typename type::Scalar
```

7.22.2.5 ScalarArray

```
template<typename Dtype , size_t ArraySize>
using particles< Dtype, ArraySize >::ScalarArray = typename type::ScalarArray
```

7.22.2.6 type

```
template<typename Dtype , size_t ArraySize>
using particles< Dtype, ArraySize >::type = SpaceH::ProtoType<Dtype, ArraySize>
```

7.22.2.7 Vector

```
template<typename Dtype , size_t ArraySize>
using particles< Dtype, ArraySize >::Vector = typename type::Vector
```

7.22.2.8 VectorArray

```
template<typename Dtype , size_t ArraySize>
using particles< Dtype, ArraySize >::VectorArray = typename type::VectorArray
```

7.22.3 Member Function Documentation

7.22.3.1 advancePos()

```
template<typename Dtype , size_t ArraySize>
void particles< Dtype, ArraySize >::advancePos (
            Scalar stepSize )  [inline]
```

Advance the position array with internal velocity array.

Parameters

stepSize	The advance step size.

Here is the call graph for this function:

7.22.3.2 advanceTime()

```
template<typename Dtype , size_t ArraySize>
void particles< Dtype, ArraySize >::advanceTime (
             Scalar dt )  [inline]
```

Advance the time.

Parameters

dt	Time increament.

Here is the call graph for this function:

7.22.3.3 advanceVel()

```
template<typename Dtype , size_t ArraySize>
void particles< Dtype, ArraySize >::advanceVel (
             const VectorArray & acc,
             Scalar stepSize )  [inline]
```

Advance the velocity array with given acceleration array.

Parameters

stepSize	The advance step size.
acc	The acceleration array.

Here is the call graph for this function:

7.22.3.4 idn() [1/2]

```
template<typename Dtype , size_t ArraySize>
const IntArray& particles< Dtype, ArraySize >::idn ( ) const  [inline]
```

Particle id array const interface. Reference to attribute.type.

7.22.3.5 idn() [2/2]

```
template<typename Dtype , size_t ArraySize>
const int& particles< Dtype, ArraySize >::idn (
            size_t i ) const  [inline]
```

Particle id const interface. Reference to attribute.type[i].

7.22.3.6 kind() [1/2]

```
template<typename Dtype , size_t ArraySize>
const IntArray& particles< Dtype, ArraySize >::kind ( ) const  [inline]
```

Particle type array const interface. Reference to attribute.type.

7.22.3.7 kind() [2/2]

```
template<typename Dtype , size_t ArraySize>
const int& particles< Dtype, ArraySize >::kind (
            size_t i ) const  [inline]
```

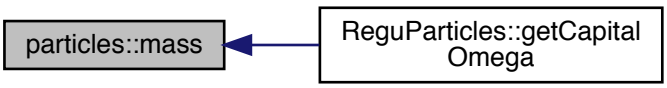
Particle type const interface. Reference to attribute.type[i].

7.22.3.8 mass() [1/2]

```
template<typename Dtype , size_t ArraySize>
const ScalarArray& particles< Dtype, ArraySize >::mass ( ) const  [inline]
```

Mass array const interface. Reference to attribute.mass.

Here is the caller graph for this function:



7.22.3.9 mass() [2/2]

```
template<typename Dtype , size_t ArraySize>
const Scalar& particles< Dtype, ArraySize >::mass (
            size_t i ) const  [inline]
```

Mass const interface. Reference to attribute.mass[i].

7.22.3.10 particleNumber()

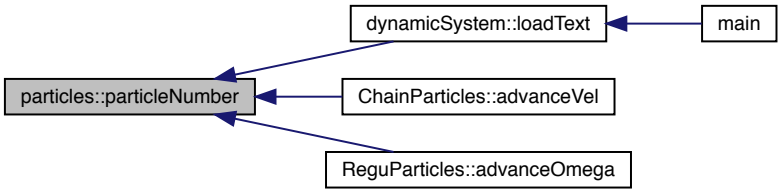
```
template<typename Dtype , size_t ArraySize>
size_t particles< Dtype, ArraySize >::particleNumber ( ) const  [inline]
```

Get the number of the particles.

Returns

The particle number.

Here is the caller graph for this function:

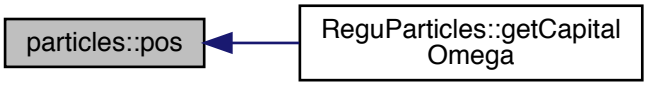


7.22.3.11 pos() [1/2]

```
template<typename Dtype , size_t ArraySize>
const VectorArray& particles< Dtype, ArraySize >::pos ( ) const  [inline]
```

Position array const interface. Reference to state.pos.

Here is the caller graph for this function:



7.22.3.12 pos() [2/2]

```
template<typename Dtype , size_t ArraySize>
const Vector& particles< Dtype, ArraySize >::pos (
            size_t i ) const  [inline]
```

Position vector const interface. Reference to state.pos[i].

7.22.3.13 radius() [1/2]

```
template<typename Dtype , size_t ArraySize>
const ScalarArray& particles< Dtype, ArraySize >::radius ( ) const  [inline]
```

Radius array const interface. Reference to attribute.radius.

7.22.3.14 radius() [2/2]

```
template<typename Dtype , size_t ArraySize>
const Scalar& particles< Dtype, ArraySize >::radius (
            size_t i ) const  [inline]
```

Radius const interface. Reference to attribute.radius[i].

7.22.3.15 time()

```
template<typename Dtype , size_t ArraySize>
const Scalar& particles< Dtype, ArraySize >::time ( ) const  [inline]
```

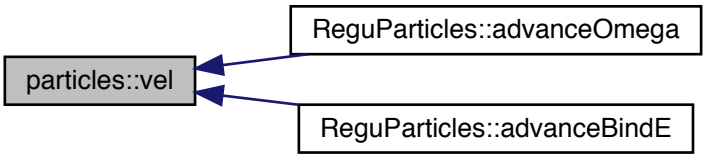
Physical time scalar const interface. Reference to state.time.

7.22.3.16 vel() [1/2]

```
template<typename Dtype , size_t ArraySize>
const VectorArray& particles< Dtype, ArraySize >::vel ( ) const  [inline]
```

Velocity array const interface. Reference to state.vel.

Here is the caller graph for this function:



7.22.3.17 vel() [2/2]

```
template<typename Dtype , size_t ArraySize>
const Vector& particles< Dtype, ArraySize >::vel (
            size_t i ) const  [inline]
```

Velocity vecotr const interface. Reference to state.vel[i].

7.22.4 Friends And Related Function Documentation

7.22.4.1 operator<< [1/2]

```
template<typename Dtype , size_t ArraySize>
std::ostream& operator<< (
            std::ostream & os,
            const particles< Dtype, ArraySize > & partc )  [friend]
```

Output variables to ostream.

7.22.4.2 operator<< [2/2]

```
template<typename Dtype , size_t ArraySize>
ActiveScalarArray& operator<< (
            ActiveScalarArray & data,
            const particles< Dtype, ArraySize > & partc )  [friend]
```

Output variables to plain scalar array.

7.22.4.3 operator>> [1/2]

```
template<typename Dtype , size_t ArraySize>
std::istream& operator>> (
            std::istream & is,
            particles< Dtype, ArraySize > & partc )  [friend]
```

Input(Initialize) variables with istream.

7.22.4.4 operator>> [2/2]

```
template<typename Dtype , size_t ArraySize>
ActiveScalarArray& operator>> (
            ActiveScalarArray & data,
            particles< Dtype, ArraySize > & partc )  [friend]
```

Input variables with plain scalar array.

7.22.5 Member Data Documentation

7.22.5.1 activeScalar

```
template<typename Dtype , size_t ArraySize>
constexpr size_t particles< Dtype, ArraySize >::activeScalar {6*type::arraySize + 1}  [static]
```

7.22.5.2 dataStruct

```
template<typename Dtype , size_t ArraySize>
constexpr SpaceH::DATASTRUCT particles< Dtype, ArraySize >::dataStruct {SpaceH::DATASTRUCT::PLAIN}
[static]
```

7.22.5.3 idn_

```
template<typename Dtype , size_t ArraySize>
IntArray particles< Dtype, ArraySize >::idn_  [protected]
```

Id Array of the particles. Element is int.

7.22.5.4 mass_

```
template<typename Dtype , size_t ArraySize>
ScalarArray particles< Dtype, ArraySize >::mass_  [protected]
```

Mass array of the particles. Element is Scalar.

7.22.5.5 pos_

```
template<typename Dtype , size_t ArraySize>
VectorArray particles< Dtype, ArraySize >::pos_  [protected]
```

Position array of the particles. Element is 3D vector.

7.22.5.6 radius_

```
template<typename Dtype , size_t ArraySize>
ScalarArray particles< Dtype, ArraySize >::radius_  [protected]
```

Radius array of the particles. Element is Scalar.

7.22.5.7 time_

```
template<typename Dtype , size_t ArraySize>
Scalar particles< Dtype, ArraySize >::time_  [protected]
```

The physical time of the dynamic system.

7.22.5.8 totalMass_

```
template<typename Dtype , size_t ArraySize>
Scalar particles< Dtype, ArraySize >::totalMass_  [protected]
```

The total mass of the system.

7.22.5.9 type_

```
template<typename Dtype , size_t ArraySize>
IntArray particles< Dtype, ArraySize >::type_  [protected]
```

Type Array of the particles. Element is int.

7.22.5.10 vel_

```
template<typename Dtype , size_t ArraySize>
VectorArray particles< Dtype, ArraySize >::vel_  [protected]
```

Velocity array of the particles. Element is 3D vector.

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

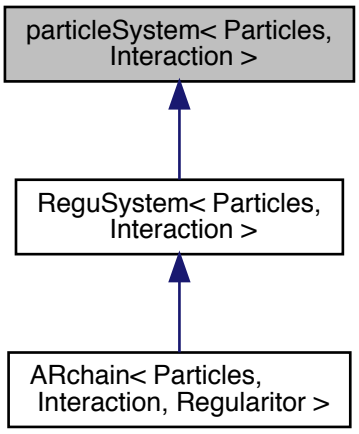
- particles.h

7.23 particleSystem< Particles, Interaction > Class Template Reference

Base class of particle System.

```
#include <particleSystem.h>
```

Inheritance diagram for particleSystem< Particles, Interaction >:



Public Types

- using type = typename Particles::type
- using Scalar = typename type::Scalar
- using Vector = typename type::Vector
- using VectorArray = typename type::VectorArray
- using ScalarArray = typename type::ScalarArray
- using IntArray = typename type::IntArray
- using ActiveScalarArray = typename Particles::ActiveScalarArray

Public Member Functions

- size_t particleNumber ()

  Get the number of the particles.
- const Scalar & time () const

  Physical time scalar const interface. Reference to state.time.
- const VectorArray & pos () const

  Position array const interface. Reference to state.pos.
- const VectorArray & vel () const

  Velocity array const interface. Reference to state.vel.
- const ScalarArray & mass () const

  Mass array const interface. Reference to attribute.mass.
- const ScalarArray & radius () const

  Radius array const interface. Reference to attribute.radius.
- const IntArray & kind () const

  Particle type array const interface. Reference to attribute.type.
- const IntArray & idn () const

  Particle id array const interface. Reference to attribute.type.
- const Vector & pos (size_t i) const

  Position vector const interface. Reference to state.pos[i].
- const Vector & vel (size_t i) const

  Velocity vecotr const interface. Reference to state.vel[i].
- const Scalar & mass (size_t i) const

  Mass const interface. Reference to attribute.mass[i].
- const Scalar & radius (size_t i) const

  Radius const interface. Reference to attribute.radius[i].
- const int & kind (size_t i) const

  Particle type const interface. Reference to attribute.type[i].
- const int & idn (size_t i) const

  Particle id const interface. Reference to attribute.type[i].
- Scalar timeScale (Scalar scale)

  Interface to rescale the time.
- void drift (Scalar stepSize)

  Advance position one step with current velocity.
- void kick (Scalar stepSize)

  Advance velocity one step with current acceleration.
- void preIterProcess ()

  Preprocess before iteration.
- void afterIterProcess ()

  After process after iteration.
- virtual ~particleSystem ()

  Virtualize default destructor.

Static Public Attributes

- static constexpr size_t arraySize {type::arraySize}

Protected Attributes

- Particles partc

  Particle class.
- Interaction act

  Interaction class.

Friends

- std::ostream & operator<< (std::ostream &os, const particleSystem &sys)

  Overload operator <<.
- std::istream & operator>> (std::istream &is, particleSystem &sys)

  Input from istream.
- ActiveScalarArray & operator>> (ActiveScalarArray &data, particleSystem &sys)

  Input variables with plain scalar array.
- ActiveScalarArray & operator<< (ActiveScalarArray &data, const particleSystem &sys)

  Output variables to plain scalar array.

7.23.1 Detailed Description

template<typename Particles, typename Interaction>
class particleSystem< Particles, Interaction >

Base class of particle System.

Base particles system class. Other particle system can inherit this class. Considering the performance, we don't set virtual function.

7.23.2 Member Typedef Documentation

7.23.2.1 ActiveScalarArray

```
template<typename Particles , typename Interaction >
using particleSystem< Particles, Interaction >::ActiveScalarArray = typename Particles::↩
ActiveScalarArray
```

7.23.2.2 IntArray

```
template<typename Particles , typename Interaction >
using particleSystem< Particles, Interaction >::IntArray = typename type::IntArray
```

7.23.2.3 Scalar

```
template<typename Particles , typename Interaction >
using particleSystem< Particles, Interaction >::Scalar = typename type::Scalar
```

7.23.2.4 ScalarArray

```
template<typename Particles , typename Interaction >
using particleSystem< Particles, Interaction >::ScalarArray = typename type::ScalarArray
```

7.23.2.5 type

```
template<typename Particles , typename Interaction >
using particleSystem< Particles, Interaction >::type = typename Particles::type
```

7.23.2.6 Vector

```
template<typename Particles , typename Interaction >
using particleSystem< Particles, Interaction >::Vector = typename type::Vector
```

7.23.2.7 VectorArray

```
template<typename Particles , typename Interaction >
using particleSystem< Particles, Interaction >::VectorArray = typename type::VectorArray
```

7.23.3 Constructor & Destructor Documentation

7.23.3.1 ~particleSystem()

```
template<typename Particles , typename Interaction >
virtual particleSystem< Particles, Interaction >::~particleSystem ( )  [inline], [virtual]
```

Virtualize default destructor.

7.23.4 Member Function Documentation

7.23.4.1 afterIterProcess()

```
template<typename Particles , typename Interaction >
void particleSystem< Particles, Interaction >::afterIterProcess ( )  [inline]
```

After process after iteration.

7.23.4.2 drift()

```
template<typename Particles , typename Interaction >
void particleSystem< Particles, Interaction >::drift (
            Scalar stepSize )  [inline]
```

Advance position one step with current velocity.

7.23.4.3 idn() [1/2]

```
template<typename Particles , typename Interaction >
const IntArray& particleSystem< Particles, Interaction >::idn ( ) const  [inline]
```

Particle id array const interface. Reference to attribute.type.

7.23.4.4 idn() [2/2]

```
template<typename Particles , typename Interaction >
const int& particleSystem< Particles, Interaction >::idn (
            size_t i ) const  [inline]
```

Particle id const interface. Reference to attribute.type[i].

7.23.4.5 kick()

```
template<typename Particles , typename Interaction >
void particleSystem< Particles, Interaction >::kick (
            Scalar stepSize )  [inline]
```

Advance velocity one step with current acceleration.

7.23.4.6 kind() [1/2]

```
template<typename Particles , typename Interaction >
const IntArray& particleSystem< Particles, Interaction >::kind ( ) const  [inline]
```

Particle type array const interface. Reference to attribute.type.

7.23.4.7 kind() [2/2]

```
template<typename Particles , typename Interaction >
const int& particleSystem< Particles, Interaction >::kind (
            size_t i ) const  [inline]
```

Particle type const interface. Reference to attribute.type[i].

7.23.4.8 mass() [1/2]

```
template<typename Particles , typename Interaction >
const ScalarArray& particleSystem< Particles, Interaction >::mass ( ) const  [inline]
```

Mass array const interface. Reference to attribute.mass.

7.23.4.9 mass() [2/2]

```
template<typename Particles , typename Interaction >
const Scalar& particleSystem< Particles, Interaction >::mass (
            size_t i ) const  [inline]
```

Mass const interface. Reference to attribute.mass[i].

7.23.4.10 particleNumber()

```
template<typename Particles , typename Interaction >
size_t particleSystem< Particles, Interaction >::particleNumber ( )  [inline]
```

Get the number of the particles.

Returns

The particle number.

7.23.4.11 pos() [1/2]

```
template<typename Particles , typename Interaction >
const VectorArray& particleSystem< Particles, Interaction >::pos ( ) const  [inline]
```

Position array const interface. Reference to state.pos.

7.23.4.12 pos() [2/2]

```
template<typename Particles , typename Interaction >
const Vector& particleSystem< Particles, Interaction >::pos (
            size_t i ) const  [inline]
```

Position vector const interface. Reference to state.pos[i].

7.23.4.13 preIterProcess()

```
template<typename Particles , typename Interaction >
void particleSystem< Particles, Interaction >::preIterProcess ( )  [inline]
```

Preprocess before iteration.

7.23.4.14 radius() [1/2]

```
template<typename Particles , typename Interaction >
const ScalarArray& particleSystem< Particles, Interaction >::radius ( ) const  [inline]
```

Radius array const interface. Reference to attribute.radius.

7.23.4.15 radius() [2/2]

```
template<typename Particles , typename Interaction >
const Scalar& particleSystem< Particles, Interaction >::radius (
            size_t i ) const  [inline]
```

Radius const interface. Reference to attribute.radius[i].

7.23.4.16 time()

```
template<typename Particles , typename Interaction >
const Scalar& particleSystem< Particles, Interaction >::time ( ) const  [inline]
```

Physical time scalar const interface. Reference to state.time.

7.23.4.17 timeScale()

```
template<typename Particles , typename Interaction >
Scalar particleSystem< Particles, Interaction >::timeScale (
            Scalar scale )  [inline]
```

Interface to rescale the time.

Interace used by dynamic system. Transfer integration time(For some system, integration time is different from physical time) to physical time.

Returns

The phsyical time.

7.23.4.18 vel() [1/2]

```
template<typename Particles , typename Interaction >
const VectorArray& particleSystem< Particles, Interaction >::vel ( ) const  [inline]
```

Velocity array const interface. Reference to state.vel.

7.23.4.19 vel() [2/2]

```
template<typename Particles , typename Interaction >
const Vector& particleSystem< Particles, Interaction >::vel (
            size_t i ) const  [inline]
```

Velocity vecotr const interface. Reference to state.vel[i].

7.23.5 Friends And Related Function Documentation

7.23.5.1 operator<< [1/2]

```
template<typename Particles , typename Interaction >
std::ostream& operator<< (
            std::ostream & os,
            const particleSystem< Particles, Interaction > & sys )  [friend]
```

Overload operator <<.

7.23.5.2 operator<< [2/2]

```
template<typename Particles , typename Interaction >
ActiveScalarArray& operator<< (
            ActiveScalarArray & data,
            const particleSystem< Particles, Interaction > & sys )  [friend]
```

Output variables to plain scalar array.

7.23.5.3 operator>> [1/2]

```
template<typename Particles , typename Interaction >
std::istream& operator>> (
            std::istream & is,
            particleSystem< Particles, Interaction > & sys )  [friend]
```

Input from istream.

7.23.5.4 operator>> [2/2]

```
template<typename Particles , typename Interaction >
ActiveScalarArray& operator>> (
            ActiveScalarArray & data,
            particleSystem< Particles, Interaction > & sys )  [friend]
```

Input variables with plain scalar array.

7.23.6 Member Data Documentation

7.23.6.1 act

```
template<typename Particles , typename Interaction >
Interaction particleSystem< Particles, Interaction >::act  [protected]
```

Interaction class.

7.23.6.2 arraySize

```
template<typename Particles , typename Interaction >
constexpr size_t particleSystem< Particles, Interaction >::arraySize {type::arraySize}  [static]
```

7.23.6.3 partc

```
template<typename Particles , typename Interaction >
Particles particleSystem< Particles, Interaction >::partc  [protected]
```

Particle class.

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

- particleSystem.h

7.24 SpaceH::PostNewtonian< Scalar > Class Template Reference

Post newtonian pair interaction functor(c++ std11)

```
#include <forces.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.

7.24.1 Detailed Description

template<typename Scalar>
class SpaceH::PostNewtonian< Scalar >

Post newtonian pair interaction functor(c++ std11)

7.24.2 Member Function Documentation

7.24.2.1 operator()()

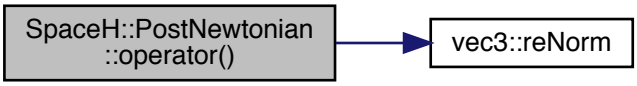
```
template<typename Scalar >
void SpaceH::PostNewtonian< Scalar >::operator() (
             Scalar m1,
             Scalar m2,
             Vector & dr,
             Vector & dv,
             Vector & v1,
             Vector & v2,
             Vector & acc1,
             Vector & acc2 )  [inline]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

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/forces.h

7.25 ProgressBar Class Reference

```
#include <timmer.h>
```

Public Types

- typedef std::chrono::high_resolution_clock Clock
- typedef std::chrono::time_point< Clock > ClockTime
- typedef std::chrono::milliseconds ms

Public Member Functions

- ProgressBar ()=delete
- ProgressBar (double upperLimit, unsigned int precision=10000)
- void autoShow (double time)
- void reset (double upperLimit, int precision=10000)
- void start ()
- double getTime ()

7.25.1 Member Typedef Documentation

7.25.1.1 Clock

```
typedef std::chrono::high_resolution_clock ProgressBar::Clock
```

7.25.1.2 ClockTime

```
typedef std::chrono::time_point<Clock> ProgressBar::ClockTime
```

7.25.1.3 ms

```
typedef std::chrono::milliseconds ProgressBar::ms
```

7.25.2 Constructor & Destructor Documentation

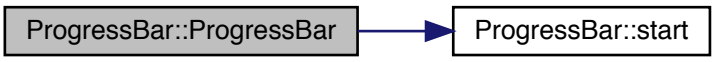
7.25.2.1 ProgressBar() [1/2]

```
ProgressBar::ProgressBar ( )  [delete]
```

7.25.2.2 ProgressBar() [2/2]

```
ProgressBar::ProgressBar (
            double upperLimit,
            unsigned int precision = 10000 )  [inline]
```

Here is the call graph for this function:



7.25.3 Member Function Documentation

7.25.3.1 autoShow()

```
void ProgressBar::autoShow (
            double time )  [inline]
```

Here is the call graph for this function:

7.25.3.2 getTime()

```
double ProgressBar::getTime ( )  [inline]
```

7.25.3.3 reset()

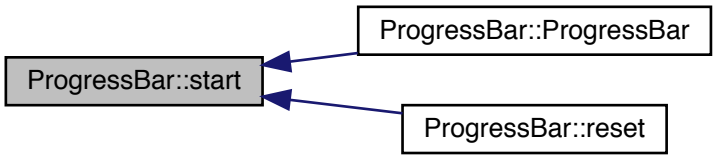
```
void ProgressBar::reset (
            double upperLimit,
            int precision = 10000 )  [inline]
```

Here is the call graph for this function:

7.25.3.4 start()

```
void ProgressBar::start ( )  [inline]
```

Here is the caller graph for this function:



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

- tools/timmer.h

7.26 SpaceH::ProtoType< Dtype, Size > Struct Template Reference

```
#include <protoType.h>
```

Public Types

- template<typename T , size_t S>
  using Container = ArrayWrapper< T, S >
- using Scalar = Dtype
- using Vector = vec3< Scalar >
- using VectorArray = Container< Vector, Size >
- using ScalarArray = Container< Scalar, Size >
- using IntArray = Container< int, Size >
- using SizeArray = Container< size_t, Size >
- using IndexArray = SizeArray

Static Public Attributes

- static constexpr size_t arraySize {Size}

7.26.1 Member Typedef Documentation

7.26.1.1 Container

```
template<typename Dtype , size_t Size>
template<typename T , size_t S>
using SpaceH::ProtoType< Dtype, Size >::Container = ArrayWrapper<T, S>
```

7.26.1.2 IndexArray

```
template<typename Dtype , size_t Size>
using SpaceH::ProtoType< Dtype, Size >::IndexArray = SizeArray
```

7.26.1.3 IntArray

```
template<typename Dtype , size_t Size>
using SpaceH::ProtoType< Dtype, Size >::IntArray = Container<int, Size>
```

7.26.1.4 Scalar

```
template<typename Dtype , size_t Size>
using SpaceH::ProtoType< Dtype, Size >::Scalar = Dtype
```

7.26.1.5 ScalarArray

```
template<typename Dtype , size_t Size>
using SpaceH::ProtoType< Dtype, Size >::ScalarArray = Container<Scalar, Size>
```

7.26.1.6 SizeArray

```
template<typename Dtype , size_t Size>
using SpaceH::ProtoType< Dtype, Size >::SizeArray = Container<size_t, Size>
```

7.26.1.7 Vector

```
template<typename Dtype , size_t Size>
using SpaceH::ProtoType< Dtype, Size >::Vector = vec3<Scalar>
```

7.26.1.8 VectorArray

```
template<typename Dtype , size_t Size>
using SpaceH::ProtoType< Dtype, Size >::VectorArray = Container<Vector, Size>
```

7.26.2 Member Data Documentation

7.26.2.1 arraySize

```
template<typename Dtype , size_t Size>
constexpr size_t SpaceH::ProtoType< Dtype, Size >::arraySize {Size}  [static]
```

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

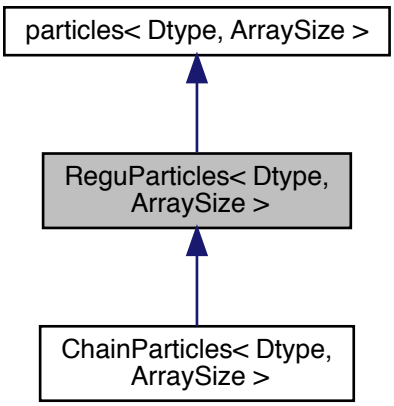
- protoType.h

7.27 ReguParticles< Dtype, ArraySize > Class Template Reference

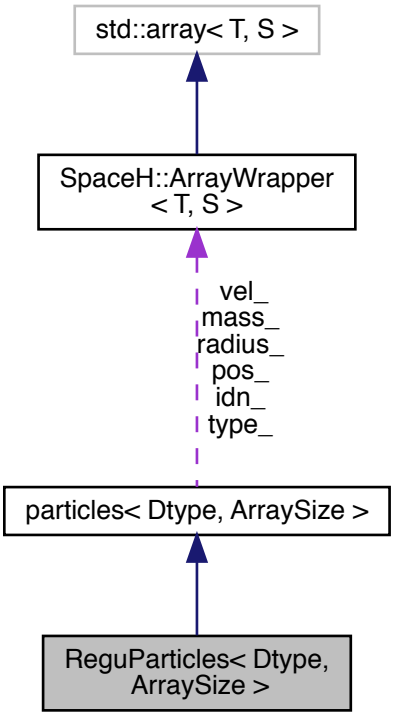
Class of dynamical system with regularization variables.

`#include <regularState.h>`

Inheritance diagram for ReguParticles< Dtype, ArraySize >:



Collaboration diagram for ReguParticles< Dtype, ArraySize >:



Public Types

- using Base = particles< Dtype, ArraySize >
- template<typename T , size_t S>
  using Container = typename type::template Container< T, S >
- using Scalar = typename type::Scalar
- using Vector = typename type::Vector
- using VectorArray = typename type::VectorArray
- using ActiveScalarArray = Container< Scalar, activeScalar >
- using type = SpaceH::ProtoType< Dtype, ArraySize >

Public Member Functions

- const Scalar & omega () const

  Omega scalar const interface. Reference to state.time.
- const Scalar & bindE () const

  BindE scalar const interface. Reference to state.time.
- void advanceOmega (const VectorArray &velIndepAcc, const VectorArray &vel, Scalar stepSize)

  Advance the Omega.
- void advanceBindE (const VectorArray &velDepAcc, const VectorArray &vel, Scalar stepSize)

  Advance the bindE.

Static Public Attributes

- static constexpr size_t activeScalar {6*type::arraySize + 3}

Protected Member Functions

- Scalar getCapitalOmega ()

  Calculate the regularized variable Omega.

Protected Attributes

- Scalar omega_
- Scalar bindE_

Friends

- std::istream & operator>> (std::istream &is, ReguParticles &partc)

  Input(Initialize) variables with istream.
- ActiveScalarArray & operator>> (ActiveScalarArray &data, ReguParticles &partc)

  Input variables with plain scalar array.
- ActiveScalarArray & operator<< (ActiveScalarArray &data, const ReguParticles &partc)

  Output variables to plain scalar array.

7.27.1 Detailed Description

template<typename Dtype, size_t ArraySize>
class ReguParticles< Dtype, ArraySize >

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 .

7.27.2 Member Typedef Documentation

7.27.2.1 ActiveScalarArray

```
template<typename Dtype , size_t ArraySize>
using ReguParticles< Dtype, ArraySize >::ActiveScalarArray = Container<Scalar, activeScalar>
```

7.27.2.2 Base

```
template<typename Dtype , size_t ArraySize>
using ReguParticles< Dtype, ArraySize >::Base = particles<Dtype, ArraySize>
```

7.27.2.3 Container

```
template<typename Dtype , size_t ArraySize>
template<typename T , size_t S>
using ReguParticles< Dtype, ArraySize >::Container = typename type::template Container<T, S>
```

7.27.2.4 Scalar

```
template<typename Dtype , size_t ArraySize>
using ReguParticles< Dtype, ArraySize >::Scalar = typename type::Scalar
```

7.27.2.5 type

```
template<typename Dtype , size_t ArraySize>
using particles< Dtype, ArraySize >::type = SpaceH::ProtoType<Dtype, ArraySize>
```

7.27.2.6 Vector

```
template<typename Dtype , size_t ArraySize>
using ReguParticles< Dtype, ArraySize >::Vector = typename type::Vector
```

7.27.2.7 VectorArray

```
template<typename Dtype , size_t ArraySize>
using ReguParticles< Dtype, ArraySize >::VectorArray = typename type::VectorArray
```

7.27.3 Member Function Documentation

7.27.3.1 advanceBindE()

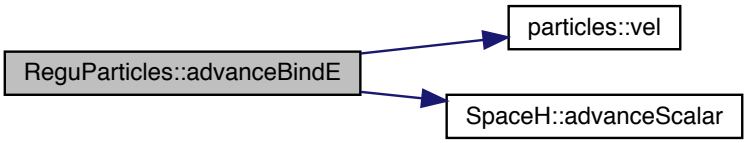
```
template<typename Dtype , size_t ArraySize>
void ReguParticles< Dtype, ArraySize >::advanceBindE (
            const VectorArray & velDepAcc,
            const VectorArray & vel,
            Scalar stepSize )  [inline]
```

Advance the bindE.

Parameters

velDepAcc	Velocity dependent acceleration array.
vel	Velocity array.
stepSize	Time stepSize.

Here is the call graph for this function:



7.27.3.2 advanceOmega()

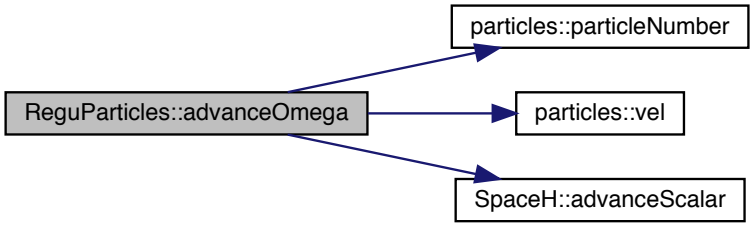
```
template<typename Dtype , size_t ArraySize>
void ReguParticles< Dtype, ArraySize >::advanceOmega (
            const VectorArray & velIndepAcc,
            const VectorArray & vel,
            Scalar stepSize )  [inline]
```

Advance the Omega.

Parameters

velIndepAcc	Velocity independent acceleration array.
vel	Velocity array.
stepSize	Time stepSize.

Here is the call graph for this function:



7.27.3.3 bindE()

```
template<typename Dtype , size_t ArraySize>
const Scalar& ReguParticles< Dtype, ArraySize >::bindE ( ) const  [inline]
```

BindE scalar const interface. Reference to state.time.

7.27.3.4 getCapitalOmega()

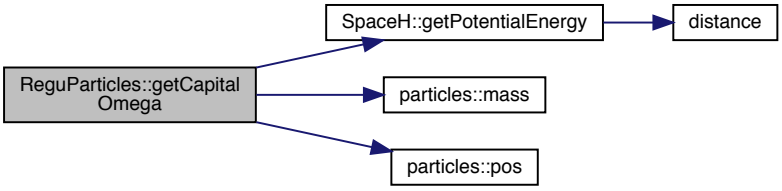
```
template<typename Dtype , size_t ArraySize>
Scalar ReguParticles< Dtype, ArraySize >::getCapitalOmega ( )  [inline], [protected]
```

Calculate the regularized variable Omega.

Returns

The value of capital omega.

Here is the call graph for this function:



7.27.3.5 omega()

```
template<typename Dtype , size_t ArraySize>
const Scalar& ReguParticles< Dtype, ArraySize >::omega ( ) const  [inline]
```

Omega scalar const interface. Reference to state.time.

7.27.4 Friends And Related Function Documentation

7.27.4.1 operator<<

```
template<typename Dtype , size_t ArraySize>
ActiveScalarArray& operator<< (
            ActiveScalarArray & data,
            const ReguParticles< Dtype, ArraySize > & partc )  [friend]
```

Output variables to plain scalar array.

7.27.4.2 operator>> [1/2]

```
template<typename Dtype , size_t ArraySize>
std::istream& operator>> (
            std::istream & is,
            ReguParticles< Dtype, ArraySize > & partc )  [friend]
```

Input(Initialize) variables with istream.

7.27.4.3 operator>> [2/2]

```
template<typename Dtype , size_t ArraySize>
ActiveScalarArray& operator>> (
            ActiveScalarArray & data,
            ReguParticles< Dtype, ArraySize > & partc )  [friend]
```

Input variables with plain scalar array.

7.27.5 Member Data Documentation

7.27.5.1 activeScalar

```
template<typename Dtype , size_t ArraySize>
constexpr size_t ReguParticles< Dtype, ArraySize >::activeScalar {6*type::arraySize + 3}  [static]
```

7.27.5.2 bindE_

```
template<typename Dtype , size_t ArraySize>
Scalar ReguParticles< Dtype, ArraySize >::bindE_  [protected]
```

7.27.5.3 omega_

```
template<typename Dtype , size_t ArraySize>
Scalar ReguParticles< Dtype, ArraySize >::omega_  [protected]
```

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

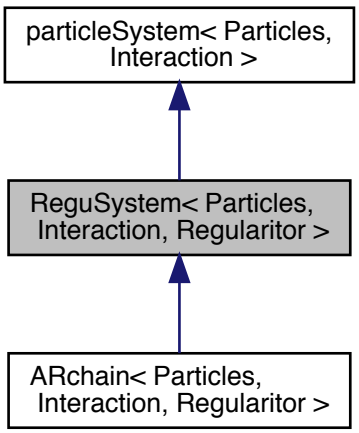
- particleSystem/reguSystem/regularState.h

7.28 ReguSystem< Particles, Interaction, Regularitor > Class Template Reference

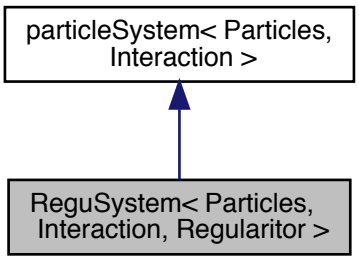
Regularized particle System.

`#include <reguSystem.h>`

Inheritance diagram for ReguSystem< Particles, Interaction, Regularitor >:



Collaboration diagram for ReguSystem< Particles, Interaction, Regularitor >:



Public Types

- using Base = particleSystem< Particles, Interaction >
- using Scalar = typename type::Scalar
- using Vector = typename type::Vector
- using VectorArray = typename type::VectorArray
- using type = typename Particles::type
- using ActiveScalarArray = typename Particles::ActiveScalarArray

Public Member Functions

- Scalar & omega ()

  Omega interface. Reference to partc.omega.
- Scalar & bindE ()

  Bindine energy interface. Reference to partc.bindE.
- void drift (Scalar stepSize)

  Advance position one step with current velocity.
- void kick (Scalar stepSize)

  Advance velocity one step with current acceleration.
- Scalar timeScale (Scalar scale)

  Interface to rescale the time.

Public Attributes

- Interaction act

  Interaction class.
- Particles partc

  Particle class.

Additional Inherited Members

7.28.1 Detailed Description

template<typename Particles, typename Interaction, typename Regularitor>
class ReguSystem< Particles, Interaction, Regularitor >

Regularized particle System.

Regularied particle system. See details in https://link.springer.com/article/10.1023%2FA%3A1008368322547 , http://iopscience.iop.org/article/10.1086/301102/meta and https://link.springer.com/article/10.1023%2FA%3A1021149313347 .

7.28.2 Member Typedef Documentation

7.28.2.1 ActiveScalarArray

```
template<typename Particles , typename Interaction , typename Regularitor >
using particleSystem< Particles, Interaction >::ActiveScalarArray = typename Particles::↩
ActiveScalarArray
```

7.28.2.2 Base

```
template<typename Particles , typename Interaction , typename Regularitor >
using ReguSystem< Particles, Interaction, Regularitor >::Base = particleSystem<Particles,
Interaction>
```

7.28.2.3 Scalar

```
template<typename Particles , typename Interaction , typename Regularitor >
using ReguSystem< Particles, Interaction, Regularitor >::Scalar = typename type::Scalar
```

7.28.2.4 type

```
template<typename Particles , typename Interaction , typename Regularitor >
using particleSystem< Particles, Interaction >::type = typename Particles::type
```

7.28.2.5 Vector

```
template<typename Particles , typename Interaction , typename Regularitor >
using ReguSystem< Particles, Interaction, Regularitor >::Vector = typename type::Vector
```

7.28.2.6 VectorArray

```
template<typename Particles , typename Interaction , typename Regularitor >
using ReguSystem< Particles, Interaction, Regularitor >::VectorArray = typename type::Vector↩
Array
```

7.28.3 Member Function Documentation

7.28.3.1 bindE()

```
template<typename Particles , typename Interaction , typename Regularitor >
Scalar& ReguSystem< Particles, Interaction, Regularitor >::bindE ( )  [inline]
```

Bindine energy interface. Reference to partc.bindE.

7.28.3.2 drift()

```
template<typename Particles , typename Interaction , typename Regularitor >
void ReguSystem< Particles, Interaction, Regularitor >::drift (
            Scalar stepSize )  [inline]
```

Advance position one step with current velocity.

Advance position array and physical time one step with current integration step size and velocity.

Parameters

timeStepSize	Integration step size, will be transfered to physical time in the function.

7.28.3.3 kick()

```
template<typename Particles , typename Interaction , typename Regularitor >
void ReguSystem< Particles, Interaction, Regularitor >::kick (
            Scalar stepSize )  [inline]
```

Advance velocity one step with current acceleration.

Advance velocity array one step with current integration step size and accelerations.

Parameters

stepSize	Integration step size, will be transfered to physical time in the function.

7.28.3.4 omega()

```
template<typename Particles , typename Interaction , typename Regularitor >
Scalar& ReguSystem< Particles, Interaction, Regularitor >::omega ( )  [inline]
```

Omega interface. Reference to partc.omega.

7.28.3.5 timeScale()

```
template<typename Particles , typename Interaction , typename Regularitor >
Scalar ReguSystem< Particles, Interaction, Regularitor >::timeScale (
            Scalar scale )  [inline]
```

Interface to rescale the time.

Interace used by dynamic system. Transfer integration time to physical time.

Returns

The phsyical time.

7.28.4 Member Data Documentation

7.28.4.1 act

```
template<typename Particles , typename Interaction , typename Regularitor >
Interaction particleSystem< Particles, Interaction >::act
```

Interaction class.

7.28.4.2 partc

```
template<typename Particles , typename Interaction , typename Regularitor >
Particles particleSystem< Particles, Interaction >::partc
```

Particle class.

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

- particleSystem/reguSystem/reguSystem.h

7.29 symplectic10th< ParticSys > Class Template Reference

Eighth order symplectic integrator.

```
#include <symplectic10th.h>
```

Public Types

- using type = typename ParticSys::type
- using Scalar = typename type::Scalar

Public Member Functions

- void integrate (ParticSys &particles, Scalar stepLength)

  Interface to integrate particle system.

Static Public Attributes

- static const int order {10}

  Order of the integrator.

7.29.1 Detailed Description

template<typename ParticSys>
class symplectic10th< ParticSys >

Eighth order symplectic integrator.

7.29.2 Member Typedef Documentation

7.29.2.1 Scalar

```
template<typename ParticSys >
using symplectic10th< ParticSys >::Scalar = typename type::Scalar
```

7.29.2.2 type

```
template<typename ParticSys >
using symplectic10th< ParticSys >::type = typename ParticSys::type
```

7.29.3 Member Function Documentation

7.29.3.1 integrate()

```
template<typename ParticSys >
void symplectic10th< 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.

7.29.4 Member Data Documentation

7.29.4.1 order

```
template<typename ParticSys >
const int symplectic10th< ParticSys >::order {10}  [static]
```

Order of the integrator.

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

- integrator/symplectic/symplectic10th.h

7.30 symplectic2th< ParticSys > Class Template Reference

Second order symplectic integrator.

`#include <symplectic2th.h>`

Public Types

- using type = typename ParticSys::type
- using Scalar = typename type::Scalar

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.

7.30.1 Detailed Description

template<typename ParticSys>
class symplectic2th< ParticSys >

Second order symplectic integrator.

7.30.2 Member Typedef Documentation

7.30.2.1 Scalar

```
template<typename ParticSys >
using symplectic2th< ParticSys >::Scalar = typename type::Scalar
```

7.30.2.2 type

```
template<typename ParticSys >
using symplectic2th< ParticSys >::type = typename ParticSys::type
```

7.30.3 Member Function Documentation

7.30.3.1 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.

7.30.4 Member Data Documentation

7.30.4.1 order

```
template<typename ParticSys >
const int symplectic2th< ParticSys >::order {2}  [static]
```

Order of the integrator.

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

- integrator/symplectic/symplectic2th.h

7.31 symplectic4th< ParticSys > Class Template Reference

Fourth order symplectic integrator.

```
#include <symplectic4th.h>
```

Public Types

- using type = typename ParticSys::type
- using Scalar = typename type::Scalar

Public Member Functions

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

Static Public Attributes

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

7.31.1 Detailed Description

template<typename ParticSys>
class symplectic4th< ParticSys >

Fourth order symplectic integrator.

7.31.2 Member Typedef Documentation

7.31.2.1 Scalar

```
template<typename ParticSys >
using symplectic4th< ParticSys >::Scalar = typename type::Scalar
```

7.31.2.2 type

```
template<typename ParticSys >
using symplectic4th< ParticSys >::type = typename ParticSys::type
```

7.31.3 Member Function Documentation

7.31.3.1 integrate()

```
template<typename ParticSys >
void symplectic4th< 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.

7.31.4 Member Data Documentation

7.31.4.1 order

```
template<typename ParticSys >
const int symplectic4th< ParticSys >::order {4}  [static]
```

Order of the integrator.

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

- integrator/symplectic/symplectic4th.h

7.32 symplectic6th< ParticSys > Class Template Reference

Sixth order symplectic integrator.

```
#include <symplectic6th.h>
```

Public Types

- using type = typename ParticSys::type
- using Scalar = typename type::Scalar

Public Member Functions

- void integrate (ParticSys &particles, Scalar stepLength)

  Interface to integrate particle system.

Static Public Attributes

- static const int order {6}

  Order of the integrator.

7.32.1 Detailed Description

template<typename ParticSys>
class symplectic6th< ParticSys >

Sixth order symplectic integrator.

7.32.2 Member Typedef Documentation

7.32.2.1 Scalar

```
template<typename ParticSys >
using symplectic6th< ParticSys >::Scalar = typename type::Scalar
```

7.32.2.2 type

```
template<typename ParticSys >
using symplectic6th< ParticSys >::type = typename ParticSys::type
```

7.32.3 Member Function Documentation

7.32.3.1 integrate()

```
template<typename ParticSys >
void symplectic6th< 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.

7.32.4 Member Data Documentation

7.32.4.1 order

```
template<typename ParticSys >
const int symplectic6th< ParticSys >::order {6}  [static]
```

Order of the integrator.

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

- integrator/symplectic/symplectic6th.h

7.33 symplectic8th< ParticSys > Class Template Reference

Eighth order symplectic integrator.

```
#include <symplectic8th.h>
```

Public Types

- using type = typename ParticSys::type
- using Scalar = typename type::Scalar

Public Member Functions

- void integrate (ParticSys &particles, Scalar stepLength)

  Interface to integrate particle system.

Static Public Attributes

- static const int order {8}

  Order of the integrator.

7.33.1 Detailed Description

template<typename ParticSys>
class symplectic8th< ParticSys >

Eighth order symplectic integrator.

7.33.2 Member Typedef Documentation

7.33.2.1 Scalar

```
template<typename ParticSys >
using symplectic8th< ParticSys >::Scalar = typename type::Scalar
```

7.33.2.2 type

```
template<typename ParticSys >
using symplectic8th< ParticSys >::type = typename ParticSys::type
```

7.33.3 Member Function Documentation

7.33.3.1 integrate()

```
template<typename ParticSys >
void symplectic8th< 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.

7.33.4 Member Data Documentation

7.33.4.1 order

```
template<typename ParticSys >
const int symplectic8th< ParticSys >::order {8}  [static]
```

Order of the integrator.

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

- integrator/symplectic/symplectic8th.h

7.34 TTL< Particles > Class Template Reference

Time Transform Leapfrog algorithmatic regularization interface.

`#include <regularization.h>`

Public Types

- using type = typename Particles::type
- using Scalar = typename type::Scalar

Public Member Functions

- Scalar getPhysicalPosTime (Particles &partc, Scalar stepSize)

  Calculate the physical time for position advance from integration step size.
- Scalar getPhysicalVelTime (Particles &partc, Scalar stepSize)

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

7.34.1 Detailed Description

template<typename Particles>
class TTL< Particles >

Time Transform Leapfrog algorithmatic regularization interface.

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

7.34.2 Member Typedef Documentation

7.34.2.1 Scalar

```
template<typename Particles >
using TTL< Particles >::Scalar = typename type::Scalar
```

7.34.2.2 type

```
template<typename Particles >
using TTL< Particles >::type = typename Particles::type
```

7.34.3 Member Function Documentation

7.34.3.1 getPhysicalPosTime()

```
template<typename Particles >
Scalar TTL< Particles >::getPhysicalPosTime (
            Particles & partc,
            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.

7.34.3.2 getPhysicalVelTime()

```
template<typename Particles >
Scalar TTL< Particles >::getPhysicalVelTime (
            Particles & partc,
            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/reguSystem/regularization.h

7.35 vec3< T > Struct Template Reference

Self 3D vector class.

`#include <vector3.h>`

Public Types

- using value_type = T

Public Member Functions

- vec3 ()
- 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 T c) const

  Add scalar by wise.
- vec3 operator- (const T c) const

  Subtract scalar by wise.
- vec3 operator* (const T c) const

  Multiply scalar by wise.
- vec3 operator/ (const T 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 T c)
- const vec3 & operator-= (const T c)
- const vec3 & operator*= (const T c)
- const vec3 & operator/= (const T c)
- const vec3 & operator= (const vec3 &v)
- T operator* (const vec3 &v) const

  Inner product.
- T norm () const

  Calculate the norm.
- T normSquare () const

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

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

Public Attributes

- T x
- T y
- T z

Friends

- vec3 operator+ (const T c, const vec3 &v)
- vec3 operator- (const T c, const vec3 &v)
- vec3 operator∗ (const T 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.

7.35.1 Detailed Description

template<typename T>
struct vec3< T >

Self 3D vector class.

7.35.2 Member Typedef Documentation

7.35.2.1 value_type

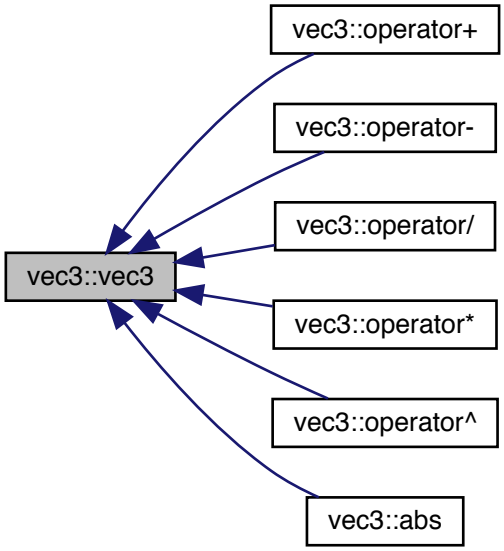
```
template<typename T>
using vec3< T >::value_type = T
```

7.35.3 Constructor & Destructor Documentation

7.35.3.1 vec3() [1/3]

```
template<typename T>
vec3< T >::vec3 ( )  [inline]
```

Here is the caller graph for this function:



7.35.3.2 vec3() [2/3]

```
template<typename T>
vec3< T >::vec3 (
            T vx,
            T vy,
            T vz )  [inline]
```

7.35.3.3 vec3() [3/3]

```
template<typename T>
vec3< T >::vec3 (
            const vec3< T > & v )  [inline]
```

7.35.4 Member Function Documentation

7.35.4.1 abs()

```
template<typename T>
vec3 vec3< T >::abs ( ) const  [inline]
```

Absolute value by wise.

Here is the call graph for this function:

7.35.4.2 norm()

```
template<typename T>
T vec3< T >::norm ( ) const  [inline]
```

Calculate the norm.

7.35.4.3 normSquare()

```
template<typename T>
T vec3< T >::normSquare ( ) const  [inline]
```

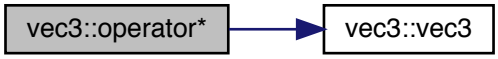
Calcualte the square of the norm.

7.35.4.4 operator∗() [1/2]

```
template<typename T>
vec3 vec3< T >::operator* (
            const T c ) const  [inline]
```

Multiply scalar by wise.

Here is the call graph for this function:



7.35.4.5 operator*() [2/2]

```
template<typename T>
T vec3< T >::operator* (
            const vec3< T > & v ) const  [inline]
```

Inner product.

7.35.4.6 operator*=()

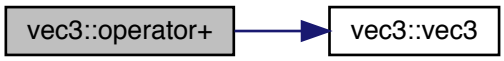
```
template<typename T>
const vec3& vec3< T >::operator*= (
            const T c )  [inline]
```

7.35.4.7 operator+() [1/2]

```
template<typename T>
vec3 vec3< T >::operator+ (
            const vec3< T > & v ) const  [inline]
```

Addition by wise.

Here is the call graph for this function:

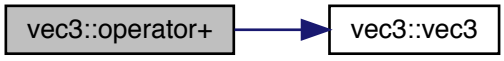


7.35.4.8 operator+() [2/2]

```
template<typename T>
vec3 vec3< T >::operator+ (
            const T c ) const  [inline]
```

Add scalar by wise.

Here is the call graph for this function:



7.35.4.9 operator+=() [1/2]

```
template<typename T>
const vec3& vec3< T >::operator+= (
            const vec3< T > & v )  [inline]
```

7.35.4.10 operator+=() [2/2]

```
template<typename T>
const vec3& vec3< T >::operator+= (
            const T c )  [inline]
```

7.35.4.11 operator-() [1/3]

```
template<typename T>
vec3 vec3< T >::operator- (
            const vec3< T > & v ) const  [inline]
```

Subtraction by wise.

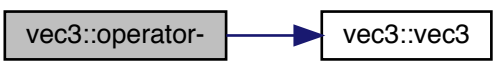
Here is the call graph for this function:

7.35.4.12 operator-() [2/3]

```
template<typename T>
vec3 vec3< T >::operator- (
            const T c ) const  [inline]
```

Subtract scalar by wise.

Here is the call graph for this function:

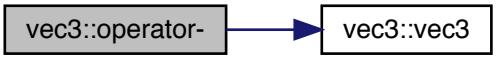


7.35.4.13 operator-() [3/3]

```
template<typename T>
vec3 vec3< T >::operator- ( ) const  [inline]
```

Opposite vector.

Here is the call graph for this function:



7.35.4.14 operator-=() [1/2]

```
template<typename T>
const vec3& vec3< T >::operator-= (
            const vec3< T > & v )  [inline]
```

7.35.4.15 operator-=() [2/2]

```
template<typename T>
const vec3& vec3< T >::operator-= (
            const T c )  [inline]
```

7.35.4.16 operator/() [1/2]

```
template<typename T>
vec3 vec3< T >::operator/ (
            const vec3< T > & v ) const  [inline]
```

Division by wise.

Here is the call graph for this function:

7.35.4.17 operator/() [2/2]

```
template<typename T>
vec3 vec3< T >::operator/ (
            const T c ) const  [inline]
```

Divide scalar by wise.

Here is the call graph for this function:

7.35.4.18 operator/=() [1/2]

```
template<typename T>
const vec3& vec3< T >::operator/= (
            const vec3< T > & v )  [inline]
```

7.35.4.19 operator/=() [2/2]

```
template<typename T>
const vec3& vec3< T >::operator/= (
            const T c )  [inline]
```

7.35.4.20 operator=()

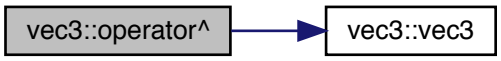
```
template<typename T>
const vec3& vec3< T >::operator= (
            const vec3< T > & v )  [inline]
```

7.35.4.21 operator^()

```
template<typename T>
vec3 vec3< T >::operator^ (
            const vec3< T > & v ) const  [inline]
```

Cross product.

Here is the call graph for this function:

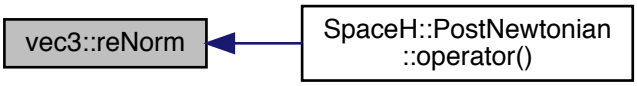


7.35.4.22 reNorm()

```
template<typename T>
T vec3< T >::reNorm ( ) const  [inline]
```

Calculate the inverse of the norm.

Here is the caller graph for this function:



7.35.4.23 setZero()

```
template<typename T>
void vec3< T >::setZero ( )  [inline]
```

7.35.5 Friends And Related Function Documentation

7.35.5.1 operator*

```
template<typename T>
vec3 operator* (
            const T c,
            const vec3< T > & v )  [friend]
```

7.35.5.2 operator+

```
template<typename T>
vec3 operator+ (
            const T c,
            const vec3< T > & v )  [friend]
```

7.35.5.3 operator-

```
template<typename T>
vec3 operator- (
            const T c,
            const vec3< T > & v )  [friend]
```

7.35.5.4 operator<<

```
template<typename T>
std::ostream& operator<< (
            std::ostream & output,
            const vec3< T > & v )  [friend]
```

Output to ostream.

7.35.5.5 operator>>

```
template<typename T>
std::istream& operator>> (
            std::istream & input,
            vec3< T > & v )  [friend]
```

Input from istream.

7.35.6 Member Data Documentation

7.35.6.1 x

```
template<typename T>
T vec3< T >::x
```

7.35.6.2 y

```
template<typename T>
T vec3< T >::y
```

7.35.6.3 z

```
template<typename T>
T vec3< T >::z
```

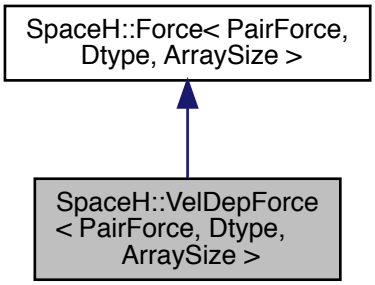
The documentation for this struct was generated from the following file:

- vector3.h

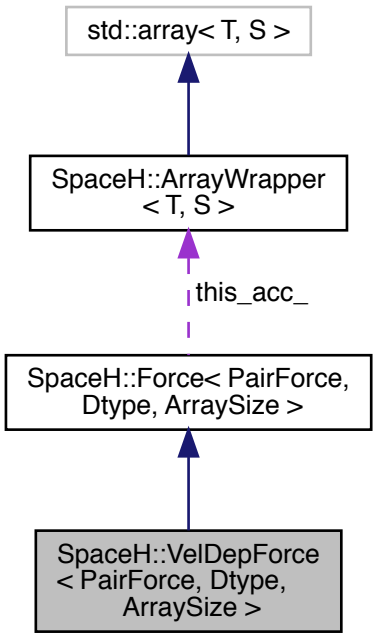
7.36 SpaceH::VelDepForce< PairForce, Dtype, ArraySize > Struct Template Reference

```
#include <forces.h>
```

Inheritance diagram for SpaceH::VelDepForce< PairForce, Dtype, ArraySize >:



Collaboration diagram for SpaceH::VelDepForce< PairForce, Dtype, ArraySize >:



Public Types

- using Base = Force< PairForce, Dtype, ArraySize >
- using type = typename Base::type
- using Scalar = typename type::Scalar
- using Vector = typename type::Vector
- using ScalarArray = typename type::ScalarArray
- using VectorArray = typename type::VectorArray
- using IndexArray = typename type::IndexArray

Public Member Functions

- void calcuAcc (const ScalarArray &mass, const VectorArray &pos, const VectorArray &vel)
- void calcuAcc (const ScalarArray &mass, const VectorArray &pos, const VectorArray &vel, const VectorArray &chainPos, const VectorArray &chainVel, const IndexArray &chainIndex)

Public Attributes

- PairForce force_
- VectorArray this_acc_

Static Public Attributes

- static constexpr bool isVelDep {true}

Additional Inherited Members

7.36.1 Member Typedef Documentation

7.36.1.1 Base

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelDepForce< PairForce, Dtype, ArraySize >::Base = Force<PairForce, Dtype,
ArraySize>
```

7.36.1.2 IndexArray

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelDepForce< PairForce, Dtype, ArraySize >::IndexArray = typename type::IndexArray
```

7.36.1.3 Scalar

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelDepForce< PairForce, Dtype, ArraySize >::Scalar = typename type::Scalar
```

7.36.1.4 ScalarArray

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelDepForce< PairForce, Dtype, ArraySize >::ScalarArray = typename type::ScalarArray
```

7.36.1.5 type

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelDepForce< PairForce, Dtype, ArraySize >::type = typename Base::type
```

7.36.1.6 Vector

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelDepForce< PairForce, Dtype, ArraySize >::Vector = typename type::Vector
```

7.36.1.7 VectorArray

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelDepForce< PairForce, Dtype, ArraySize >::VectorArray = typename type::VectorArray
```

7.36.2 Member Function Documentation

7.36.2.1 calcuAcc() [1/2]

```
template<typename PairForce , typename Dtype , size_t ArraySize>
void SpaceH::VelDepForce< PairForce, Dtype, ArraySize >::calcuAcc (
            const ScalarArray & mass,
            const VectorArray & pos,
            const VectorArray & vel )  [inline]
```

7.36.2.2 calcuAcc() [2/2]

```
template<typename PairForce , typename Dtype , size_t ArraySize>
void SpaceH::VelDepForce< PairForce, Dtype, ArraySize >::calcuAcc (
            const ScalarArray & mass,
            const VectorArray & pos,
            const VectorArray & vel,
            const VectorArray & chainPos,
            const VectorArray & chainVel,
            const IndexArray & chainIndex )  [inline]
```

7.36.3 Member Data Documentation

7.36.3.1 force_

```
template<typename PairForce , typename Dtype , size_t ArraySize>
PairForce SpaceH::Force< PairForce, Dtype, ArraySize >::force_
```

7.36.3.2 isVelDep

```
template<typename PairForce , typename Dtype , size_t ArraySize>
constexpr bool SpaceH::VelDepForce< PairForce, Dtype, ArraySize >::isVelDep {true}  [static]
```

7.36.3.3 this_acc_

```
template<typename PairForce , typename Dtype , size_t ArraySize>
VectorArray SpaceH::Force< PairForce, Dtype, ArraySize >::this_acc_
```

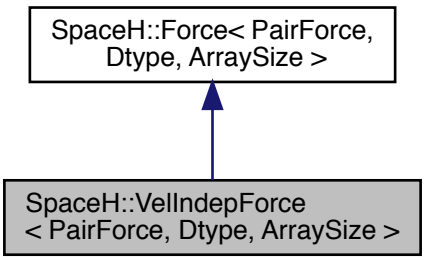
The documentation for this struct was generated from the following file:

- interaction/forces.h

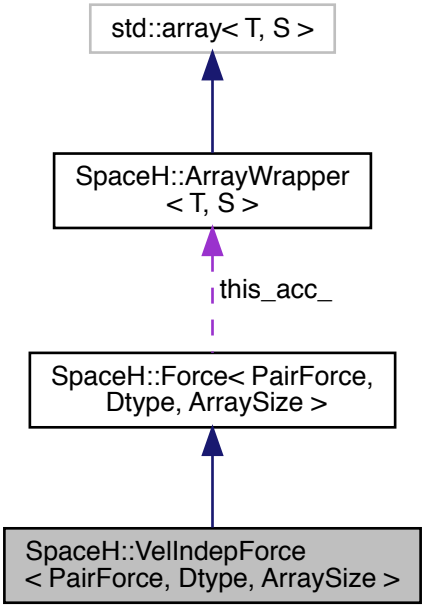
7.37 SpaceH::VelIndepForce< PairForce, Dtype, ArraySize > Struct Template Reference

```
#include <forces.h>
```

Inheritance diagram for SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >:



Collaboration diagram for SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >:



Public Types

- using Base = Force< PairForce, Dtype, ArraySize >
- using type = typename Base::type
- using Scalar = typename type::Scalar
- using Vector = typename type::Vector
- using ScalarArray = typename type::ScalarArray
- using VectorArray = typename type::VectorArray
- using IndexArray = typename type::IndexArray

Public Member Functions

- void calcuAcc (const ScalarArray &mass, const VectorArray &pos)
- void calcuAcc (const ScalarArray &mass, const VectorArray &pos, const VectorArray &chainPos, const IndexArray &chainIndex)

Public Attributes

- PairForce force_
- VectorArray this_acc_

Static Public Attributes

- static constexpr bool isVelDep {false}

Additional Inherited Members

7.37.1 Member Typedef Documentation

7.37.1.1 Base

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >::Base = Force<PairForce, Dtype,
ArraySize>
```

7.37.1.2 IndexArray

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >::IndexArray = typename type::IndexArray
```

7.37.1.3 Scalar

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >::Scalar = typename type::Scalar
```

7.37.1.4 ScalarArray

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >::ScalarArray = typename type::ScalarArray
```

7.37.1.5 type

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >::type = typename Base::type
```

7.37.1.6 Vector

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >::Vector = typename type::Vector
```

7.37.1.7 VectorArray

```
template<typename PairForce , typename Dtype , size_t ArraySize>
using SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >::VectorArray = typename type::VectorArray
```

7.37.2 Member Function Documentation

7.37.2.1 calcuAcc() [1/2]

```
template<typename PairForce , typename Dtype , size_t ArraySize>
void SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >::calcuAcc (
            const ScalarArray & mass,
            const VectorArray & pos )  [inline]
```

7.37.2.2 calcuAcc() [2/2]

```
template<typename PairForce , typename Dtype , size_t ArraySize>
void SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >::calcuAcc (
            const ScalarArray & mass,
            const VectorArray & pos,
            const VectorArray & chainPos,
            const IndexArray & chainIndex )  [inline]
```

7.37.3 Member Data Documentation

7.37.3.1 force_

```
template<typename PairForce , typename Dtype , size_t ArraySize>
PairForce SpaceH::Force< PairForce, Dtype, ArraySize >::force_
```

7.37.3.2 isVelDep

```
template<typename PairForce , typename Dtype , size_t ArraySize>
constexpr bool SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >::isVelDep {false}  [static]
```

7.37.3.3 this_acc_

```
template<typename PairForce , typename Dtype , size_t ArraySize>
VectorArray SpaceH::Force< PairForce, Dtype, ArraySize >::this_acc_
```

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

- interaction/forces.h

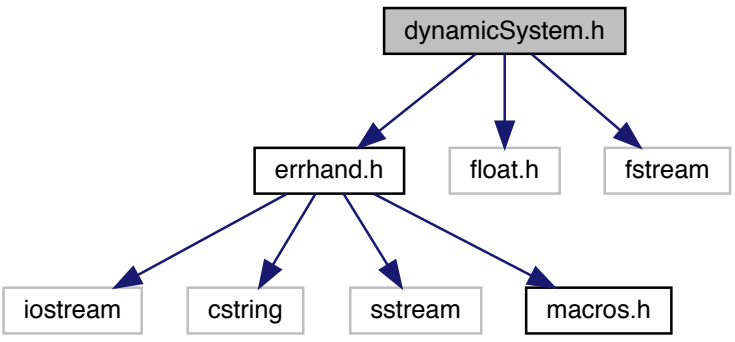
Chapter 8

File Documentation

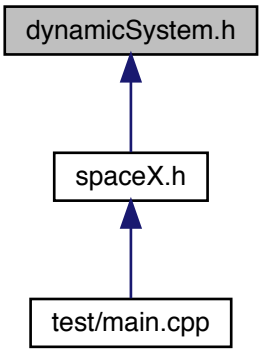
8.1 dynamicSystem.h File Reference

```
#include "errhand.h"
#include <float.h>
#include <fstream>
```

Include dependency graph for dynamicSystem.h:



This graph shows which files directly or indirectly include this file:



Classes

- class dynamicSystem< ParticSys, ODEiterator >

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

Typedefs

- template<typename ParticSys , template< typename > class Integrator, template< typename, typename > class ODEiterator>
  using spaceX = dynamicSystem< ParticSys, ODEiterator< ParticSys, Integrator< ParticSys > >>

  Alias of template name, linking the particle system, integrator and ODE iterator.

8.1.1 Typedef Documentation

8.1.1.1 spaceX

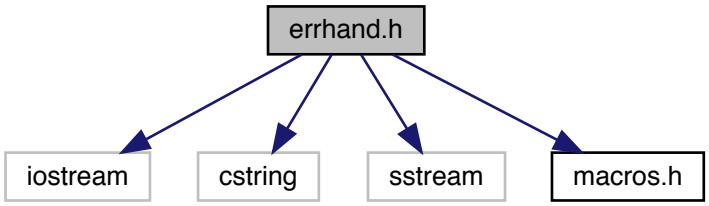
```
template<typename ParticSys , template< typename > class Integrator, template< typename,
typename > class ODEiterator>
using spaceX = dynamicSystem<ParticSys, ODEiterator<ParticSys, Integrator<ParticSys> >>
```

Alias of template name, linking the particle system, integrator and ODE iterator.

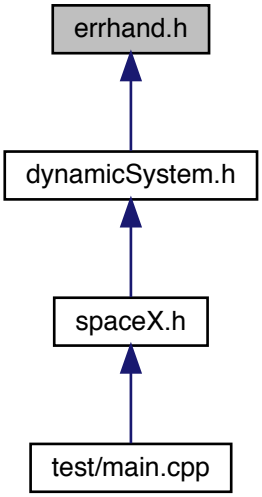
8.2 errhand.h File Reference

```
#include <iostream>
#include <cstring>
#include <sstream>
#include "macros.h"
```

Include dependency graph for errhand.h:



This graph shows which files directly or indirectly include this file:



Classes

- class errhand

Namespaces

- NOTICE

Macros

- #define ANSI_COLOR_RED "\x1b[31m"
- #define ANSI_COLOR_GREEN "\x1b[32m"
- #define ANSI_COLOR_YELLOW "\x1b[33m"
- #define ANSI_COLOR_BLUE "\x1b[34m"
- #define ANSI_COLOR_MAGENTA "\x1b[35m"
- #define ANSI_COLOR_CYAN "\x1b[36m"
- #define ANSI_COLOR_RESET "\x1b[0m"
- #define NEWLINE printf("\n");

Functions

- void NOTICE::Telegram (const char *host, const char *msg)
- void NOTICE::Title (const char *T)
- void NOTICE::SubTitle (const char *T)
- void NOTICE::EraseLine ()
- void NOTICE::Line ()
- void NOTICE::SubLine ()
- void NOTICE::RunInfo (double timeLimit, double outputsize_terval, double tolerance)

Variables

- constexpr size_t NOTICE::WIDTH = 80
- bool NOTICE::Message

8.2.1 Macro Definition Documentation

8.2.1.1 ANSI_COLOR_BLUE

```
#define ANSI_COLOR_BLUE "\x1b[34m"
```

8.2.1.2 ANSI_COLOR_CYAN

```
#define ANSI_COLOR_CYAN "\x1b[36m"
```

8.2.1.3 ANSI_COLOR_GREEN

```
#define ANSI_COLOR_GREEN "\x1b[32m"
```

8.2.1.4 ANSI_COLOR_MAGENTA

```
#define ANSI_COLOR_MAGENTA "\x1b[35m"
```

8.2.1.5 ANSI_COLOR_RED

```
#define ANSI_COLOR_RED "\x1b[31m"
```

8.2.1.6 ANSI_COLOR_RESET

```
#define ANSI_COLOR_RESET "\x1b[0m"
```

8.2.1.7 ANSI_COLOR_YELLOW

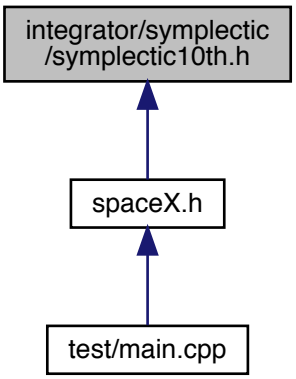
```
#define ANSI_COLOR_YELLOW "\x1b[33m"
```

8.2.1.8 NEWLINE

```
#define NEWLINE printf("\n");
```

8.3 integrator/symplectic/symplectic10th.h File Reference

This graph shows which files directly or indirectly include this file:



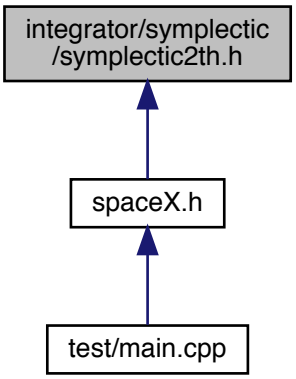
Classes

- class symplectic10th< ParticSys >

  Eighth order symplectic integrator.

8.4 integrator/symplectic/symplectic2th.h File Reference

This graph shows which files directly or indirectly include this file:



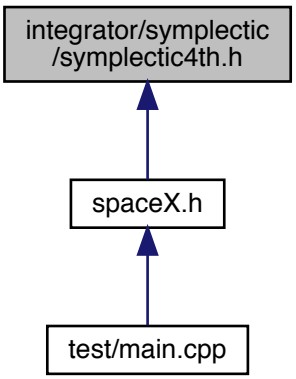
Classes

- class symplectic2th< ParticSys >

  Second order symplectic integrator.

8.5 integrator/symplectic/symplectic4th.h File Reference

This graph shows which files directly or indirectly include this file:



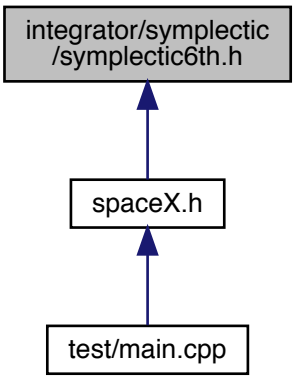
Classes

- class symplectic4th< ParticSys >

  Fourth order symplectic integrator.

8.6 integrator/symplectic/symplectic6th.h File Reference

This graph shows which files directly or indirectly include this file:



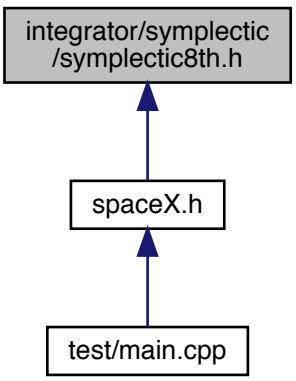
Classes

- class symplectic6th< ParticSys >

  Sixth order symplectic integrator.

8.7 integrator/symplectic/symplectic8th.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

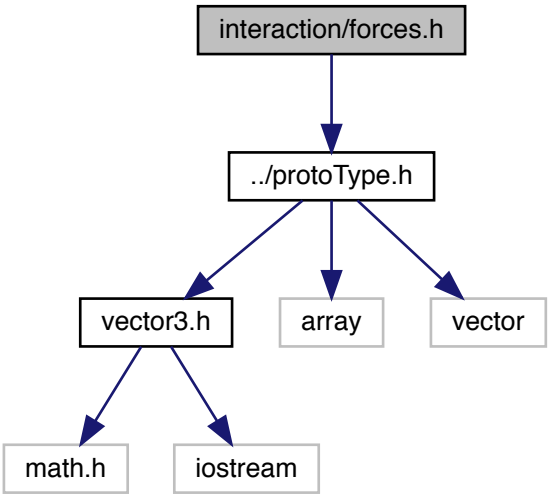
- class symplectic8th< ParticSys >

  Eighth order symplectic integrator.

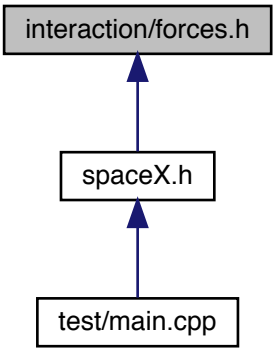
8.8 interaction/forces.h File Reference

```
#include "../protoType.h"
```

Include dependency graph for forces.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct SpaceH::EmptyForce< Dtype, ArraySize >
- struct SpaceH::Force< PairForce, Dtype, ArraySize >
- struct SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >
- struct SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize >
- struct SpaceH::VelDepForce< PairForce, Dtype, ArraySize >
- struct SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize >

- struct SpaceH::NewtonForce< Dtype, ArraySize >
- class SpaceH::PostNewtonian< Scalar >

  Post newtonian pair interaction functor(c++ std11)

Namespaces

- SpaceH

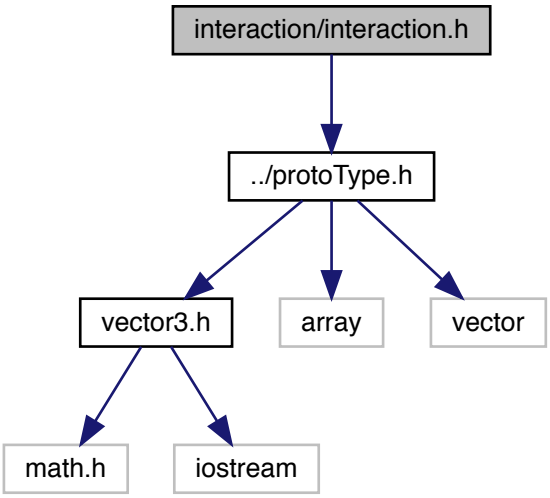
Variables

- constexpr double SpaceH::INV_C = 1 / C
- constexpr double SpaceH::INV_C2 = INV_C * INV_C
- constexpr double SpaceH::INV_C3 = INV_C2 * INV_C
- constexpr double SpaceH::INV_C4 = INV_C3 * INV_C
- constexpr double SpaceH::INV_C5 = INV_C4 * INV_C

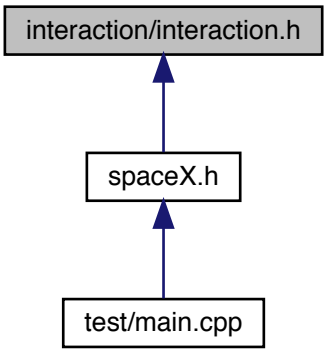
8.9 interaction/interaction.h File Reference

```
#include "../protoType.h"
```

Include dependency graph for interaction.h:



This graph shows which files directly or indirectly include this file:



Classes

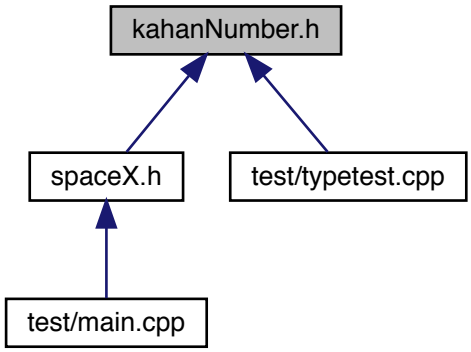
- class SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep >

Namespaces

- SpaceH

8.10 kahanNumber.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct SpaceH::kahan< T >

  Kahan number.

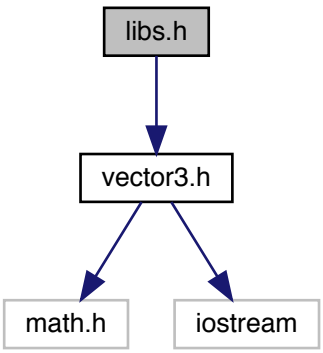
Namespaces

- SpaceH

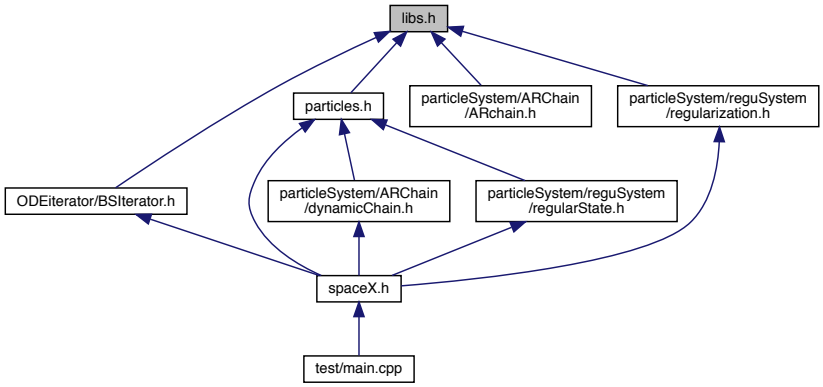
8.11 libs.h File Reference

```
#include "vector3.h"
```

Include dependency graph for libs.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- SpaceH

Functions

- template<typename T1 , typename T2 >
  const T2 SpaceH::min (const T1 &x, const T2 &y)

  Self min()

- template<typename T1 , typename T2 >
  const T2 SpaceH::max (const T1 &x, const T2 &y)

  Self max()

- template<class T >
  const T SpaceH::abs (const T &x)

  Self abs()

- template<typename Scalar1 , typename Scalar2 >
  void SpaceH::advanceScalar (Scalar1 &var, Scalar2 increase)

  Self swap()

- template<typename Scalar , typename Vector1 , typename Vector2 >
  void SpaceH::advanceVector (Vector1 &var, const Vector2 &increase, Scalar stepSize)

- template<typename ScalarArray , typename VectorArray >
  void SpaceH::moveToCMCoord (const ScalarArray &mass, VectorArray &phyVar)

  Move variables to central mass coordinates.

- template<typename Scalar , size_t N>
  double SpaceH::getKineticEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &vel)

  Calculate the kinetic energy of particles.

- template<typename Scalar , size_t N>
  double SpaceH::getPotentialEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &pos)

  Calculate the potential energy of particles.

- template<typename Scalar , size_t N>
  double SpaceH::getTotalEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &pos, const std::array< vec3< Scalar >, N > &vel)

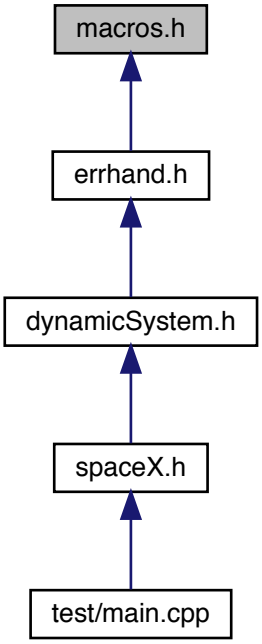
  Calculate the total(potential + kinetic) energy of particles.

- template<typename T >
  void SpaceH::print (T &var)

  print an array. Used for debug

8.12 macros.h File Reference

This graph shows which files directly or indirectly include this file:



Variables

- constexpr double PI = 3.14159265358979323
- constexpr double AU = (PI / 648000)
- constexpr double PC = 1
- constexpr double M_SOLAR = 1
- constexpr double M_JUPITER = 0.9547919E-3
- constexpr double R_SOLAR = 2.25461E-8
- constexpr double YEAR = 6.694685210039141E-08
- constexpr double DAY = YEAR / 365.25636042
- constexpr double G = 1
- constexpr double V_UNIT = 6.54589713446219E-2
- constexpr double C = 299792.458 / V_UNIT
- constexpr double KM = 3.2407557442395564e-14

8.12.1 Variable Documentation

8.12.1.1 AU

```
constexpr double AU = (PI / 648000)
```

8.12.1.2 C

```
constexpr double C = 299792.458 / V_UNIT
```

8.12.1.3 DAY

```
constexpr double DAY = YEAR / 365.25636042
```

8.12.1.4 G

```
constexpr double G = 1
```

8.12.1.5 KM

```
constexpr double KM = 3.2407557442395564e-14
```

8.12.1.6 M_JUPITER

```
constexpr double M_JUPITER = 0.9547919E-3
```

8.12.1.7 M_SOLAR

```
constexpr double M_SOLAR = 1
```

8.12.1.8 PC

```
constexpr double PC = 1
```

8.12.1.9 PI

```
constexpr double PI = 3.14159265358979323
```

8.12.1.10 R_SOLAR

```
constexpr double R_SOLAR = 2.25461E-8
```

8.12.1.11 V_UNIT

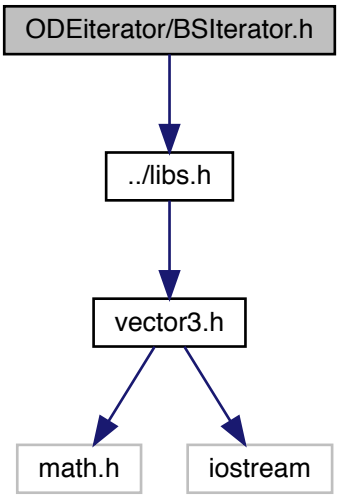
```
constexpr double V_UNIT = 6.54589713446219E-2
```

8.12.1.12 YEAR

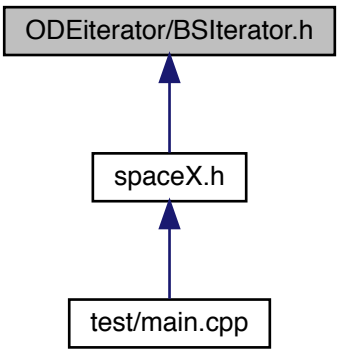
```
constexpr double YEAR = 6.694685210039141E-08
```

8.13 ODEiterator/BSIterator.h File Reference

```
#include "../libs.h"
```
Include dependency graph for BSIterator.h:



This graph shows which files directly or indirectly include this file:



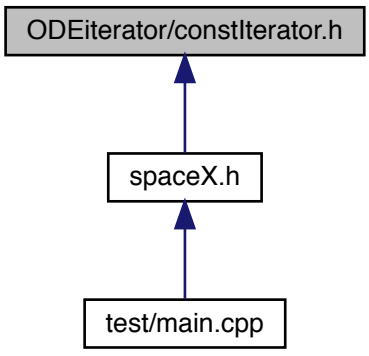
Classes

- class BSIterator< ParticSys, Integrator >

  Bulirsch-Stoer extrapolation algorithm.

8.14 ODEiterator/constIterator.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class constIterator< ParticSys, Integrator >

  Most common iterator.

8.15 ODEiterator/dichotomy.h File Reference

Classes

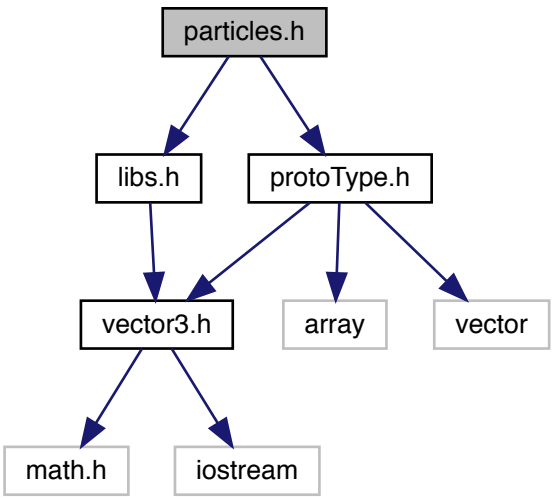
- class dichoIterator< ParticSys, Integrator >

  Dichotomy iterator.

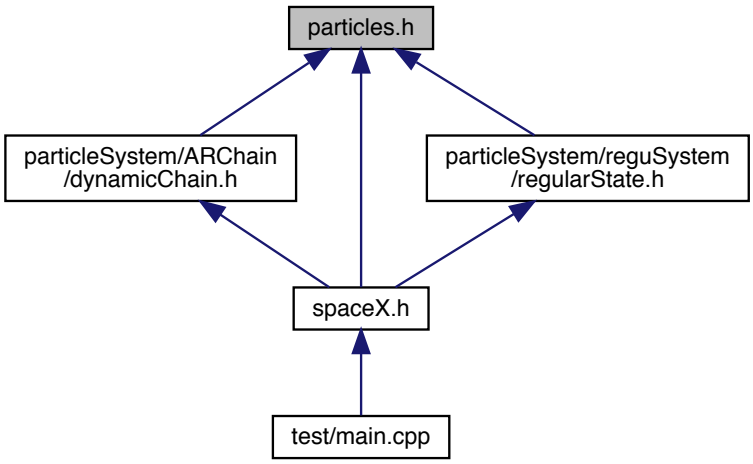
8.16 particles.h File Reference

```
#include "protoType.h"
#include "libs.h"
```

Include dependency graph for particles.h:



This graph shows which files directly or indirectly include this file:



Classes

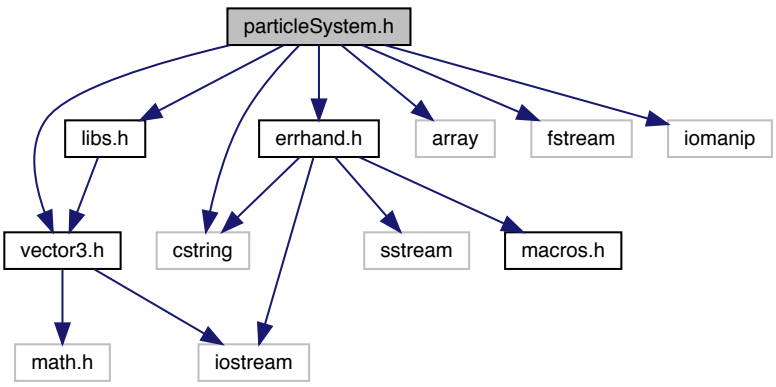
- class particles< Dtype, ArraySize >

  Class of dynamical variable.

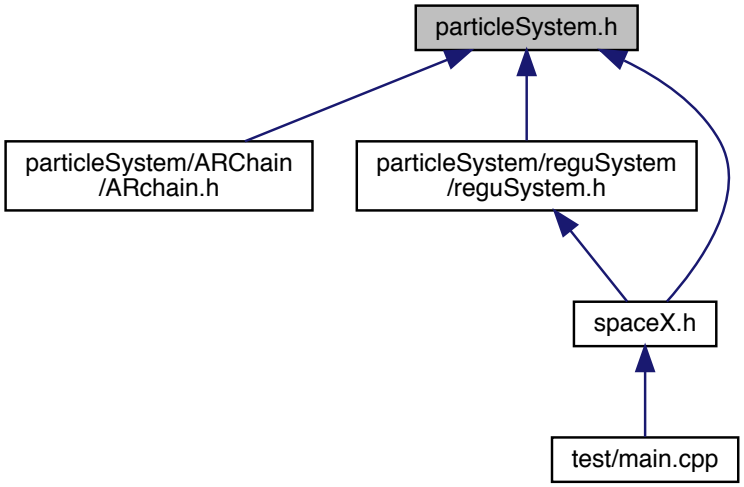
8.17 particleSystem.h File Reference

```
#include <fstream>
#include <cstring>
#include <iomanip>
```

Include dependency graph for particleSystem.h:



This graph shows which files directly or indirectly include this file:



Classes

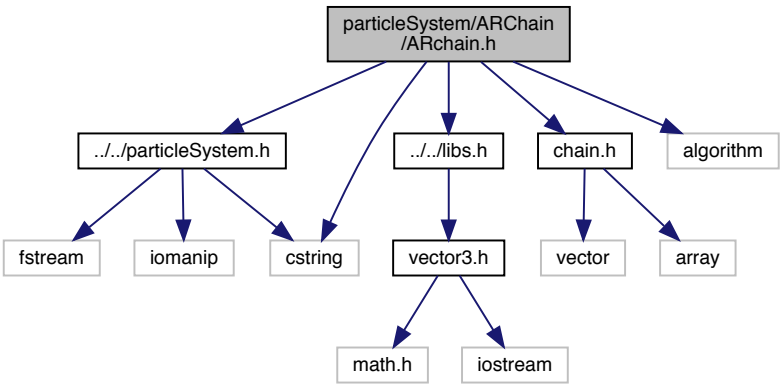
- class particleSystem< Particles, Interaction >

  Base class of particle System.

8.18 particleSystem/ARChain/ARchain.h File Reference

```
#include "../../particleSystem.h"
#include "../../libs.h"
#include "chain.h"
#include <cstring>
#include <algorithm>
```

Include dependency graph for ARchain.h:



Classes

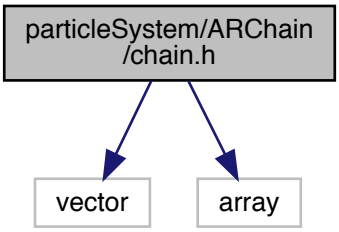
- class ARchain< Particles, Interaction, Regularitor >

  Algorithmatic Regularization chain System.

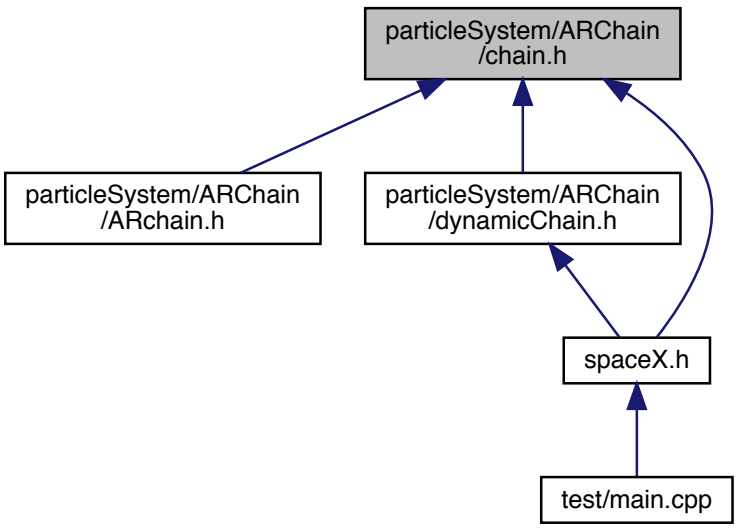
8.19 particleSystem/ARChain/chain.h File Reference

```
#include <vector>
#include <array>
```

Include dependency graph for chain.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct SpaceH::chain::Node< Scalar >

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

Namespaces

- SpaceH
- SpaceH::chain

Functions

- template<typename VectorArray , typename IndexArray >
  void SpaceH::chain::getChainIndex (const VectorArray &pos, IndexArray &chainIndex)

  Calculate the mapping index from Cartesian coordinate to chain coordinate.
- template<typename VectorArray , typename NodeArray >
  void SpaceH::chain::createAdjMartix (const VectorArray &pos, NodeArray &AdjMatrix)

  Create the adjoint matrix for particle pairs.
- template<typename NodeArray , typename IndexArray >
  void SpaceH::chain::createChainIndex (NodeArray &AdjMatrix, IndexArray &chainIndex)

  Create mapping index from adjoint matrix.
- template<typename IndexArray >
  bool SpaceH::chain::IsDiff (const IndexArray &Index1, const IndexArray &Index2)

  Check if two mapping indexes are the same.
- template<typename VectorArray , typename IndexArray >
  void SpaceH::chain::updateChain (VectorArray &pos, IndexArray &chainIndex, IndexArray &newIndex)

  Update the position chain.
- template<typename VectorArray , typename IndexArray >
  void SpaceH::chain::synChain (VectorArray &data, VectorArray &chainData, IndexArray &chainIndex)

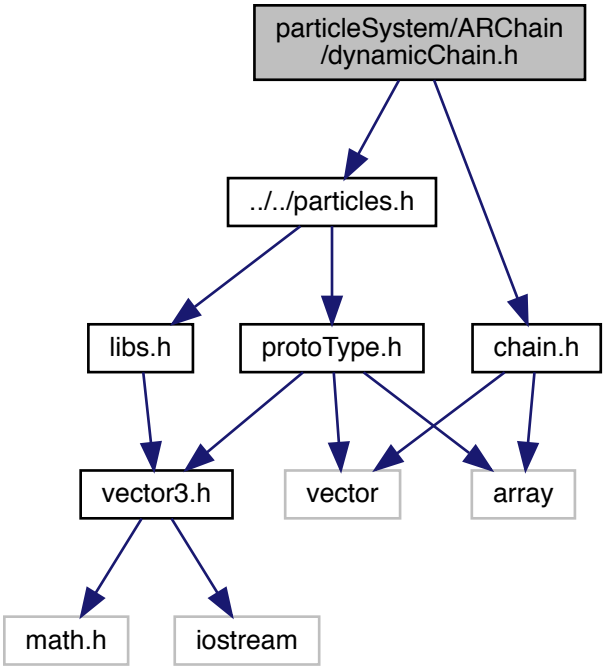
  Calulate the chain data from Cartesian data and chain index mapping.
- template<typename VectorArray , typename IndexArray >
  void SpaceH::chain::synCartesian (VectorArray &chainData, VectorArray &data, IndexArray &chainIndex)

  Calulate the Cartesian data from chain data and chain index mapping.

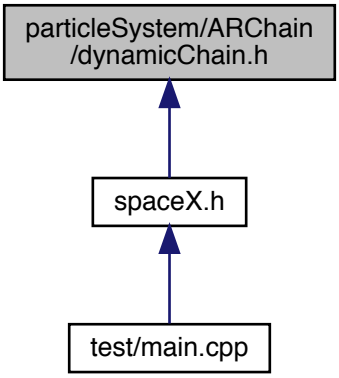
8.20 particleSystem/ARChain/dynamicChain.h File Reference

```
#include "../../particles.h"
#include "chain.h"
```

Include dependency graph for dynamicChain.h:



This graph shows which files directly or indirectly include this file:



Classes

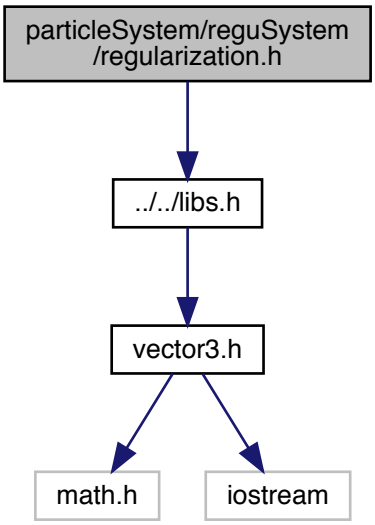
- class ChainParticles< Dtype, ArraySize >

  Class of dynamical variable.

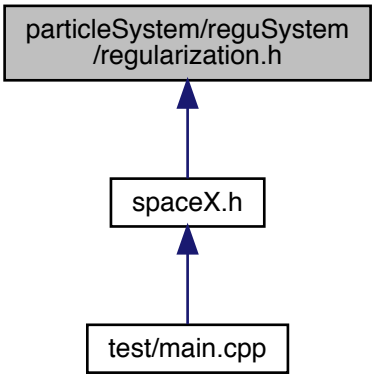
8.21 particleSystem/reguSystem/regularization.h File Reference

```
#include "../../libs.h"
```

Include dependency graph for regularization.h:



This graph shows which files directly or indirectly include this file:



Classes

- class logH< Particles >

logH extention algorithmatic regularization interface

- class TTL< Particles >

  Time Transform Leapfrog algorithmatic regularization interface.

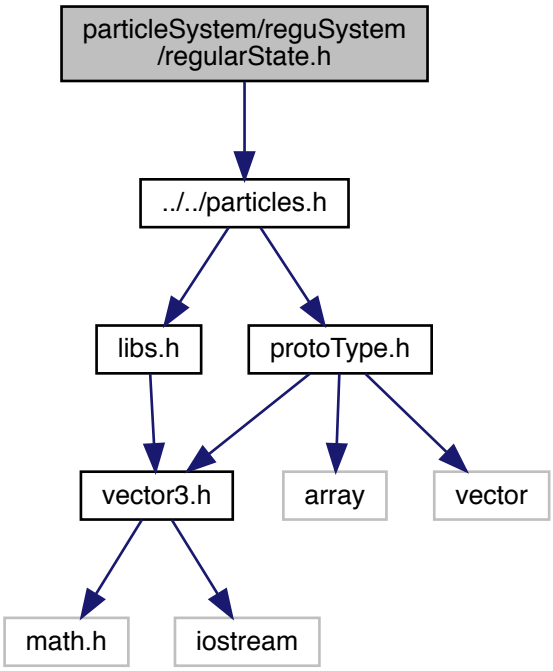
- class NoRegu< Particles >

  Ordinary algorithmatic regularization interface.

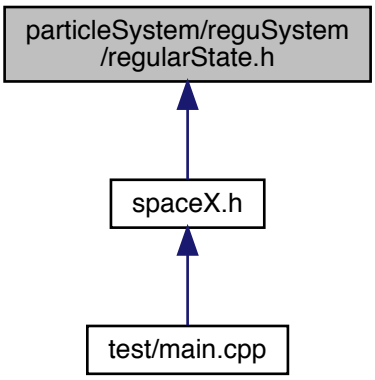
8.22 particleSystem/reguSystem/regularState.h File Reference

```
#include "../../particles.h"
```

Include dependency graph for regularState.h:



This graph shows which files directly or indirectly include this file:



Classes

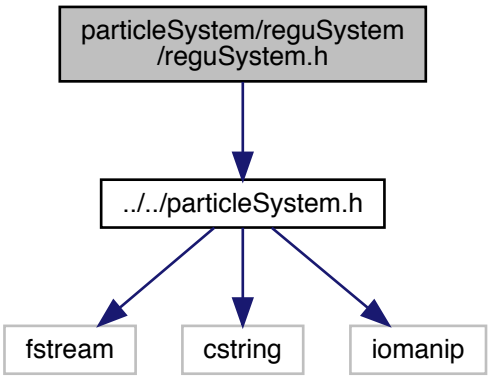
- class ReguParticles< Dtype, ArraySize >

  Class of dynamical system with regularization variables.

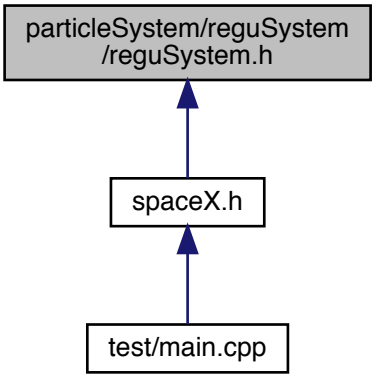
8.23 particleSystem/reguSystem/reguSystem.h File Reference

```
#include "../../particleSystem.h"
```

Include dependency graph for reguSystem.h:



This graph shows which files directly or indirectly include this file:



Classes

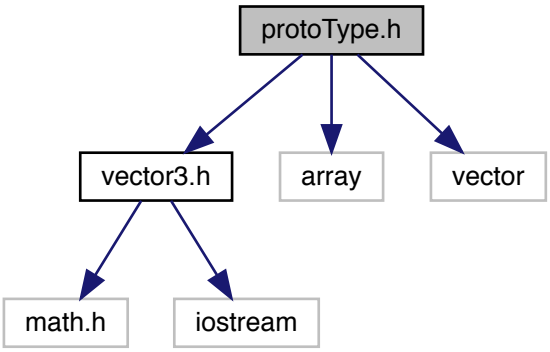
- class ReguSystem< Particles, Interaction, Regularitor >

  Regularized particle System.

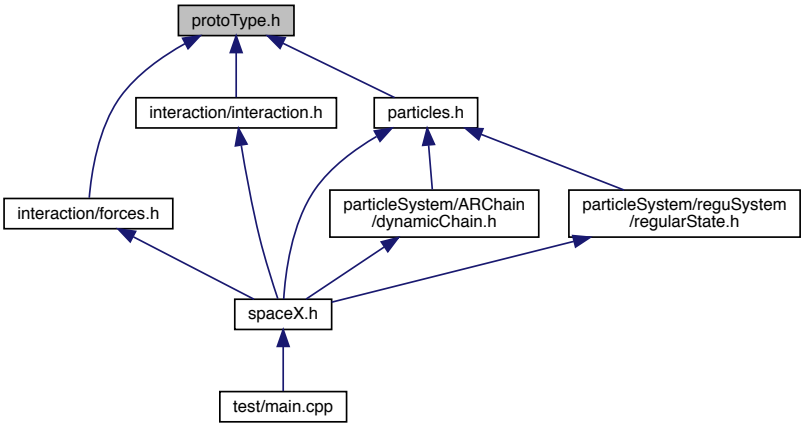
8.24 protoType.h File Reference

```
#include "vector3.h"
#include <array>
#include <vector>
```

Include dependency graph for protoType.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct SpaceH::ArrayWrapper< T, S >
- struct SpaceH::ArrayWrapper< T, DYNAMICAL >
- struct SpaceH::get_value_type< T >
- struct SpaceH::ProtoType< Dtype, Size >

Namespaces

- SpaceH

Enumerations

- enum SpaceH::PARTICTYPE {
  SpaceH::NEUTRONSTAR, SpaceH::STAR, SpaceH::BLACKHOLE, SpaceH::POINT,
  SpaceH::NONE = 0 }
- enum SpaceH::EVENTTYPE {
  SpaceH::TDE, SpaceH::MERGE, SpaceH::ESCAPE, SpaceH::DISRUPTED,
  SpaceH::UNEVENTFUL, SpaceH::HVS }
- enum SpaceH::INTEGRATORTYPE {
  SpaceH::INTEGRATORTYPE::DKDLEAPFROG, SpaceH::INTEGRATORTYPE::KDKLEAPFROG, SpaceH::INTEGRATORTY
  SpaceH::INTEGRATORTYPE::PEFRL,
  SpaceH::INTEGRATORTYPE::SYM6, SpaceH::INTEGRATORTYPE::SYM8, SpaceH::INTEGRATORTYPE::SYM10
  }
- enum SpaceH::SYSTEMTYPE { SpaceH::SYSTEMTYPE::PLAIN, SpaceH::SYSTEMTYPE::CHAIN }
- enum SpaceH::REGUTYPE { SpaceH::REGUTYPE::LOGH, SpaceH::REGUTYPE::TTL, SpaceH::REGUTYPE::NONE
  }
- enum SpaceH::ITERTYPE { SpaceH::ITERTYPE::BSITER, SpaceH::ITERTYPE::SEQITER }
- enum SpaceH::DATASTRUCT { SpaceH::DATASTRUCT::PLAIN =0, SpaceH::DATASTRUCT::CHAIN }

Variables

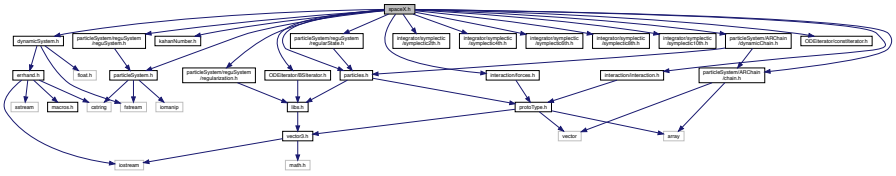
- constexpr size_t SpaceH::DYNAMICAL = 0

8.25 README.md File Reference

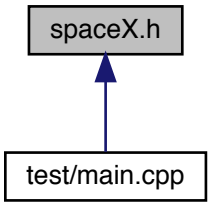
8.26 spaceX.h File Reference

```
#include "dynamicSystem.h"
#include "particles.h"
#include "kahanNumber.h"
#include "particleSystem.h"
#include "particleSystem/reguSystem/regularization.h"
#include "particleSystem/reguSystem/reguSystem.h"
#include "particleSystem/reguSystem/regularState.h"
#include "particleSystem/ARChain/chain.h"
#include "particleSystem/ARChain/dynamicChain.h"
#include "integrator/symplectic/symplectic2th.h"
#include "integrator/symplectic/symplectic4th.h"
#include "integrator/symplectic/symplectic6th.h"
#include "integrator/symplectic/symplectic8th.h"
#include "integrator/symplectic/symplectic10th.h"
#include "ODEiterator/BSIterator.h"
#include "ODEiterator/constIterator.h"
#include "interaction/interaction.h"
#include "interaction/forces.h"
```

Include dependency graph for spaceX.h:



This graph shows which files directly or indirectly include this file:

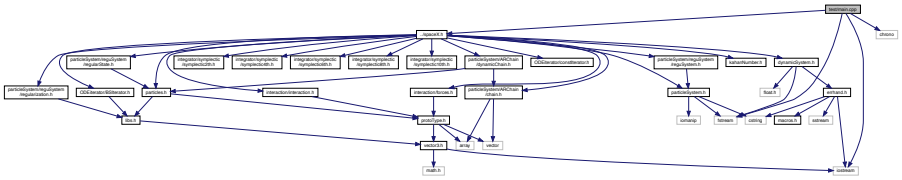


8.27 test/main.cpp File Reference

```
#include "../spaceX.h"
#include <chrono>
```

```
#include <fstream>
#include <iostream>
```

Include dependency graph for main.cpp:



Typedefs

- typedef std::chrono::time_point< std::chrono::high_resolution_clock > resolutionClock
- using scalar = double

Functions

- int main (int argc, char **argv)

Variables

- const size_t N = 3

8.27.1 Typedef Documentation

8.27.1.1 resolutionClock

```
typedef std::chrono::time_point<std::chrono::high_resolution_clock> resolutionClock
```

8.27.1.2 scalar

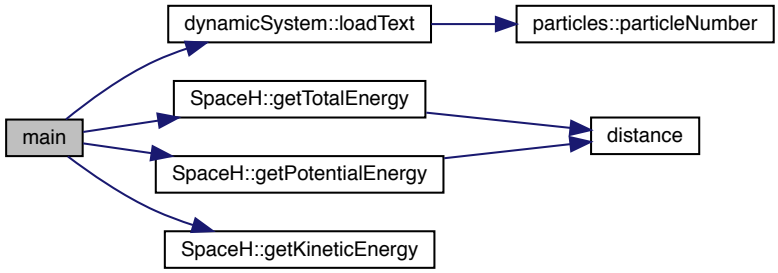
```
using scalar = double
```

8.27.2 Function Documentation

8.27.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

Here is the call graph for this function:



8.27.3 Variable Documentation

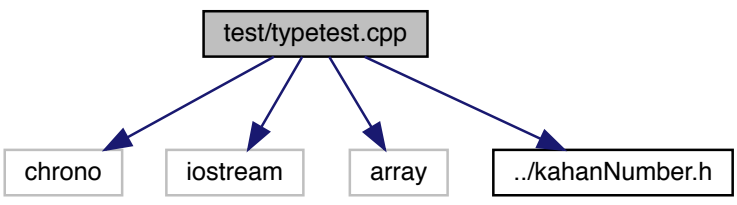
8.27.3.1 N

```
const size_t N = 3
```

8.28 test/typetest.cpp File Reference

```
#include <chrono>
#include <iostream>
#include <array>
#include "../kahanNumber.h"
```

Include dependency graph for typetest.cpp:



Classes

- struct dtype< T >

Typedefs

- typedef std::chrono::time_point< std::chrono::high_resolution_clock > resolutionClock

Functions

- int main ()

8.28.1 Typedef Documentation

8.28.1.1 resolutionClock

```
typedef std::chrono::time_point<std::chrono::high_resolution_clock> resolutionClock
```

8.28.2 Function Documentation

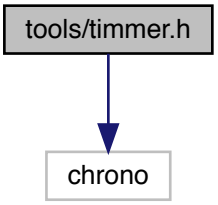
8.28.2.1 main()

```
int main ( )
```

8.29 tools/timmer.h File Reference

```
#include <chrono>
```

Include dependency graph for timmer.h:



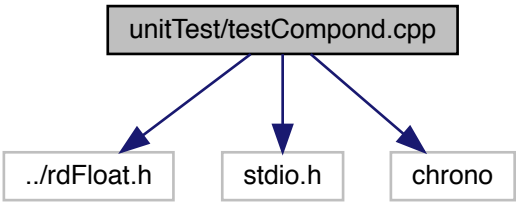
Classes

- class ProgressBar

8.30 unitTest/testCompond.cpp File Reference

```
#include "../rdFloat.h"
#include <stdio.h>
#include <chrono>
```

Include dependency graph for testCompond.cpp:



Typedefs

- typedef std::chrono::time_point< std::chrono::high_resolution_clock > resolutionClock

Functions

- int main (int argc, char **argv)

8.30.1 Typedef Documentation

8.30.1.1 resolutionClock

```
typedef std::chrono::time_point<std::chrono::high_resolution_clock> resolutionClock
```

8.30.2 Function Documentation

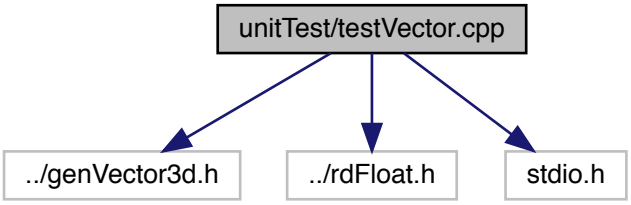
8.30.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

8.31 unitTest/testVector.cpp File Reference

```
#include "../genVector3d.h"
#include "../rdFloat.h"
#include <stdio.h>
```

Include dependency graph for testVector.cpp:



Functions

- int main (int argc, char **argv)

8.31.1 Function Documentation

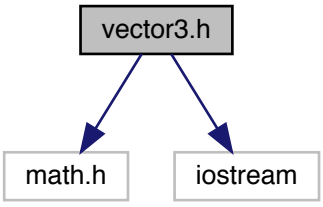
8.31.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

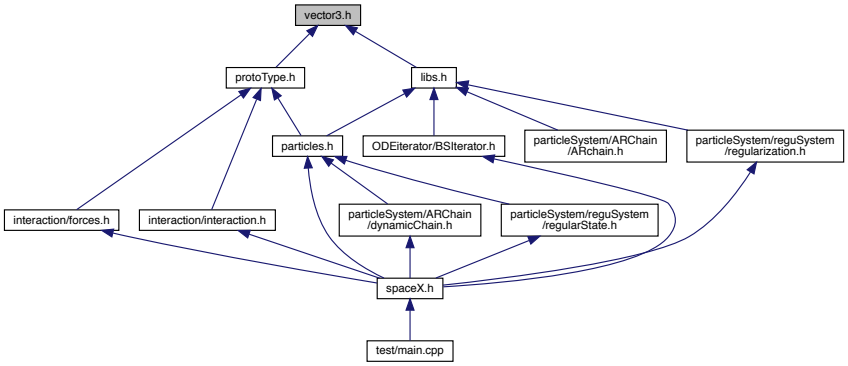
8.32 vector3.h File Reference

```
#include <math.h>
#include <iostream>
```

Include dependency graph for vector3.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct vec3< T >

  Self 3D vector class.

Typedefs

- typedef vec3< double > vec3d
- typedef vec3< float > vec3f
- typedef vec3< int > vec3i
- typedef vec3< char > vec3c

Functions

- template<typename T >
  T distance (const vec3< T > &v1, const vec3< T > &v2)

  Calculate the Euclid distance of two vectors.

8.32.1 Typedef Documentation

8.32.1.1 vec3c

```
typedef vec3<char> vec3c
```

8.32.1.2 vec3d

```
typedef vec3<double> vec3d
```

8.32.1.3 vec3f

```
typedef vec3<float> vec3f
```

8.32.1.4 vec3i

```
typedef vec3<int> vec3i
```

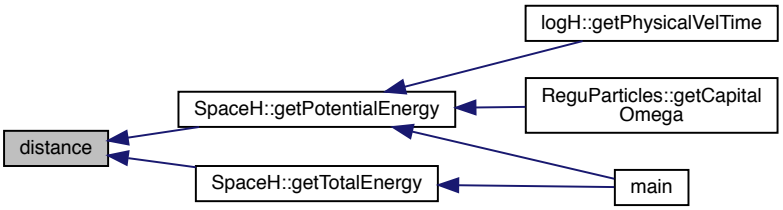
8.32.2 Function Documentation

8.32.2.1 distance()

```
template<typename T >
T distance (
             const vec3< T > & v1,
             const vec3< T > & v2 )  [inline]
```

Calculate the Euclid distance of two vectors.

Here is the caller graph for this function:



Index