

spacex

Generated by Doxygen 1.8.14

Contents

1	Template-SpaceX	2
2	Namespace Index	2
2.1	Namespace List	2
3	Hierarchical Index	2
3.1	Class Hierarchy	2
4	Class Index	3
4.1	Class List	3
5	File Index	4
5.1	File List	4
6	Namespace Documentation	5
6.1	chain Namespace Reference	5
6.1.1	Typedef Documentation	6
6.1.2	Function Documentation	7
6.2	NOTICE Namespace Reference	13
6.2.1	Function Documentation	13
6.2.2	Variable Documentation	14
7	Class Documentation	15
7.1	ARchain< Interaction, EvolvedData, Regularitor > Class Template Reference	15
7.1.1	Detailed Description	17
7.1.2	Member Typedef Documentation	17
7.1.3	Member Function Documentation	18
7.1.4	Member Data Documentation	25
7.2	ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor > Class Template Reference	26
7.2.1	Detailed Description	28
7.2.2	Member Typedef Documentation	28
7.2.3	Member Function Documentation	29

7.2.4	Member Data Documentation	34
7.3	BSIterator< ParticSys, Integrator > Class Template Reference	35
7.3.1	Detailed Description	37
7.3.2	Member Typedef Documentation	37
7.3.3	Constructor & Destructor Documentation	37
7.3.4	Member Function Documentation	37
7.3.5	Member Data Documentation	41
7.4	constIterator< ParticSys, Integrator > Class Template Reference	44
7.4.1	Detailed Description	44
7.4.2	Member Typedef Documentation	44
7.4.3	Member Function Documentation	45
7.5	dynamics< DataType, N > Class Template Reference	45
7.5.1	Detailed Description	46
7.5.2	Member Typedef Documentation	46
7.5.3	Member Function Documentation	47
7.5.4	Member Data Documentation	48
7.6	dynamicSystem< ParticSys, Integrator, ODEiterator > Class Template Reference	49
7.6.1	Detailed Description	50
7.6.2	Constructor & Destructor Documentation	50
7.6.3	Member Function Documentation	50
7.6.4	Member Data Documentation	52
7.7	errhand Class Reference	53
7.7.1	Constructor & Destructor Documentation	53
7.7.2	Member Function Documentation	53
7.7.3	Member Data Documentation	56
7.8	GAR< DataType, N > Class Template Reference	57
7.8.1	Detailed Description	58
7.8.2	Member Typedef Documentation	58
7.8.3	Member Function Documentation	58
7.8.4	Member Data Documentation	63

7.9	logH< DynamicState > Class Template Reference	64
7.9.1	Detailed Description	64
7.9.2	Member Typedef Documentation	65
7.9.3	Member Function Documentation	65
7.10	Newtonian< Scalar > Class Template Reference	67
7.10.1	Detailed Description	67
7.10.2	Member Function Documentation	68
7.11	chain::Node< Scalar > Struct Template Reference	68
7.11.1	Detailed Description	68
7.11.2	Member Data Documentation	68
7.12	NoRegu< DynamicState > Class Template Reference	69
7.12.1	Detailed Description	69
7.12.2	Member Typedef Documentation	70
7.12.3	Member Function Documentation	70
7.13	particleSystem< Derived, EvolvedData > Class Template Reference	71
7.13.1	Detailed Description	72
7.13.2	Member Typedef Documentation	73
7.13.3	Constructor & Destructor Documentation	73
7.13.4	Member Function Documentation	74
7.13.5	Member Data Documentation	76
7.14	PN1th< Scalar > Class Template Reference	77
7.14.1	Detailed Description	78
7.14.2	Member Typedef Documentation	78
7.14.3	Member Function Documentation	78
7.15	reguDynamics< DataType, N > Class Template Reference	79
7.15.1	Detailed Description	80
7.15.2	Member Typedef Documentation	80
7.15.3	Member Function Documentation	81
7.15.4	Member Data Documentation	85
7.16	reguSystem< Interaction, EvolvedData, Regularitor > Class Template Reference	86

7.16.1 Detailed Description	88
7.16.2 Member Typedef Documentation	88
7.16.3 Member Function Documentation	89
7.16.4 Member Data Documentation	94
7.17 reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor > Class Template Reference	95
7.17.1 Detailed Description	96
7.17.2 Member Typedef Documentation	97
7.17.3 Member Function Documentation	97
7.17.4 Member Data Documentation	101
7.18 symplectic10th< ParticSys > Class Template Reference	102
7.18.1 Detailed Description	102
7.18.2 Member Function Documentation	102
7.18.3 Member Data Documentation	103
7.19 symplectic2th< ParticSys > Class Template Reference	103
7.19.1 Detailed Description	103
7.19.2 Member Typedef Documentation	104
7.19.3 Member Function Documentation	104
7.19.4 Member Data Documentation	104
7.20 symplectic4th< ParticSys > Class Template Reference	104
7.20.1 Detailed Description	105
7.20.2 Member Function Documentation	105
7.20.3 Member Data Documentation	105
7.21 symplectic6th< ParticSys > Class Template Reference	106
7.21.1 Detailed Description	106
7.21.2 Member Function Documentation	106
7.21.3 Member Data Documentation	106
7.22 symplectic8th< ParticSys > Class Template Reference	107
7.22.1 Detailed Description	107
7.22.2 Member Function Documentation	107
7.22.3 Member Data Documentation	108
7.23 TTL< DynamicState > Class Template Reference	108
7.23.1 Detailed Description	108
7.23.2 Member Typedef Documentation	109
7.23.3 Member Function Documentation	109
7.24 vec3< T > Struct Template Reference	110
7.24.1 Detailed Description	111
7.24.2 Constructor & Destructor Documentation	111
7.24.3 Member Function Documentation	112
7.24.4 Friends And Related Function Documentation	118
7.24.5 Member Data Documentation	119

8 File Documentation	120
8.1 dynamicState.h File Reference	120
8.2 dynamicSystem.h File Reference	121
8.2.1 Typedef Documentation	122
8.3 errhand.h File Reference	123
8.3.1 Macro Definition Documentation	124
8.4 integrator/symplectic/symplectic10th.h File Reference	125
8.5 integrator/symplectic/symplectic2th.h File Reference	125
8.6 integrator/symplectic/symplectic4th.h File Reference	126
8.7 integrator/symplectic/symplectic6th.h File Reference	126
8.8 integrator/symplectic/symplectic8th.h File Reference	126
8.9 interaction/interaction.h File Reference	126
8.9.1 Variable Documentation	127
8.10 libs.h File Reference	128
8.10.1 Function Documentation	129
8.11 macros.h File Reference	139
8.11.1 Enumeration Type Documentation	140
8.11.2 Variable Documentation	141
8.12 ODEiterator/BSIterator.h File Reference	143
8.13 ODEiterator/constIterator.h File Reference	144
8.14 particleSystem.h File Reference	145
8.14.1 Function Documentation	146
8.15 particleSystem/ARchain.h File Reference	146
8.16 particleSystem/chain.h File Reference	147
8.17 particleSystem/GAR.h File Reference	149
8.18 particleSystem/regularization.h File Reference	150
8.19 particleSystem/regularState.h File Reference	151
8.20 particleSystem/reguSystem.h File Reference	153
8.21 README.md File Reference	154
8.22 spaceX.h File Reference	154
8.22.1 Typedef Documentation	155
8.23 test/main.cpp File Reference	156
8.23.1 Typedef Documentation	156
8.23.2 Function Documentation	156
8.24 unitTest/testCompond.cpp File Reference	157
8.24.1 Typedef Documentation	158
8.24.2 Function Documentation	158
8.25 unitTest/testVector.cpp File Reference	158
8.25.1 Function Documentation	158
8.26 vector3.h File Reference	159
8.26.1 Typedef Documentation	160
8.26.2 Function Documentation	160

Index	163
-----------------------	-----

1 Template-SpaceX

2 Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

chain	5
NOTICE	13

3 Hierarchical Index

3.1 Class Hierarchy

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

BSIterator< ParticSys, Integrator >	35
constIterator< ParticSys, Integrator >	44
dynamics< DataType, N >	45
dynamicSystem< ParticSys, Integrator, ODEiterator >	49
errhand	53
GAR< DataType, N >	57
logH< DynamicState >	64
Newtonian< Scalar >	67
chain::Node< Scalar >	68
NoRegu< DynamicState >	69
particleSystem< Derived, EvolvedData >	71
particleSystem< ARchain< Interaction, EvolvedData, Regularitor >, EvolvedData >	71
 ARchain< Interaction, EvolvedData, Regularitor >	15
particleSystem< ARchain< Newtonian< EvolvedData::Scalar >, EvolvedData, Regularitor >, EvolvedData >	71
 ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >	26
particleSystem< reguSystem< Interaction, EvolvedData, Regularitor >, EvolvedData >	71

reguSystem < Interaction , EvolvedData , Regularitor >	86
particleSystem < reguSystem < Newtonian < EvolvedData::Scalar >, EvolvedData , Regularitor >, EvolvedData >	71
reguSystem < Newtonian < typename EvolvedData::Scalar >, EvolvedData , Regularitor >	95
PN1th < Scalar >	77
reguDynamics < DataType , N >	79
symplectic10th < ParticSys >	102
symplectic2th < ParticSys >	103
symplectic4th < ParticSys >	104
symplectic6th < ParticSys >	106
symplectic8th < ParticSys >	107
TTL < DynamicState >	108
vec3 < T >	110

4 Class Index

4.1 Class List

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

ARchain < Interaction , EvolvedData , Regularitor > Algorithmatic Regularization chain System	15
ARchain < Newtonian < typename EvolvedData::Scalar >, EvolvedData , Regularitor > Partial specilization of template ARchain	26
BSIterator < ParticSys , Integrator > Bulirsch-Stoer extrapolation algorithm	35
constIterator < ParticSys , Integrator > Most common iterator	44
dynamics < DataType , N > Class of dynamical variable	45
dynamicSystem < ParticSys , Integrator , ODEiterator > A wrapper to make particle system, integrator and ODE iterator work together	49
errhand	53
GAR < DataType , N > Class of velocity dependent dynamical system with regularization variables	57
logH < DynamicState > LogH extention algorithmatic regularization interface	64
Newtonian < Scalar > Marker of None velocity dependent force functor(c++ std11)	67

chain::Node< Scalar >	
Struture to store the relative distance and index of two particles	68
NoRegu< DynamicState >	
Ordinary algorithmatic regularization interface	69
particleSystem< Derived, EvolvedData >	
Base class of particle System	71
PN1th< Scalar >	
Post newtonian pair interaction functor(c++ std11)	77
reguDynamics< DataType, N >	
Class of dynamical system with regularization variables	79
reguSystem< Interaction, EvolvedData, Regularitor >	
Regularized particle System	86
reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >	
Partial specialization of reguSystem for velocity independent system	95
symplectic10th< ParticSys >	
Tenth order symplectic integrator	102
symplectic2th< ParticSys >	
Second order symplectic integrator	103
symplectic4th< ParticSys >	
Fourth order symplectic integrator	104
symplectic6th< ParticSys >	
Sixth order symplectic integrator	106
symplectic8th< ParticSys >	
Eighth order symplectic integrator	107
TTL< DynamicState >	
Time Transform Leapfrog algorithmatic regularization interface	108
vec3< T >	
Self 3D vector class	110

5 File Index

5.1 File List

Here is a list of all files with brief descriptions:

dynamicState.h	120
dynamicSystem.h	121
errhand.h	123
libs.h	128
macros.h	139

particleSystem.h	145
spaceX.h	154
vector3.h	159
integrator/symplectic/symplectic10th.h	125
integrator/symplectic/symplectic2th.h	125
integrator/symplectic/symplectic4th.h	126
integrator/symplectic/symplectic6th.h	126
integrator/symplectic/symplectic8th.h	126
interaction/interaction.h	126
ODEiterator/BSIterator.h	143
ODEiterator/constIterator.h	144
particleSystem/ARchain.h	146
particleSystem/chain.h	147
particleSystem/GAR.h	149
particleSystem/regularization.h	150
particleSystem/regularState.h	151
particleSystem/reguSystem.h	153
test/main.cpp	156
unitTest/testCompond.cpp	157
unitTest/testVector.cpp	158

6 Namespace Documentation

6.1 chain Namespace Reference

Classes

- struct [Node](#)
Struture to store the relative distance and index of two particles.

Typedefs

- `template<typename Scalar , size_t N>`
using [VectorArray](#) = `std::array< vec3< Scalar >, N >`
- `template<size_t N>`
using [IndexArray](#) = `std::array< size_t, N >`
- `template<typename Scalar , size_t N>`
using [NodeArray](#) = `std::array< Node< Scalar >, N >`

Functions

- `template<typename Scalar , size_t N>`
`void getChainIndex (const VectorArray< Scalar, N > &pos, IndexArray< N > &chainIndex)`
Calculate the mapping index from Cartesian coordinate to chain coordinate.
- `template<typename Scalar , size_t N>`
`void createAdjMartix (const VectorArray< Scalar, N > &pos, NodeArray< Scalar, N *(N - 1)/2 > &AdjMatrix)`
Create the adjoint matrix for particle pairs.
- `template<typename Scalar , size_t N>`
`void createChainIndex (NodeArray< Scalar, N *(N - 1)/2 > &AdjMatrix, IndexArray< N > &chainIndex)`
Create mapping index from adjoint matrix.
- `template<size_t N>`
`bool IsDiff (const IndexArray< N > &Index1, const IndexArray< N > &Index2)`
Check if two mapping indexes are the same.
- `template<typename Scalar , size_t N>`
`void updateChain (VectorArray< Scalar, N > &pos, IndexArray< N > &chainIndex, IndexArray< N > &newIndex)`
Update the position chain.
- `template<typename Scalar , size_t N>`
`void synChain (VectorArray< Scalar, N > &data, VectorArray< Scalar, N > &chainData, IndexArray< N > &chainIndex)`
Calculate the chain data from Cartesian data and chain index mapping.
- `template<typename Scalar , size_t N>`
`void synCartesian (VectorArray< Scalar, N > &chainData, VectorArray< Scalar, N > &data, IndexArray< N > &chainIndex)`
Calculate the Cartesian data from chain data and chain index mapping.

6.1.1 Typedef Documentation

6.1.1.1 [IndexArray](#)

```
template<size_t N>
using chain::IndexArray = typedef std::array<size_t, N>
```

6.1.1.2 [NodeArray](#)

```
template<typename Scalar , size_t N>
using chain::NodeArray = typedef std::array<Node<Scalar>, N>
```

6.1.1.3 [VectorArray](#)

```
template<typename Scalar , size_t N>
using chain::VectorArray = typedef std::array<vec3<Scalar>, N>
```

6.1.2 Function Documentation

6.1.2.1 createAdjMartix()

```
template<typename Scalar , size_t N>
void chain::createAdjMartix (
    const VectorArray< Scalar, N > & pos,
    NodeArray< Scalar, N *(N - 1)/2 > & AdjMatrix )
```

Create the adjoint matrix for particle pairs.

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

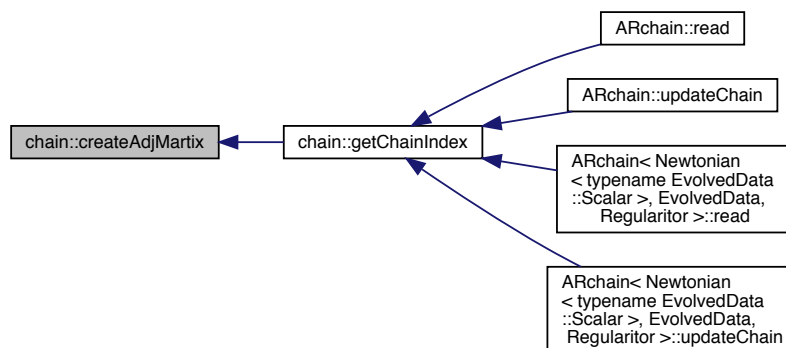
Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.2 createChainIndex()

```
template<typename Scalar , size_t N>
void chain::createChainIndex (
    NodeArray< Scalar, N *(N - 1)/2 > & AdjMatrix,
    IndexArray< N > & chainIndex )
```

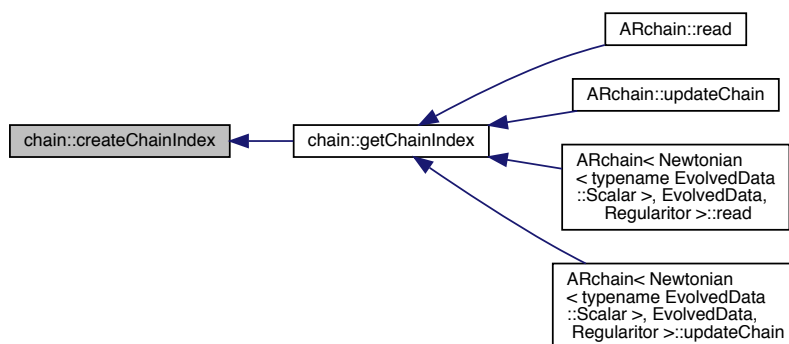
Create mapping index from adjoint matrix.

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

Parameters

<i>AdjMatrix</i>	The adjoint matrix.
<i>chainIndex</i>	The mapping index needs to be calculated as a return value.

Here is the caller graph for this function:



6.1.2.3 getChainIndex()

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

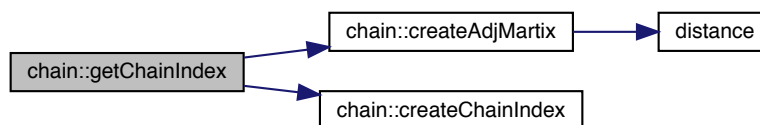
Calculate the mapping index from Cartesian coordinate to chain coordinate.

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

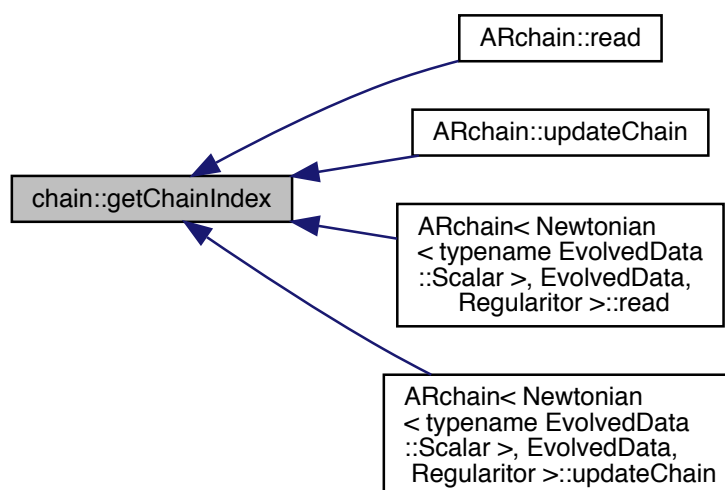
Parameters

<i>pos</i>	The array of particle position, used to calculate the distance of particle pairs.
<i>chainIndex</i>	The mapping index needs to be calculated as a return value.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.4 IsDiff()

```

template<size_t N>
bool chain::IsDiff (
    const IndexArray< N > & Index1,
    const IndexArray< N > & Index2 )
  
```

Check if two mapping indexes are the same.

Checking the identity of two chain index mappings.

Parameters

<i>Index1</i>	The first index array.
<i>Index2</i>	The second index array.

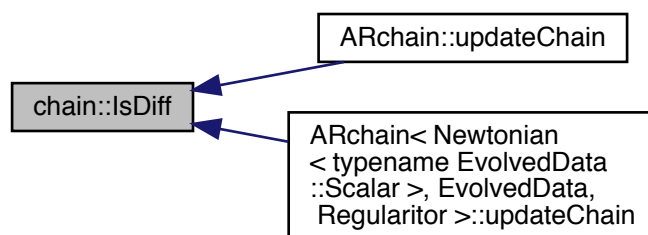
Returns

boolean

Note

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

Here is the caller graph for this function:

**6.1.2.5 synCartesian()**

```

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

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

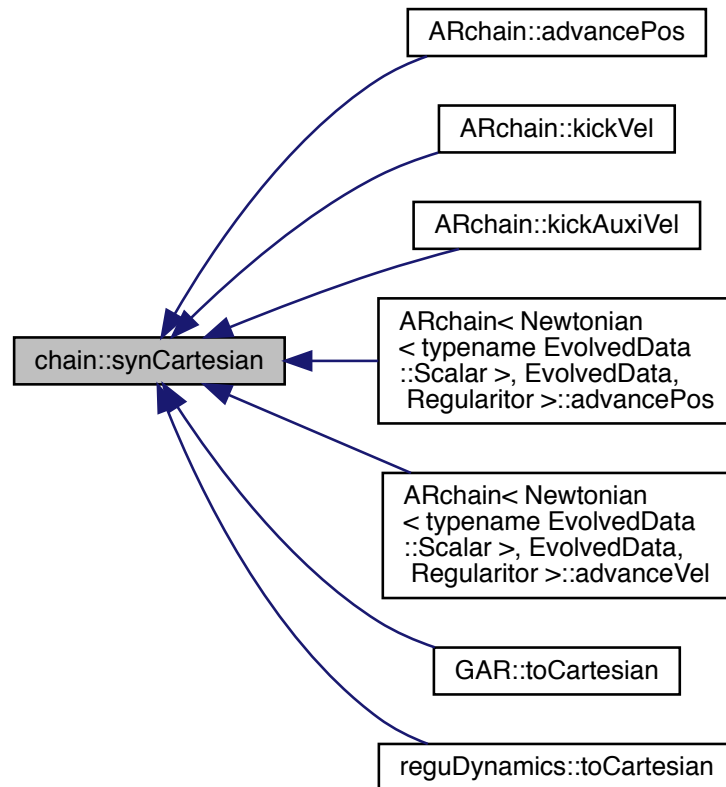
Parameters

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

Note

This function should be a inverse transformation of [synChain\(\)](#).

Here is the caller graph for this function:

**6.1.2.6 synChain()**

```

template<typename Scalar , size_t N>
void chain::synChain (
    VectorArray< Scalar, N > & data,
    VectorArray< Scalar, N > & chainData,
    IndexArray< N > & chainIndex )

```

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

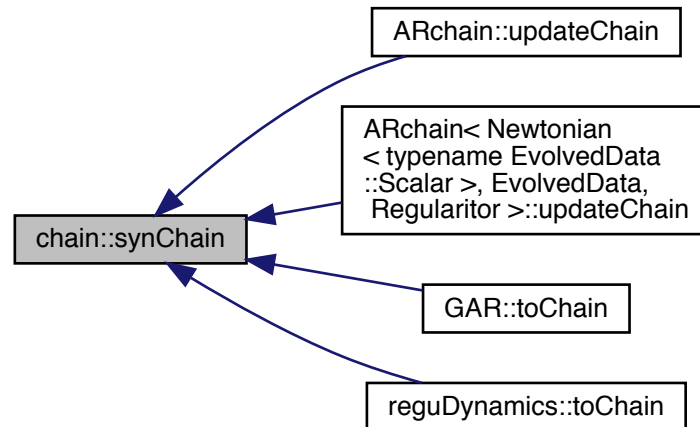
Parameters

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

Note

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

Here is the caller graph for this function:



6.1.2.7 updateChain()

```

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

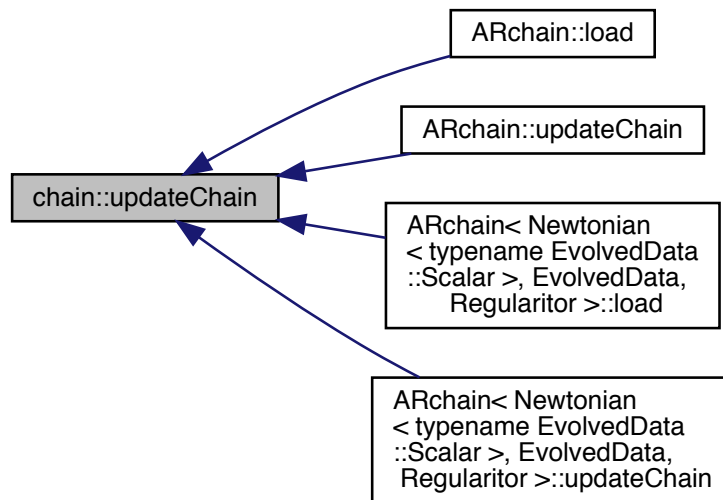
Update the position chain.

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

<i>pos</i>	The old chain position array needs update.
<i>chainIndex</i>	The old chain index mapping.
<i>newIndex</i>	The new chain index mapping.

Here is the caller graph for this function:



6.2 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](#) = true

6.2.1 Function Documentation

6.2.1.1 EraseLine()

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

6.2.1.2 Line()

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

6.2.1.3 RunInfo()

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

6.2.1.4 SubLine()

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

6.2.1.5 SubTitle()

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

6.2.1.6 Telegram()

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

6.2.1.7 Title()

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

6.2.2 Variable Documentation

6.2.2.1 Message

```
bool NOTICE::Message = true
```

6.2.2.2 WIDTH

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

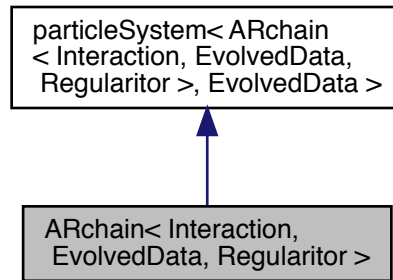
7 Class Documentation

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

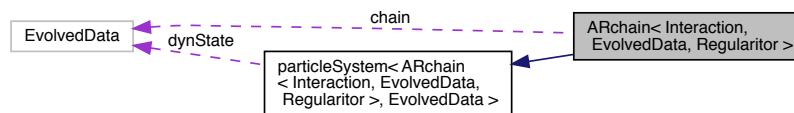
Algorithmatic Regularization chain System.

```
#include <ARchain.h>
```

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



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



Public Types

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

Public Member Functions

- void `advancePos` (`Scalar` timeStepSize)
Advance position one step with current velocity.
- void `advanceVel` (`Scalar` timeStepSize)
Advance velocity one step with current acceleration.
- const `ARchain` & `operator=` (const `ARchain` &other)
Overload operator =.
- `std::istream` & `read` (`std::istream` &)
Input data from standard c++ istream.
- void `load` (`PlainArray` &data)
Load data from a plain array.
- `Scalar` timeScale (`Scalar` scale)
Interface to rescale the time.
- `PlainArray` & `array` ()
Transfer the evolved data to a plain scalar array.

Static Public Member Functions

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

Private Member Functions

- void `advanceOmega` (`Scalar` stepSize)
Advance regularization variable omega.
- void `advanceB` (`Scalar` stepSize)
Advance regularization variable bindE.
- void `kickVel` (`Scalar` stepSize)
Advance velocity with current acceleration.
- void `kickAuxiVel` (`Scalar` stepSize)
Advance auxiliary velocity with current acceleration.
- void `updateAccWith` (`VectorArray` &vel, `VectorArray` &chainVel)
Update velocity dependent acceleration with given velocity. Then update the total acceleration.
- void `updateVelIndepAcc` ()
Update velocity independent acceleration.
- void `updateChain` ()
Update the chain data based on current Cartesian coordinates.

Private Attributes

- EvolvedData `chain`
Evolved variables in chain coordinates.
- `IndexArray` `chainIndex`
Mapping index from Cartesian coordinates to chain coordinates.
- `VectorArray` `velIndepAcc`
Velocity independent acceleration array in Cartesian coordiantes.
- `VectorArray` `chainVelIndepAcc`
Velocity independent acceleration array in chain coordiantes.

- [VectorArray velDepAcc](#)
Velocity dependent acceleration array in Cartesian coordiantes.
- [VectorArray chainVelDepAcc](#)
Velocity dependent acceleration array in chain coordiantes.
- [Interaction velDepForce](#)
Velocity dependent pair force functor.
- [Regularitor regular](#)
Regularization interface.

Additional Inherited Members

7.1.1 Detailed Description

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

Algorithmatic Regularization chain System.

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

7.1.2 Member Typedef Documentation

7.1.2.1 IndexArray

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
typedef std::array<size_t, EvolvedData::size()> ARchain< Interaction, EvolvedData, Regularitor
>::IndexArray
```

7.1.2.2 PlainArray

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
typedef std::array<Scalar, EvolvedData::volume()> ARchain< Interaction, EvolvedData, Regularitor
>::PlainArray
```

7.1.2.3 Scalar

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
typedef EvolvedData::Scalar ARchain< Interaction, EvolvedData, Regularitor >::Scalar
```

7.1.2.4 ScalarArray

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
typedef EvolvedData::ScalarArray ARchain< Interaction, EvolvedData, Regularitor >::ScalarArray
```

7.1.2.5 Vector

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
typedef EvolvedData::Vector ARchain< Interaction, EvolvedData, Regularitor >::Vector
```

7.1.2.6 VectorArray

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
typedef EvolvedData::VectorArray ARchain< Interaction, EvolvedData, Regularitor >::VectorArray
```

7.1.3 Member Function Documentation

7.1.3.1 advanceB()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void ARchain< Interaction, EvolvedData, Regularitor >::advanceB (
    Scalar stepSize ) [private]
```

Advance regularization variable bindE.

Advance bindE with velocity dependent acceleration and auxiliar velocity with physical time step size.

Parameters

<i>stepSize</i>	Physical time step.
-----------------	---------------------

Note

Update the bindE in chain, which will be processed by the integrator.

Here is the call graph for this function:



7.1.3.2 advanceOmega()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void ARchain< Interaction, EvolvedData, Regularitor >::advanceOmega (
    Scalar stepSize ) [private]
```

Advance regularization variable omega.

Advance omega with velocity independent acceleration and auxiliar velocity with physical time step size.

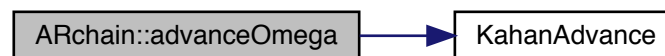
Parameters

<i>stepSize</i>	Physical time step.
-----------------	---------------------

Note

Update the omega in chain, which will be processed by the integrator.

Here is the call graph for this function:



7.1.3.3 advancePos()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void ARchain< Interaction, EvolvedData, Regularitor >::advancePos (
    Scalar timeStepSize )
```

Advance position one step with current velocity.

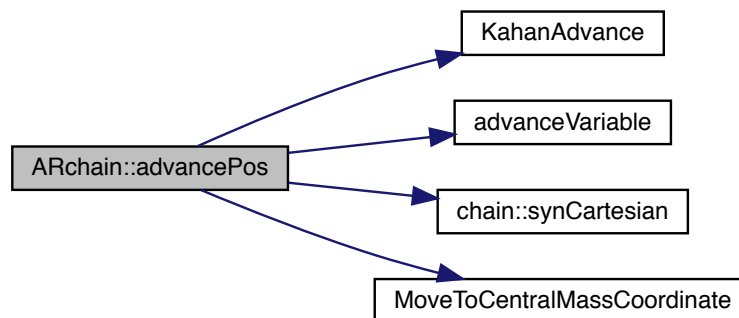
Advance chain position one step with current velocity.

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

Parameters

<i>timeStepSize</i>	Integration step size, will be transfered to physical time in the function.
---------------------	---

Here is the call graph for this function:



7.1.3.4 `advanceVel()`

```

template<typename Interaction , typename EvolvedData , typename Regularitor >
void ARchain< Interaction, EvolvedData, Regularitor >::advanceVel (
    Scalar timeStepSize )
  
```

Advance velocity one step with current acceleration.

Advance chain velocity one step with current acceleration.

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

Parameters

<i>timeStepSize</i>	Integration step size, will be transferred to physical time in the function.
---------------------	--

7.1.3.5 `array()`

```

template<typename Interaction, typename EvolvedData, typename Regularitor>
PlainArray& ARchain< Interaction, EvolvedData, Regularitor >::array ( ) [inline]
  
```

Transfer the evolved data to a plain scalar array.

Note

Overload base class `array()`.

7.1.3.6 kickAuxiVel()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void ARchain< Interaction, EvolvedData, Regularitor >::kickAuxiVel (
    Scalar stepSize ) [private]
```

Advance auxiliary velocity with current acceleration.

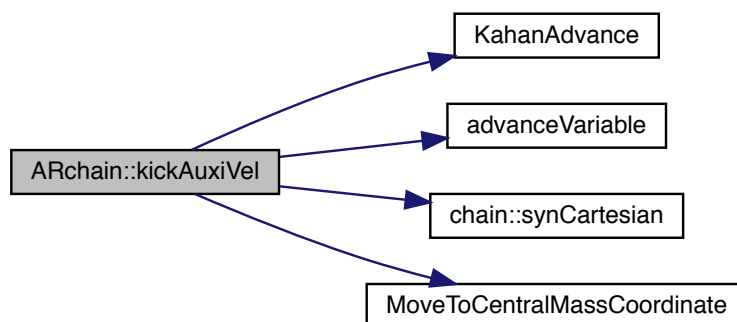
Advance chain auxiliary velocity with current acceleration.

Advance the chain auxiliary velocity with current total acceleration variable 'acc' and synchronize the Cartesian data.

Parameters

<i>stepSize</i>	Integration step size.
-----------------	------------------------

Here is the call graph for this function:



7.1.3.7 kickVel()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void ARchain< Interaction, EvolvedData, Regularitor >::kickVel (
    Scalar stepSize ) [private]
```

Advance velocity with current acceleration.

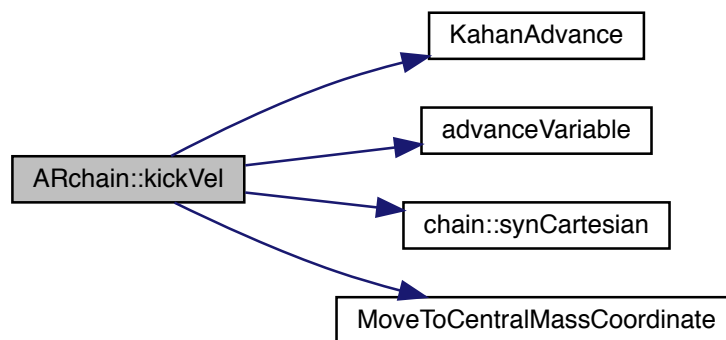
Advance chain velocity with current acceleration.

Advance the chain velocity with current total acceleration variable 'acc' and synchronize the Cartesian data.

Parameters

<i>stepSize</i>	Integration step size.
-----------------	------------------------

Here is the call graph for this function:



7.1.3.8 load()

```

template<typename Interaction , typename EvolvedData , typename Regularitor >
void ARchain< Interaction, EvolvedData, Regularitor >::load (
    PlainArray & data )
  
```

Load data from a plain array.

Note

Overload base class `load()`.

Interface used by integrator and ODE iterator. Load data from a plain array processed by itegrator and iterator to chain data. Then synchrosize Cartesian data and update the chain. Derived class could overload this function to additional process.

Parameters

<i>data</i>	Plain scalar array.
-------------	---------------------

Here is the call graph for this function:



7.1.3.9 operator=()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
const ARchain< Interaction, EvolvedData, Regularitor > & ARchain< Interaction, EvolvedData,
Regularitor >::operator= (
    const ARchain< Interaction, EvolvedData, Regularitor > & other )
```

Overload operator =.

7.1.3.10 read()

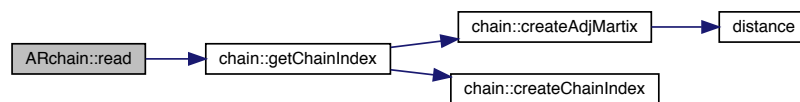
```
template<typename Interaction , typename EvolvedData , typename Regularitor >
std::istream & ARchain< Interaction, EvolvedData, Regularitor >::read (
    std::istream & input )
```

Input data from standard c++ istream.

Note

Overload base class [read\(\)](#).

Implement of CRTP '>>' method. Here is the call graph for this function:



7.1.3.11 size()

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
static constexpr size_t ARchain< Interaction, EvolvedData, Regularitor >::size ( ) [inline],
[static]
```

Get the number of the particles.

Returns

The particle number.

Note

Overload base class [size\(\)](#).

7.1.3.12 timeScale()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
EvolvedData::Scalar ARchain< Interaction, EvolvedData, Regularitor >::timeScale (
    Scalar scale )
```

Interface to rescale the time.

Note

Overload base class `timeScale()`.

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

Returns

The physical time.

7.1.3.13 updateAccWith()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void ARchain< Interaction, EvolvedData, Regularitor >::updateAccWith (
    VectorArray & velocity,
    VectorArray & chainVel ) [private]
```

Update velocity dependent acceleration with given velocity. Then update the total acceleration.

Update the velocity dependent acceleration variable 'velDepAcc' with given velocity and velocity dependent pair force 'velDepForce'. Then update the total acceleration 'acc' by adding 'velIndepAcc' and 'velDepAcc'.

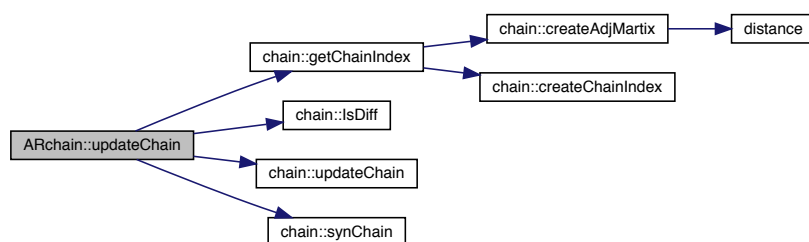
7.1.3.14 updateChain()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void ARchain< Interaction, EvolvedData, Regularitor >::updateChain ( ) [private]
```

Update the chain data based on current Cartesian coordinates.

Update chain index and chain.

Due to the evolution, the relative position of particles may vary with time. This function update the chain index and chain if needed. Here is the call graph for this function:



7.1.3.15 updateVelIndepAcc()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void ARchain< Interaction, EvolvedData, Regularitor >::updateVelIndepAcc ( ) [private]
```

Update velocity independent acceleration.

Update velocity independent acceleration 'velIndepAcc' based on [Newtonian](#) interaction with chain data.

7.1.4 Member Data Documentation

7.1.4.1 chain

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
EvolvedData ARchain< Interaction, EvolvedData, Regularitor >::chain [private]
```

Evolved variables in chain coordinates.

7.1.4.2 chainIndex

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
IndexArray ARchain< Interaction, EvolvedData, Regularitor >::chainIndex [private]
```

Mapping index from Cartesian coordinates to chain coordinates.

7.1.4.3 chainVelDepAcc

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
VectorArray ARchain< Interaction, EvolvedData, Regularitor >::chainVelDepAcc [private]
```

Velocity dependent acceleration array in chain coordiantes.

7.1.4.4 chainVelIndepAcc

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
VectorArray ARchain< Interaction, EvolvedData, Regularitor >::chainVelIndepAcc [private]
```

Velocity independent acceleration array in chain coordiantes.

7.1.4.5 regular

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
Regularitor ARchain< Interaction, EvolvedData, Regularitor >::regular [private]
```

Regularization interface.

7.1.4.6 velDepAcc

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
VectorArray ARchain< Interaction, EvolvedData, Regularitor >::velDepAcc [private]
```

Velocity dependent acceleration array in Cartesian coordiantes.

7.1.4.7 velDepForce

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
Interaction ARchain< Interaction, EvolvedData, Regularitor >::velDepForce [private]
```

Velocity dependent pair force functor.

7.1.4.8 velIndepAcc

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
VectorArray ARchain< Interaction, EvolvedData, Regularitor >::velIndepAcc [private]
```

Velocity independent acceleration array in Cartesian coordiantes.

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

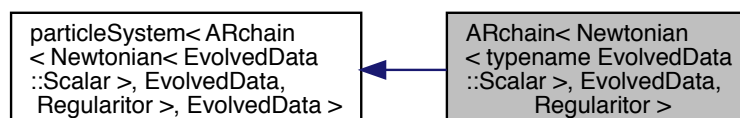
- particleSystem/[ARchain.h](#)

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

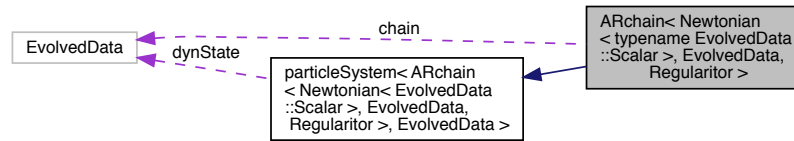
Partial specilization of template [ARchain](#).

```
#include <ARchain.h>
```

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



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



Public Types

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

Public Member Functions

- void [advancePos](#) ([Scalar](#) timeStepSize)
Advance position one step with current velocity.
- void [advanceVel](#) ([Scalar](#) timeStepSize)
Advance velocity one step with current acceleration.
- const [ARchain](#) & [operator=](#) (const [ARchain](#) &other)
Overload operator =.
- [std::istream](#) & [read](#) ([std::istream](#) &)
Input data from standard c++ istream.
- void [load](#) ([PlainArray](#) &data)
Load data from a plain array.
- [Scalar](#) [timeScale](#) ([Scalar](#) scale)
Interface to rescale the time.
- [PlainArray](#) & [array](#) ()
Transfer the evolved data to a plain scalar array.

Static Public Member Functions

- static constexpr size_t [size](#) ()
Get the number of the particles.

Private Member Functions

- void [advanceOmega](#) ([Scalar](#) stepSize)
Advance regularization variable omega.
- void [updateVelIndepAcc](#) ()
Update velocity independent acceleration.
- void [updateChain](#) ()
Update the chain data based on current Cartesian coordinates.

Private Attributes

- EvolvedData [chain](#)
Evolved variables in chain coordinates.
- [IndexArray](#) [chainIndex](#)
Mapping index from Cartesian coordinates to chain coordinates.
- [VectorArray](#) [vellIndepAcc](#)
Velocity independent acceleration array in Cartesian coordiantes.
- Regularitor [regular](#)
Regularization interface.

Additional Inherited Members

7.2.1 Detailed Description

```
template<typename EvolvedData, typename Regularitor>
class ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >
```

Partial specilization of template [ARchain](#).

7.2.2 Member Typedef Documentation

7.2.2.1 IndexArray

```
template<typename EvolvedData , typename Regularitor >
typedef std::array<size_t, EvolvedData::size()> ARchain< Newtonian< typename EvolvedData::Scalar
>, EvolvedData, Regularitor >::IndexArray
```

7.2.2.2 PlainArray

```
template<typename EvolvedData , typename Regularitor >
typedef std::array<Scalar, EvolvedData::volume()> ARchain< Newtonian< typename EvolvedData::Scalar
>, EvolvedData, Regularitor >::PlainArray
```

7.2.2.3 Scalar

```
template<typename EvolvedData , typename Regularitor >
typedef EvolvedData::Scalar ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData,
Regularitor >::Scalar
```

7.2.2.4 ScalarArray

```
template<typename EvolvedData , typename Regularitor >
typedef EvolvedData::ScalarArray ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::ScalarArray
```

7.2.2.5 Vector

```
template<typename EvolvedData , typename Regularitor >
typedef EvolvedData::Vector ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::Vector
```

7.2.2.6 VectorArray

```
template<typename EvolvedData , typename Regularitor >
typedef EvolvedData::VectorArray ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::VectorArray
```

7.2.3 Member Function Documentation

7.2.3.1 advanceOmega()

```
template<typename EvolvedData , typename Regularitor >
void ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::advanceOmega (
    Scalar stepSize ) [private]
```

Advance regularization variable omega.

Advance omega with velocity independent acceleration and velocity with physical time step size.

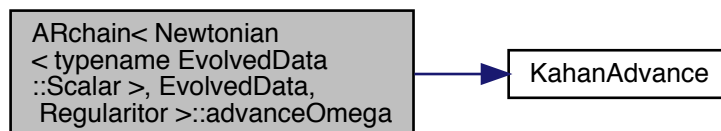
Parameters

<i>stepSize</i>	Physical time step.
-----------------	---------------------

Note

Update the omega in chain, which will be processed by the integrator.

Here is the call graph for this function:

**7.2.3.2 advancePos()**

```

template<typename EvolvedData , typename Regularitor >
void ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::advance←
Pos (
    Scalar timeStepSize )
  
```

Advance position one step with current velocity.

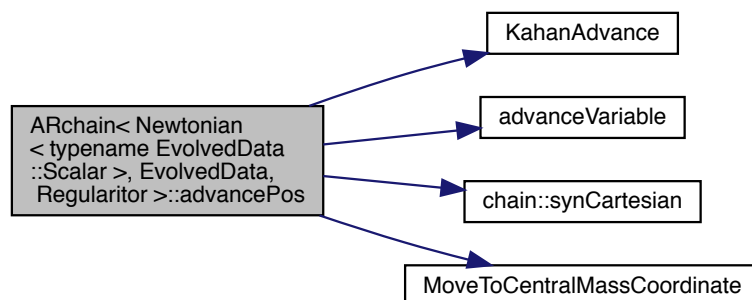
Advance chain position one step with current velocity.

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

Parameters

<i>timeStepSize</i>	Integration step size, will be transfered to physical time in the function.
---------------------	---

Here is the call graph for this function:



7.2.3.3 advanceVel()

```
template<typename EvolvedData , typename Regularitor >
void ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::advanceVel (
    Scalar timeStepSize )
```

Advance velocity one step with current acceleration.

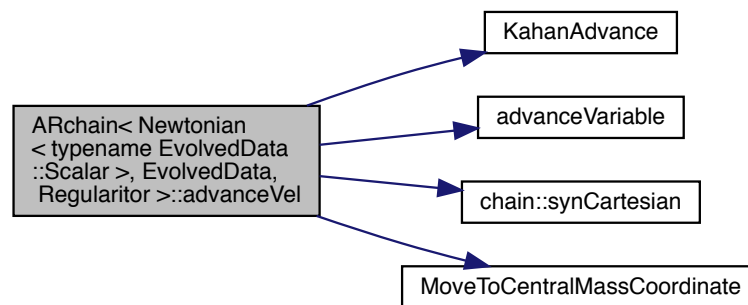
Advance chain velocity one step with current acceleration.

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

Parameters

<i>timeStepSize</i>	Integration step size, will be transfered to physical time in the function.
---------------------	---

Here is the call graph for this function:



7.2.3.4 array()

```
template<typename EvolvedData , typename Regularitor >
PlainArray& ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::array ( ) [inline]
```

Transfer the evolved data to a plain scalar array.

Note

Overload base class `array()`.

7.2.3.5 load()

```
template<typename EvolvedData , typename Regularitor >
void ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::load (
    PlainArray & data )
```

Load data from a plain array.

Note

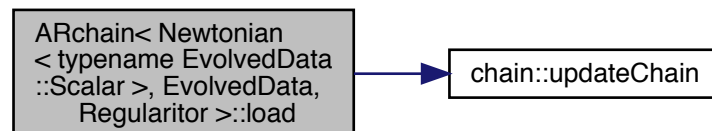
Overload base class `load()`.

Interface used by integrator and ODE iterator. Load data from a plain array processed by itegrator and iterator to chain data. Then synchrosize Cartesian data and update the chain. Derived class could overload this function to additional process.

Parameters

<i>data</i>	Plain scalar array.
-------------	---------------------

Here is the call graph for this function:



7.2.3.6 operator=()

```
template<typename EvolvedData , typename Regularitor >
const ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor > & ARchain<
Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::operator= (
    const ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor
> & other )
```

Overload operator =.

7.2.3.7 read()

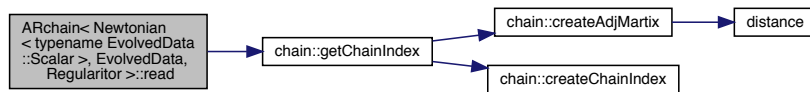
```
template<typename EvolvedData , typename Regularitor >
std::istream & ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor
>::read (
    std::istream & input )
```

Input data from standard c++ istream.

Note

Overload base class `read()`.

Implement of CRTP '>>' method. Here is the call graph for this function:



7.2.3.8 size()

```
template<typename EvolvedData , typename Regularitor >
static constexpr size_t ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData,
Regularitor >::size ( ) [inline], [static]
```

Get the number of the particles.

Returns

The particle number.

Note

Overload base class `size()`.

7.2.3.9 timeScale()

```
template<typename EvolvedData , typename Regularitor >
EvolvedData::Scalar ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor
>::timeScale (
    Scalar scale )
```

Interface to rescale the time.

Note

Overload base class `timeScale()`.

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

Returns

The physical time.

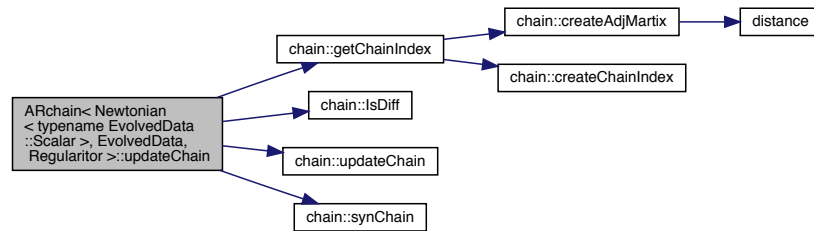
7.2.3.10 updateChain()

```
template<typename EvolvedData , typename Regularitor >
void ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::updateChain ( ) [private]
```

Update the chain data based on current Cartesian coordinates.

Update chain index and chain.

Due to the evolution, the relative position of particles may vary with time. This function update the chain index and chain if needed. Here is the call graph for this function:



7.2.3.11 updateVelIndepAcc()

```
template<typename EvolvedData , typename Regularitor >
void ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::updateVelIndepAcc ( ) [private]
```

Update velocity independent acceleration.

Update velocity independent acceleration 'velIndepAcc' based on `Newtonian` interaction with chain data. Then calculate the corresponding chain acceleration.

7.2.4 Member Data Documentation

7.2.4.1 chain

```
template<typename EvolvedData , typename Regularitor >
EvolvedData ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::chain [private]
```

Evolved variables in chain coordinates.

7.2.4.2 chainIndex

```
template<typename EvolvedData , typename Regularitor >
IndexArray ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >↔
::chainIndex [private]
```

Mapping index from Cartesian coordinates to chain coordinates.

7.2.4.3 regular

```
template<typename EvolvedData , typename Regularitor >
Regularitor ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >↔
::regular [private]
```

Regularization interface.

7.2.4.4 velIndepAcc

```
template<typename EvolvedData , typename Regularitor >
VectorArray ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >↔
::velIndepAcc [private]
```

Velocity independent acceleration array in Cartesian coordiantes.

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

- particleSystem/[ARchain.h](#)

7.3 BSIterator< ParticSys, Integrator > Class Template Reference

Bulirsch-Stoer extrapolation algorithm.

```
#include <BSIterator.h>
```

Public Types

- typedef ParticSys::Scalar [Scalar](#)
- template<typename Scalar , size_t N>
using [scalarArray](#) = std::array< [Scalar](#), N >

Public Member Functions

- [BSIterator](#) ()
Constructor for initializing cost, nSteps, fmin and CC.
- [Scalar](#) [iterate](#) (ParticSys &particles, Integrator &integrator, [Scalar](#) stepLength)
Interface of ODE iterator.
- void [setRelativeError](#) ([Scalar](#) relError)
Set the local relative error.
- void [setAbsoluteError](#) ([Scalar](#) absError)
Set the local absolute error.

Private Member Functions

- void `copyDataToExtrapTab` (size_t k)
Copy the current results of localsystem as a plain array to the first column of extrapolation talbe.
- bool `checkRejection` (Scalar error, size_t k) const
Check the rejection criteria for current iteration.
- void `extrapolate` (size_t k)
Extrapolate the kth row to the right end.
- Scalar `getError` (size_t k) const
Calculate the error of the k row of the extrapolation table.
- Scalar `getTimeStepCoef` (Scalar error, size_t order)
Calculate the new iteration integration step coefficient.
- Scalar `prepareForNewIteration` (size_t k, bool lastRejection)
Calculate the new iteration integration step and new iteration depth for next iteration.

Private Attributes

- ParticSys `localSystem`
The local partical system used to iterate.
- scalarArray< scalarArray< Scalar, ParticSys::volume()>,(MaxDepth+1) *(MaxDepth+2)/2 > `extrapTab`
Extrapolation table.
- scalarArray< Scalar,(MaxDepth+1) *(MaxDepth+2)/2 > `CC`
Extrapolation coefficient.
- scalarArray< Scalar, MaxDepth+1 > `macroStepLength`
Macro step length for different iteration depth.
- scalarArray< Scalar, MaxDepth+1 > `work`
The work(calculation quantities) per integration length of each iteration depth.
- scalarArray< Scalar, MaxDepth+1 > `fmin`
The minimal coefficient of integration step estimation.
- scalarArray< size_t, MaxDepth+1 > `cost`
The work(calculation quantities) of each iteration depth.
- scalarArray< size_t, MaxDepth+1 > `nSteps`
Steps of integration of each iteration depth.
- Scalar `absoluteError` {1e-15}
Local absolute error.
- Scalar `relativeError` {1e-15}
Local relative error.
- Scalar `s1` {0.94}
Coefficient of new step length estimation. See "Numerical recipes" on page 926.
- Scalar `s2` {0.95}
Coefficient of new step length estimation.
- Scalar `s3` {0.02}
Coefficient of new step length estimation.
- Scalar `s4` {4.0}
Coefficient of new step length estimation.
- size_t `iterDepth` {7}
Current iteraation depth.

Static Private Attributes

- static const size_t `MaxDepth` {8}
The Maximum iteration depth.

7.3.1 Detailed Description

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

Bulirsch-Stoer extrapolation algorithm.

7.3.2 Member Typedef Documentation

7.3.2.1 Scalar

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

7.3.2.2 scalarArray

```
template<typename ParticSys , typename Integrator >
template<typename Scalar , size_t N>
using BSIterator< ParticSys, Integrator >::scalarArray = std::array<Scalar, N>
```

7.3.3 Constructor & Destructor Documentation

7.3.3.1 BSIterator()

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

Constructor for initializing cost, nSteps, fmin and CC.

7.3.4 Member Function Documentation

7.3.4.1 checkRejection()

```
template<typename ParticSys , typename Integrator >
bool BSIterator< ParticSys, Integrator >::checkRejection (
    Scalar error,
    size_t k ) const [private]
```

Check the rejection criteria for current iteration.

Parameters

k	The k th iteration depth.
-----	-----------------------------

Returns

If current iteration is rejected.

7.3.4.2 copyDataToExtrapTab()

```
template<typename ParticSys , typename Integrator >
void BSIterator< ParticSys, Integrator >::copyDataToExtrapTab (
    size_t k ) [private]
```

Copy the current results of localsystem as a plain array to the first column of extrapolation talbe.

Parameters

k	The k th row of extrapolation table.
-----	--

7.3.4.3 extrapolate()

```
template<typename ParticSys , typename Integrator >
void BSIterator< ParticSys, Integrator >::extrapolate (
    size_t k ) [private]
```

Extrapolate the k th row to the right end.

Extrapolate the k -th row to the right end.

Parameters

k	The k th row of extrapolation table.
-----	--

7.3.4.4 getError()

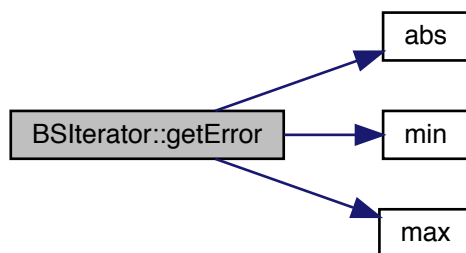
```
template<typename ParticSys , typename Integrator >
ParticSys::Scalar BSIterator< ParticSys, Integrator >::getError (
    size_t k ) const [private]
```

Calculate the error of the k row of the extrapolation table.

Parameters

k	The k th row of extrapolation table.
-----	--

Here is the call graph for this function:



7.3.4.5 getTimeStepCoef()

```

template<typename ParticSys , typename Integrator >
ParticSys::Scalar BSIterator< ParticSys, Integrator >::getTimeStepCoef (
    Scalar error,
    size_t order ) [private]
  
```

Calculate the new iteration integration step coefficient.

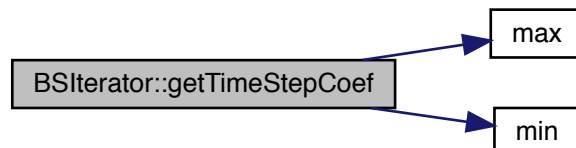
Parameters

<i>error</i>	The error of the k-th row of extrapolation table.
<i>order</i>	The order of the error in k-th row of extrapolation table.

Returns

The new iteration integration step length coefficient.

Here is the call graph for this function:



7.3.4.6 `iterate()`

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

Interface of ODE iterator.

Note

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

Parameters

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

Returns

The next macro integration step length.

7.3.4.7 `prepareForNewIteration()`

```
template<typename ParticSys , typename Integrator >
ParticSys::Scalar BSIterator< ParticSys, Integrator >::prepareForNewIteration (
    size_t k,
    bool lastRejection ) [private]
```

Calculate the new iteration integration step and new iteration depth for next iteration.

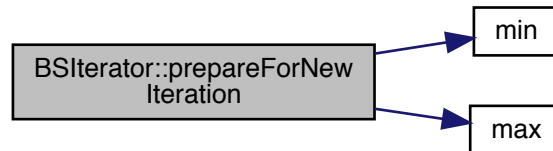
Parameters

<i>k</i>	The k-th iteration depth.
<i>lastRejection</i>	The rejection status of last trail of iteration.

Returns

The new iteration step length.

Here is the call graph for this function:



7.3.4.8 setAbsoluteError()

```

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

Set the local absolute error.

7.3.4.9 setRelativeError()

```

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

Set the local relative error.

7.3.5 Member Data Documentation

7.3.5.1 absoluteError

```

template<typename ParticSys , typename Integrator >
Scalar BSIterator< ParticSys, Integrator >::absoluteError {1e-15} [private]
  
```

Local absolute error.

7.3.5.2 CC

```
template<typename ParticSys , typename Integrator >
scalarArray< Scalar, (MaxDepth + 1)* (MaxDepth + 2) / 2 > BSIterator< ParticSys, Integrator
>::CC [private]
```

Extrapolation coefficient.

7.3.5.3 cost

```
template<typename ParticSys , typename Integrator >
scalarArray< size_t, MaxDepth + 1 > BSIterator< ParticSys, Integrator >::cost [private]
```

The work(calculation quantities) of each iteration depth.

7.3.5.4 extrapTab

```
template<typename ParticSys , typename Integrator >
scalarArray< scalarArray<Scalar, ParticSys::volume()>, (MaxDepth + 1)* (MaxDepth + 2) / 2 >
BSIterator< ParticSys, Integrator >::extrapTab [private]
```

Extrapolation table.

7.3.5.5 fmin

```
template<typename ParticSys , typename Integrator >
scalarArray< Scalar, MaxDepth + 1 > BSIterator< ParticSys, Integrator >::fmin [private]
```

The minimal coefficient of integration step estimation.

7.3.5.6 iterDepth

```
template<typename ParticSys , typename Integrator >
size_t BSIterator< ParticSys, Integrator >::iterDepth {7} [private]
```

Current iteration depth.

7.3.5.7 localSystem

```
template<typename ParticSys , typename Integrator >
ParticSys BSIterator< ParticSys, Integrator >::localSystem [private]
```

The local partial system used to iterate.

7.3.5.8 macroStepLength

```
template<typename ParticSys , typename Integrator >
scalarArray< Scalar, MaxDepth + 1 > BSIterator< ParticSys, Integrator >::macroStepLength
[private]
```

Macro step length for different iteration depth.

7.3.5.9 MaxDepth

```
template<typename ParticSys , typename Integrator >
const size_t BSIterator< ParticSys, Integrator >::MaxDepth {8} [static], [private]
```

The Maximum iteration depth.

7.3.5.10 nSteps

```
template<typename ParticSys , typename Integrator >
scalarArray< size_t, MaxDepth + 1 > BSIterator< ParticSys, Integrator >::nSteps [private]
```

Steps of integration of each iteration depth.

7.3.5.11 relativeError

```
template<typename ParticSys , typename Integrator >
Scalar BSIterator< ParticSys, Integrator >::relativeError {1e-15} [private]
```

Local relative error.

7.3.5.12 s1

```
template<typename ParticSys , typename Integrator >
Scalar BSIterator< ParticSys, Integrator >::s1 {0.94} [private]
```

Coefficient of new step length estimation. See "Numerical recipes" on page 926.

7.3.5.13 s2

```
template<typename ParticSys , typename Integrator >
Scalar BSIterator< ParticSys, Integrator >::s2 {0.95} [private]
```

Coefficient of new step length estimation.

7.3.5.14 s3

```
template<typename ParticSys , typename Integrator >
Scalar BSIterator< ParticSys, Integrator >::s3 {0.02} [private]
```

Coefficient of new step length estimation.

7.3.5.15 s4

```
template<typename ParticSys , typename Integrator >
Scalar BSIterator< ParticSys, Integrator >::s4 {4.0} [private]
```

Coefficient of new step length estimation.

7.3.5.16 work

```
template<typename ParticSys , typename Integrator >
scalarArray< Scalar, MaxDepth + 1 > BSIterator< ParticSys, Integrator >::work [private]
```

The work(calculation quantities) per integration length of each iteration depth.

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

- [ODEiterator/BSIterator.h](#)

7.4 constIterator< ParticSys, Integrator > Class Template Reference

Most common iterator.

```
#include <constIterator.h>
```

Public Types

- typedef ParticSys::Scalar [Scalar](#)
interface to iterate particle system for one step

Public Member Functions

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

7.4.1 Detailed Description

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

Most common iterator.

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

7.4.2 Member Typedef Documentation

7.4.2.1 Scalar

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

interface to iterate particle system for one step

Parameters

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

Returns

step length for next iteration.

7.4.3 Member Function Documentation

7.4.3.1 iterate()

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

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

- ODEiterator/[constIterator.h](#)

7.5 dynamics< DataType, N > Class Template Reference

Class of dynamical variable.

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

Public Types

- typedef DataType [Scalar](#)
- typedef [vec3](#)< [Scalar](#) > [Vector](#)
- typedef std::array< [vec3](#)< [Scalar](#) >, N > [VectorArray](#)
- typedef std::array< [Scalar](#), N > [ScalarArray](#)

Public Member Functions

- std::array< [Scalar](#), [volume](#)()> & [array](#) ()
Transfer this class to a plain array.
- void [initAddiVariable](#) ([ScalarArray](#) &mass)
Initialize extra user defined variables. Interface required for other class.
- void [setZero](#) ()
Set all data to be zero.

Static Public Member Functions

- static constexpr size_t `size` ()
Get the number of the particles.
- static constexpr size_t `volume` ()
Get the total data number.

Public Attributes

- `VectorArray pos`
Array of position of the particles. Element is 3D vector.
- `VectorArray vel`
Array of velocity of the particles. Element is 3D vector.
- `Scalar time {0.0}`
The physical time of the dynamic system.

7.5.1 Detailed Description

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

Class of dynamical variable.

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

7.5.2 Member Typedef Documentation

7.5.2.1 Scalar

```
template<typename DataType , size_t N>
typedef DataType dynamics< DataType, N >::Scalar
```

7.5.2.2 ScalarArray

```
template<typename DataType , size_t N>
typedef std::array<Scalar, N> dynamics< DataType, N >::ScalarArray
```

7.5.2.3 Vector

```
template<typename DataType , size_t N>
typedef vec3<Scalar> dynamics< DataType, N >::Vector
```

7.5.2.4 VectorArray

```
template<typename DataType , size_t N>
typedef std::array<vec3<Scalar>, N> dynamics< DataType, N >::VectorArray
```

7.5.3 Member Function Documentation

7.5.3.1 array()

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

Transfer this class to a plain array.

Returns

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

7.5.3.2 initAddiVariable()

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

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

Parameters

<i>mass</i>	The mass of particles, might be required for initialization.
-------------	--

7.5.3.3 setZero()

```
template<typename DataType , size_t N>
void dynamics< DataType, N >::setZero ( ) [inline]
```

Set all data to be zero.

Here is the call graph for this function:



7.5.3.4 size()

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

Get the number of the particles.

Returns

The particle number.

7.5.3.5 volume()

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

Get the total data number.

Returns

The data number.

Here is the caller graph for this function:



7.5.4 Member Data Documentation

7.5.4.1 pos

```
template<typename DataType , size_t N>
VectorArray dynamics< DataType, N >::pos
```

Array of position of the particles. Element is 3D vector.

7.5.4.2 time

```
template<typename DataType , size_t N>
Scalar dynamics< DataType, N >::time {0.0}
```

The physical time of the dynamic system.

7.5.4.3 vel

```
template<typename DataType , size_t N>
VectorArray dynamics< DataType, N >::vel
```

Array of velocity of the particles. Element is 3D vector.

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

- [dynamicState.h](#)

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

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

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

Public Member Functions

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

Public Attributes

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

Private Member Functions

- void `getInitStepLength()`
Calculate the initial step length of the particle system.

7.6.1 Detailed Description

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

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

7.6.2 Constructor & Destructor Documentation

7.6.2.1 ~dynamicSystem()

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

Default destructor, virtualize for inherent class.

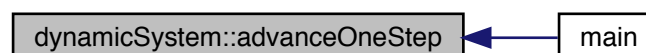
7.6.3 Member Function Documentation

7.6.3.1 advanceOneStep()

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

Advance the particle system for one step.

Advance the particle system with current steplength `stepLength`. The ODE iterator iterate the integrator to convergence by its own implement. The step length will also be updated by its own implement. Here is the caller graph for this function:



7.6.3.2 getInitStepLength()

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

Calculate the initial step length of the particle system.

If the user didn't set the step length with [setStepLength\(\)](#), calculate the proper initial step length automatically.

7.6.3.3 loadText()

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

Load particle system initial condition from file.

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

Parameters

<i>initFilePath</i>	The relative path of initial conditions file
---------------------	--

Exceptions

<i>If</i>	the particile number in the header is inconsisitent with the size of particles, this function will throw an exception.
-----------	--

Here is the caller graph for this function:



7.6.3.4 setStepLength()

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

Set the step length.

Here is the caller graph for this function:



7.6.4 Member Data Documentation

7.6.4.1 integrator

```
template<typename ParticSys, typename Integrator, typename ODEiterator>
Integrator dynamicSystem< ParticSys, Integrator, ODEiterator >::integrator
```

Integrator.

7.6.4.2 iterator

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

ODE Iterator.

7.6.4.3 particles

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

Particle system.

7.6.4.4 stepLength

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

Macro step size for ODE iterator.

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

- [dynamicSystem.h](#)

7.7 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](#) ()

Private Attributes

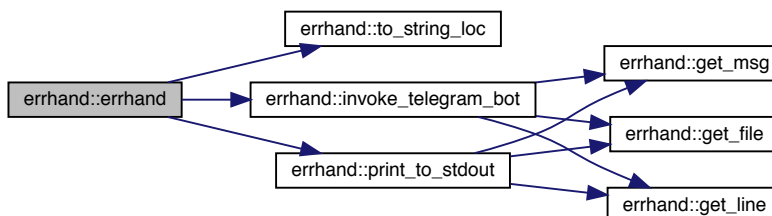
- std::string [err_msg](#)
- std::string [file](#)
- size_t [line](#)

7.7.1 Constructor & Destructor Documentation

7.7.1.1 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:

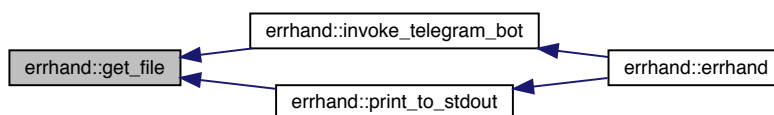


7.7.2 Member Function Documentation

7.7.2.1 get_file()

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

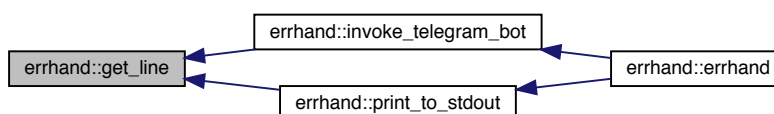
Here is the caller graph for this function:



7.7.2.2 get_line()

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

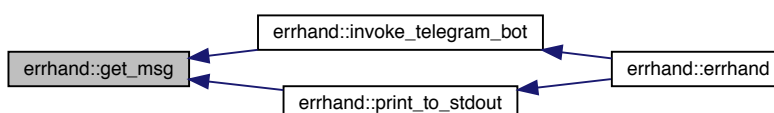
Here is the caller graph for this function:



7.7.2.3 get_msg()

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

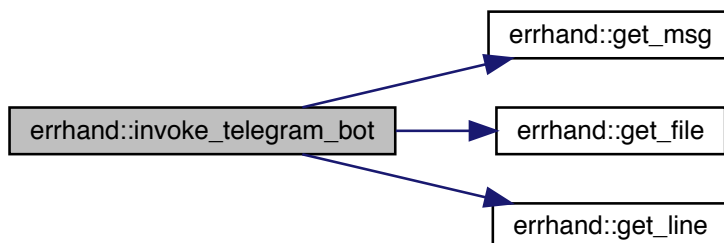
Here is the caller graph for this function:



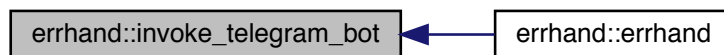
7.7.2.4 invoke_telegram_bot()

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

Here is the call graph for this function:



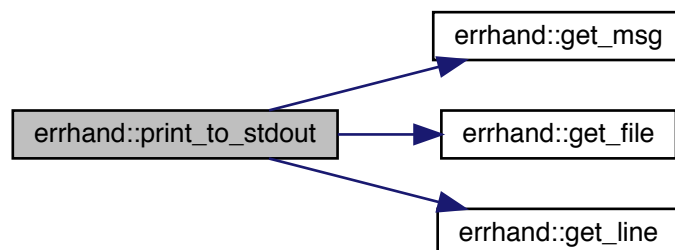
Here is the caller graph for this function:



7.7.2.5 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:



7.7.2.6 `to_string_loc()`

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

Here is the caller graph for this function:



7.7.3 Member Data Documentation

7.7.3.1 `err_msg`

```
std::string errhand::err_msg [private]
```

7.7.3.2 `file`

```
std::string errhand::file [private]
```

7.7.3.3 `line`

```
size_t errhand::line [private]
```

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

- [errhand.h](#)

7.8 GAR< DataType, N > Class Template Reference

Class of velocity dependent dynamical system with regularization variables.

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

Public Types

- typedef DataType [Scalar](#)
- typedef [vec3](#)< [Scalar](#) > [Vector](#)
- typedef std::array< [vec3](#)< [Scalar](#) >, N > [VectorArray](#)
- typedef std::array< [Scalar](#), N > [ScalarArray](#)
- typedef std::array< size_t, N > [IndexArray](#)

Public Member Functions

- std::array< [Scalar](#), [volume](#)()> & [array](#) ()
Transfer this class to a plain array.
- void [setZero](#) ()
Set all data to be zero.
- [Scalar](#) [getOmega](#) (const [ScalarArray](#) &mass)
Calculate the regularization variable omega.
- void [initAddiVariable](#) ([ScalarArray](#) &mass)
Initialize extra user defined variables. Interface required for other class.
- void [toChain](#) ([GAR](#) &chainData, [IndexArray](#) &index)
Transfer Cartesian coordinate regularization system to chain regularization system.
- void [toCartesian](#) ([GAR](#) &cartesian, [IndexArray](#) &index)
Transfer chain coordinate regularization system to Cartesian regularization system.
- void [moveToCentralMassCoords](#) ([ScalarArray](#) &mass)
Move particles to central mass coordinates.

Static Public Member Functions

- static constexpr size_t [size](#) ()
Get the number of the particles.
- static constexpr size_t [volume](#) ()
Get the total data number.

Public Attributes

- [VectorArray](#) [pos](#)
Array of position of the particles. Element is 3D vector.
- [VectorArray](#) [vel](#)
Array of velocity of the particles. Element is 3D vector.
- [VectorArray](#) [auxiVel](#)
Array of auxiliary velocity of the particles. Element is 3D vector.
- [Scalar](#) [time](#) {0.0}
The physical time of the dynamic system.
- [Scalar](#) [bindE](#) {0.0}
The binding energy(for regularization) of the dynamic system.
- [Scalar](#) [omega](#) {0.0}
The regularization variable of the dynamic system.

7.8.1 Detailed Description

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

Class of velocity dependent dynamical system with regularization variables.

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

7.8.2 Member Typedef Documentation

7.8.2.1 IndexArray

```
template<typename DataType , size_t N>
typedef std::array<size_t, N> GAR< DataType, N >::IndexArray
```

7.8.2.2 Scalar

```
template<typename DataType , size_t N>
typedef DataType GAR< DataType, N >::Scalar
```

7.8.2.3 ScalarArray

```
template<typename DataType , size_t N>
typedef std::array<Scalar, N> GAR< DataType, N >::ScalarArray
```

7.8.2.4 Vector

```
template<typename DataType , size_t N>
typedef vec3<Scalar> GAR< DataType, N >::Vector
```

7.8.2.5 VectorArray

```
template<typename DataType , size_t N>
typedef std::array<vec3<Scalar>, N> GAR< DataType, N >::VectorArray
```

7.8.3 Member Function Documentation

7.8.3.1 array()

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

Transfer this class to a plain array.

Returns

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

7.8.3.2 getOmega()

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

Calculate the regularization variable omega.

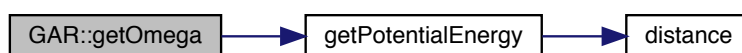
Parameters

<i>mass</i>	The mass of particles, might be required for calculation.
-------------	---

Returns

The calculated value of omega.

Here is the call graph for this function:



Here is the caller graph for this function:



7.8.3.3 initAddiVariable()

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

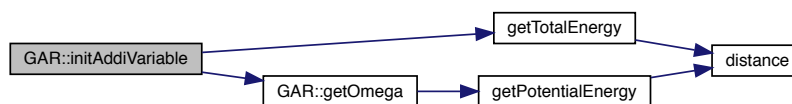
Initialize extra user defined variables. Interface required for other class.

Initialize regularizaiton variable bindE and omega.

Parameters

<i>mass</i>	The mass of particles, might be required for initialization.
-------------	--

Here is the call graph for this function:



7.8.3.4 moveToCentralMassCoords()

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

Move particles to central mass coordinates.

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

Parameters

<i>mass</i>	Mass of the particles required for moving.
-------------	--

Here is the call graph for this function:



7.8.3.5 setZero()

```
template<typename DataType , size_t N>
void GAR< DataType, N >::setZero ( ) [inline]
```

Set all data to be zero.

Here is the call graph for this function:



7.8.3.6 size()

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

Get the number of the particles.

Returns

The particle number.

7.8.3.7 toCartesian()

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

Transfer chain coordinate regularization system to Cartesian regularization system.

Coordinate transformation. From chain to Cartesian. See details in https://link.springer.com/article/10.1007%2F978-1-4939-9831-4_10.

Parameters

<i>cartesian</i>	The destination regularization system in Cartesian coordinates.
<i>index</i>	The mapping index between Cartesian coordinates and chain coordinates.

Here is the call graph for this function:



7.8.3.8 toChain()

```

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

Transfer Cartesian coordinate regularization system to chain regularization system.

Coordinate transformation. From Cartesian to chain. See details in <https://link.springer.com/article/10.1007%2F00695714>.

Parameters

<i>chainData</i>	The destination regularization system in chain coordinates.
<i>index</i>	The mapping index between Cartesian coordinates and chain coordinates.

Here is the call graph for this function:



7.8.3.9 volume()

```

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

Get the total data number.

Returns

The data number.

Here is the caller graph for this function:



7.8.4 Member Data Documentation

7.8.4.1 auxiVel

```

template<typename DataType , size_t N>
VectorArray GAR< DataType, N >::auxiVel

```

Array of auxiliary velocity of the particles. Element is 3D vector.

7.8.4.2 bindE

```

template<typename DataType , size_t N>
Scalar GAR< DataType, N >::bindE {0.0}

```

The binding energy(for regularization) of the dynamic system.

7.8.4.3 omega

```

template<typename DataType , size_t N>
Scalar GAR< DataType, N >::omega {0.0}

```

The regularization variable of the dynamic system.

7.8.4.4 pos

```

template<typename DataType , size_t N>
VectorArray GAR< DataType, N >::pos

```

Array of position of the particles. Element is 3D vector.

7.8.4.5 time

```
template<typename DataType , size_t N>
Scalar GAR< DataType, N >::time {0.0}
```

The physical time of the dynamic system.

7.8.4.6 vel

```
template<typename DataType , size_t N>
VectorArray GAR< DataType, N >::vel
```

Array of velocity of the particles. Element is 3D vector.

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

- particleSystem/[GAR.h](#)

7.9 logH< DynamicState > Class Template Reference

[logH](#) extention algorithmatic regularization interface

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

Public Types

- typedef DynamicState::Scalar [Scalar](#)

Public Member Functions

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

Static Public Member Functions

- static constexpr size_t [size](#) ()

7.9.1 Detailed Description

```
template<typename DynamicState>
class logH< DynamicState >
```

[logH](#) extention algorithmatic regularization interface

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

7.9.2 Member Typedef Documentation

7.9.2.1 Scalar

```
template<typename DynamicState >
typedef DynamicState::Scalar logH< DynamicState >::Scalar
```

7.9.3 Member Function Documentation

7.9.3.1 getPhysicalPosTime()

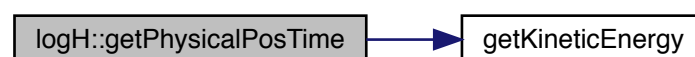
```
template<typename DynamicState >
Scalar logH< DynamicState >::getPhysicalPosTime (
    std::array< Scalar, size()> & mass,
    DynamicState & dyn,
    Scalar stepSize ) [inline]
```

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

Parameters

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

Here is the call graph for this function:



7.9.3.2 getPhysicalVelTime()

```
template<typename DynamicState >
Scalar logH< DynamicState >::getPhysicalVelTime (
    std::array< Scalar, size()> & mass,
```

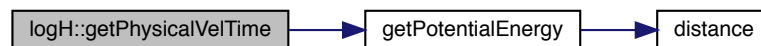
```
DynamicState & dyn,  
Scalar stepSize ) [inline]
```

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

Parameters

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

Here is the call graph for this function:



7.9.3.3 size()

```

template<typename DynamicState >
static constexpr size_t logH< DynamicState >::size ( ) [inline], [static]

```

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

- particleSystem/[regularization.h](#)

7.10 Newtonian< Scalar > Class Template Reference

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

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

Public Member Functions

- void [operator\(\)](#) ()

7.10.1 Detailed Description

```

template<typename Scalar>
class Newtonian< Scalar >

```

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

7.10.2 Member Function Documentation

7.10.2.1 operator()

```
template<typename Scalar >
void Newtonian< Scalar >::operator() ( ) [inline]
```

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

- [interaction/interaction.h](#)

7.11 chain::Node< Scalar > Struct Template Reference

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

```
#include <chain.h>
```

Public Attributes

- Scalar [Rij](#)
- size_t [i](#)
- size_t [j](#)
- bool [available](#)

7.11.1 Detailed Description

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

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

7.11.2 Member Data Documentation

7.11.2.1 available

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

State of node. If this node can be chained.

7.11.2.2 i

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

Particle index.

7.11.2.3 j

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

Particle index.

7.11.2.4 Rij

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

Relative distance of two particles.

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

- particleSystem/[chain.h](#)

7.12 NoRegu< DynamicState > Class Template Reference

Ordinary algorithmatic regularization interface.

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

Public Types

- typedef DynamicState::Scalar [Scalar](#)

Public Member Functions

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

Static Public Member Functions

- static constexpr size_t [size](#) ()

7.12.1 Detailed Description

```
template<typename DynamicState>
class NoRegu< DynamicState >
```

Ordinary algorithmatic regularization interface.

No regularization.

7.12.2 Member Typedef Documentation

7.12.2.1 Scalar

```
template<typename DynamicState >
typedef DynamicState::Scalar NoRegu< DynamicState >::Scalar
```

7.12.3 Member Function Documentation

7.12.3.1 getPhysicalPosTime()

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

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

Parameters

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

7.12.3.2 getPhysicalVelTime()

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

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

Parameters

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

7.12.3.3 size()

```
template<typename DynamicState >
static constexpr size_t NoRegu< DynamicState >::size ( ) [inline], [static]
```

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

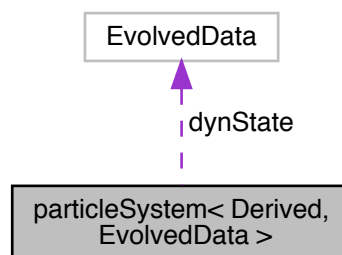
- particleSystem/regularization.h

7.13 particleSystem< Derived, EvolvedData > Class Template Reference

Base class of particle System.

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

Collaboration diagram for particleSystem< Derived, EvolvedData >:



Public Types

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

Public Member Functions

- [particleSystem](#) ()
Default construction.
- virtual [~particleSystem](#) ()
Virtualize default destructor.
- [PlainArray](#) & [array](#) ()
Transfer evolved data to a plain array.
- [Scalar](#) [timeScale](#) ([Scalar](#) scale)

- *Interface to rescale the time.*
- void `load (PlainArray &data)`
Load data from a plain array.
- `std::ostream & write (std::ostream &) const`
Output data to standard c++ ostream.
- `std::istream & read (std::istream &)`
Input data from standard c++ istream.
- const `particleSystem & operator= (const particleSystem &other)`
Overload operator =.

Static Public Member Functions

- static constexpr size_t `size ()`
Get the number of the particles.
- static constexpr size_t `volume ()`
Get the total dynamic scalar number.

Public Attributes

- EvolvedData `dynState`
Evolved variables class.
- `VectorArray & pos`
Position array interface. Reference to dynState.pos.
- `VectorArray & vel`
Velocity array interface. Reference to dynState.vel.
- `Scalar & time`
Physical time scalar interface. Reference to dynState.time.
- `ScalarArray mass`
Mass array.
- `ScalarArray radius`
Radius array.
- `IntArray type`
Particle type array.

Protected Attributes

- `VectorArray acc`
Acceleration array used to update velocity.

7.13.1 Detailed Description

```
template<typename Derived, typename EvolvedData>
class particleSystem< Derived, EvolvedData >
```

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. Here we use CRTP technique to bind derived class method. See more details in https://en.wikipedia.org/wiki/Curiously_recurring_template_pattern.

7.13.2 Member Typedef Documentation

7.13.2.1 `IntArray`

```
template<typename Derived, typename EvolvedData>
typedef std::array<size_t, EvolvedData::size()> particleSystem< Derived, EvolvedData >↵
::IntArray
```

7.13.2.2 `PlainArray`

```
template<typename Derived, typename EvolvedData>
typedef std::array<Scalar, EvolvedData::volume()> particleSystem< Derived, EvolvedData >↵
::PlainArray
```

7.13.2.3 `Scalar`

```
template<typename Derived, typename EvolvedData>
typedef EvolvedData::Scalar particleSystem< Derived, EvolvedData >::Scalar
```

7.13.2.4 `ScalarArray`

```
template<typename Derived, typename EvolvedData>
typedef EvolvedData::ScalarArray particleSystem< Derived, EvolvedData >::ScalarArray
```

7.13.2.5 `Vector`

```
template<typename Derived, typename EvolvedData>
typedef EvolvedData::Vector particleSystem< Derived, EvolvedData >::Vector
```

7.13.2.6 `VectorArray`

```
template<typename Derived, typename EvolvedData>
typedef EvolvedData::VectorArray particleSystem< Derived, EvolvedData >::VectorArray
```

7.13.3 Constructor & Destructor Documentation

7.13.3.1 `particleSystem()`

```
template<typename Derived, typename EvolvedData>
particleSystem< Derived, EvolvedData >::particleSystem ( ) [inline]
```

Default construction.

Default constructor. Bind position, velocity and time interface reference to dynState.pos, dynState.vel and dyn.time.

7.13.3.2 `~particleSystem()`

```
template<typename Derived, typename EvolvedData>
virtual particleSystem< Derived, EvolvedData >::~~particleSystem ( ) [inline], [virtual]
```

Virtualize default destructor.

7.13.4 Member Function Documentation

7.13.4.1 `array()`

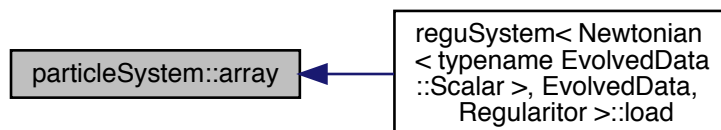
```
template<typename Derived, typename EvolvedData>
PlainArray& particleSystem< Derived, EvolvedData >::array ( ) [inline]
```

Transfer evolved data to a plain array.

Returns

The reference of a plain array.

Here is the caller graph for this function:



7.13.4.2 `load()`

```
template<typename Derived , typename EvolvedData >
void particleSystem< Derived, EvolvedData >::load (
    PlainArray & data )
```

Load data from a plain array.

Interface used by integrator and ODE iterator. Load data from a plain array processed by itegrator and iterator. Derived class could overload this function to additional process.

Parameters

<i>data</i>	Plain scalar array.
-------------	---------------------

7.13.4.3 `operator=()`

```
template<typename Derived , typename EvolvedData >
const particleSystem< Derived, EvolvedData > & particleSystem< Derived, EvolvedData >::operator=
(
    const particleSystem< Derived, EvolvedData > & other )
```

Overload operator =.

7.13.4.4 `read()`

```
template<typename Derived , typename EvolvedData >
std::istream & particleSystem< Derived, EvolvedData >::read (
    std::istream & input )
```

Input data from standard c++ istream.

Implement of CRTP '>>' method.

7.13.4.5 `size()`

```
template<typename Derived, typename EvolvedData>
static constexpr size_t particleSystem< Derived, EvolvedData >::size ( ) [inline], [static]
```

Get the number of the particles.

Returns

The particle number.

7.13.4.6 `timeScale()`

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

Interface to rescale the time.

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

Returns

The physical time.

7.13.4.7 volume()

```
template<typename Derived, typename EvolvedData>
static constexpr size_t particleSystem< Derived, EvolvedData >::volume ( ) [inline], [static]
```

Get the total dynamic scalar number.

Returns

The dynamic scalar number.

7.13.4.8 write()

```
template<typename Derived , typename EvolvedData >
std::ostream & particleSystem< Derived, EvolvedData >::write (
    std::ostream & output ) const
```

Output data to standard c++ ostream.

Implement of CRTP '<<' method.

7.13.5 Member Data Documentation

7.13.5.1 acc

```
template<typename Derived, typename EvolvedData>
VectorArray particleSystem< Derived, EvolvedData >::acc [protected]
```

Acceleration array used to update velocity.

7.13.5.2 dynState

```
template<typename Derived, typename EvolvedData>
EvolvedData particleSystem< Derived, EvolvedData >::dynState
```

Evolved variables class.

7.13.5.3 mass

```
template<typename Derived, typename EvolvedData>
ScalarArray particleSystem< Derived, EvolvedData >::mass
```

Mass array.

7.13.5.4 pos

```
template<typename Derived, typename EvolvedData>
VectorArray< particleSystem< Derived, EvolvedData >::pos
```

Position array interface. Reference to dynState.pos.

7.13.5.5 radius

```
template<typename Derived, typename EvolvedData>
ScalarArray particleSystem< Derived, EvolvedData >::radius
```

Radius array.

7.13.5.6 time

```
template<typename Derived, typename EvolvedData>
Scalar& particleSystem< Derived, EvolvedData >::time
```

Physical time scalar interface. Reference to dynState.time.

7.13.5.7 type

```
template<typename Derived, typename EvolvedData>
IntArray particleSystem< Derived, EvolvedData >::type
```

Particle type array.

7.13.5.8 vel

```
template<typename Derived, typename EvolvedData>
VectorArray< particleSystem< Derived, EvolvedData >::vel
```

Velocity array interface. Reference to dynState.vel.

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

- [particleSystem.h](#)

7.14 PN1th< Scalar > Class Template Reference

Post newtonian pair interaction functor(c++ std11)

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

Public Member Functions

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

Update the velocity dependent acceleration of particle 1 and 2.

Private Types

- typedef `vec3`< Scalar > `Vector`

7.14.1 Detailed Description

```
template<typename Scalar>
class PN1th< Scalar >
```

Post newtonian pair interaction functor(c++ std11)

7.14.2 Member Typedef Documentation

7.14.2.1 Vector

```
template<typename Scalar >
typedef vec3<Scalar> PN1th< Scalar >::Vector [private]
```

7.14.3 Member Function Documentation

7.14.3.1 `operator()`

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

Update the velocity dependent acceleration of particle 1 and 2.

Parameters

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

Here is the call graph for this function:



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

- [interaction/interaction.h](#)

7.15 reguDynamics< DataType, N > Class Template Reference

Class of dynamical system with regularization variables.

```
#include <regularState.h>
```

Public Types

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

Public Member Functions

- `std::array< Scalar, volume()> & array ()`
Transfer this class to a plain array.
- `void setZero ()`
Set all data to be zero.
- `Scalar getOmega (ScalarArray &mass)`
Calculate the regularization variable omega.
- `void initAddiVariable (ScalarArray &mass)`
Initialize extra user defined variables. Interface required for other class.
- `void toChain (reguDynamics &chainData, IndexArray &index)`
Transfer Cartesian coordinate regularization system to chain regularization system.
- `void toCartesian (reguDynamics &cartesian, IndexArray &index)`
Transfer chain coordinate regularization system to Cartesian regularization system.
- `void moveToCentralMassCoords (ScalarArray &mass)`
Move particles to central mass coordinates.

Static Public Member Functions

- static constexpr size_t `size` ()
Get the number of the particles.
- static constexpr size_t `volume` ()
Get the total data number.

Public Attributes

- `VectorArray` `pos`
Array of position of the particles. Element is 3D vector.
- `VectorArray` `vel`
Array of velocity of the particles. Element is 3D vector.
- `Scalar` `time` {0.0}
The physical time of the dynamic system.
- `Scalar` `bindE` {0.0}
The binding energy(for regularization) of the dynamic system.
- `Scalar` `omega` {0.0}
The regularization variable of the dynamic system.

7.15.1 Detailed Description

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

Class of dynamical system with regularization variables.

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

7.15.2 Member Typedef Documentation

7.15.2.1 IndexArray

```
template<typename DataType , size_t N>
typedef std::array<size_t, N> reguDynamics< DataType, N >::IndexArray
```

7.15.2.2 Scalar

```
template<typename DataType , size_t N>
typedef DataType reguDynamics< DataType, N >::Scalar
```

7.15.2.3 ScalarArray

```
template<typename DataType , size_t N>
typedef std::array<Scalar, N> reguDynamics< DataType, N >::ScalarArray
```

7.15.2.4 Vector

```
template<typename DataType , size_t N>
typedef vec3<Scalar> reguDynamics< DataType, N >::Vector
```

7.15.2.5 VectorArray

```
template<typename DataType , size_t N>
typedef std::array<vec3<Scalar>, N> reguDynamics< DataType, N >::VectorArray
```

7.15.3 Member Function Documentation

7.15.3.1 array()

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

Transfer this class to a plain array.

Returns

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

7.15.3.2 getOmega()

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

Calculate the regularization variable omega.

Parameters

<i>mass</i>	The mass of particles, might be required for calculation.
-------------	---

Returns

The calculated value of omega.

Here is the call graph for this function:



Here is the caller graph for this function:



7.15.3.3 initAddiVariable()

```

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

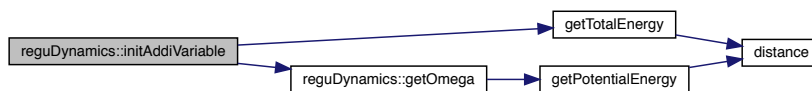
Initialize extra user defined variables. Interface required for other class.

Initialize regularizaiton variable bindE and omega.

Parameters

<i>mass</i>	The mass of particles, might be required for initialization.
-------------	--

Here is the call graph for this function:



7.15.3.4 moveToCentralMassCoords()

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

Move particles to central mass coordinates.

Move position and velocity to central mass coordinates.

Parameters

<i>mass</i>	Mass of the particles required for moving.
-------------	--

Here is the call graph for this function:



7.15.3.5 setZero()

```
template<typename DataType , size_t N>
void reguDynamics< DataType, N >::setZero ( ) [inline]
```

Set all data to be zero.

Here is the call graph for this function:



7.15.3.6 size()

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

Get the number of the particles.

Returns

The particle number.

7.15.3.7 toCartesian()

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

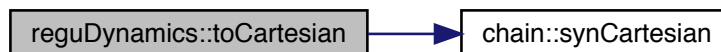
Transfer chain coordinate regularization system to Cartesian regularization system.

Coordinate transformation. From chain to Cartesian. See details in <https://link.springer.com/article/10.1007%2F00695714>.

Parameters

<i>cartesian</i>	The destination regularization system in Cartesian coordinates.
<i>index</i>	The mapping index between Cartesian coordinates and chain coordinates.

Here is the call graph for this function:



7.15.3.8 toChain()

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

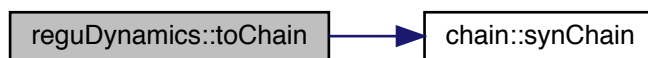
Transfer Cartesian coordinate regularization system to chain regularization system.

Coordinate transformation. From Cartesian to chain. See details in <https://link.springer.com/article/10.1007%2F00695714>.

Parameters

<i>chainData</i>	The destination regularization system in chain coordinates.
<i>index</i>	The mapping index between Cartesian coordinates and chain coordinates.

Here is the call graph for this function:



7.15.3.9 volume()

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

Get the total data number.

Returns

The data number.

Here is the caller graph for this function:



7.15.4 Member Data Documentation

7.15.4.1 bindE

```
template<typename DataType , size_t N>
Scalar reguDynamics< DataType, N >::bindE {0.0}
```

The binding energy(for regularization) of the dynamic system.

7.15.4.2 omega

```
template<typename DataType , size_t N>
Scalar reguDynamics< DataType, N >::omega {0.0}
```

The regularization variable of the dynamic system.

7.15.4.3 pos

```
template<typename DataType , size_t N>
VectorArray reguDynamics< DataType, N >::pos
```

Array of position of the particles. Element is 3D vector.

7.15.4.4 time

```
template<typename DataType , size_t N>
Scalar reguDynamics< DataType, N >::time {0.0}
```

The physical time of the dynamic system.

7.15.4.5 vel

```
template<typename DataType , size_t N>
VectorArray reguDynamics< DataType, N >::vel
```

Array of velocity of the particles. Element is 3D vector.

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

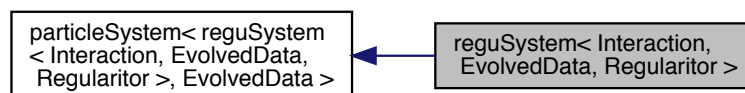
- [particleSystem/regularState.h](#)

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

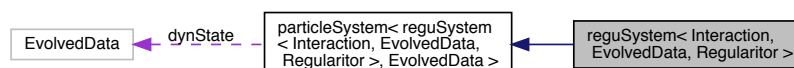
Regularized particle System.

```
#include <reguSystem.h>
```

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



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



Public Types

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

Public Member Functions

- void [advancePos](#) ([Scalar](#) timeStepSize)
Advance position one step with current velocity.
- void [advanceVel](#) ([Scalar](#) timeStepSize)
Advance velocity one step with current acceleration.
- const [reguSystem](#) & [operator=](#) (const [reguSystem](#) &other)
Overload operator =.
- std::istream & [read](#) (std::istream &)
Input data from standard c++ istream.
- void [load](#) ([PlainArray](#) &data)
Load data from a plain array.
- [Scalar](#) [timeScale](#) ([Scalar](#) scale)
Interface to rescale the time.

Static Public Member Functions

- static constexpr size_t [size](#) ()
Get the number of the particles.
- static constexpr size_t [volume](#) ()
Get the total dynamic scalar number.

Private Member Functions

- void [advanceOmega](#) ([Scalar](#) stepSize)
Advance regularization variable omega.
- void [advanceB](#) ([Scalar](#) stepSize)
Advance regularization variable bindE.
- void [kickVel](#) ([Scalar](#) stepSize)
Advance velocity with current acceleration.
- void [kickAuxiVel](#) ([Scalar](#) stepSize)
Advance auxiliar velocity with current acceleration.
- void [updateAccWith](#) ([VectorArray](#) &vel)
Update velocity dependent acceleration with given velocity. Then update the total acceleration.
- void [updateVelIndepAcc](#) ()
Update velocity independent acceleration.

Private Attributes

- [VectorArray velIndepAcc](#)
Velocity independent acceleration array.
- [VectorArray velDepAcc](#)
Velocity dependent acceleration array.
- Interaction [velDepForce](#)
Velocity dependent pair force functor.
- Regularitor [regular](#)
Regularization interface.

Additional Inherited Members

7.16.1 Detailed Description

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
class reguSystem< Interaction, EvolvedData, 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> .

7.16.2 Member Typedef Documentation

7.16.2.1 PlainArray

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
typedef std::array<Scalar, EvolvedData::volume()> reguSystem< Interaction, EvolvedData, Regularitor
>::PlainArray
```

7.16.2.2 Scalar

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
typedef EvolvedData::Scalar reguSystem< Interaction, EvolvedData, Regularitor >::Scalar
```

7.16.2.3 ScalarArray

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
typedef EvolvedData::ScalarArray reguSystem< Interaction, EvolvedData, Regularitor >::ScalarArray
```

7.16.2.4 Vector

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
typedef EvolvedData::Vector reguSystem< Interaction, EvolvedData, Regularitor >::Vector
```

7.16.2.5 VectorArray

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
typedef EvolvedData::VectorArray reguSystem< Interaction, EvolvedData, Regularitor >::VectorArray
```

7.16.3 Member Function Documentation

7.16.3.1 advanceB()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void reguSystem< Interaction, EvolvedData, Regularitor >::advanceB (
    Scalar stepSize ) [private]
```

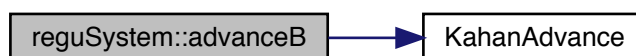
Advance regularization variable bindE.

Advance bindE with velocity dependent acceleration and auxiliar velocity with physical time step size.

Parameters

<i>stepSize</i>	Physical time step.
-----------------	---------------------

Here is the call graph for this function:



7.16.3.2 advanceOmega()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void reguSystem< Interaction, EvolvedData, Regularitor >::advanceOmega (
    Scalar stepSize ) [private]
```

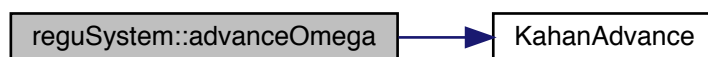
Advance regularization variable omega.

Advance omega with velocity independent acceleration and auxiliar velocity with physical time step size.

Parameters

<i>stepSize</i>	Physical time step.
-----------------	---------------------

Here is the call graph for this function:



7.16.3.3 advancePos()

```

template<typename Interaction , typename EvolvedData , typename Regularitor >
void reguSystem< Interaction, EvolvedData, Regularitor >::advancePos (
    Scalar timeStepSize )
  
```

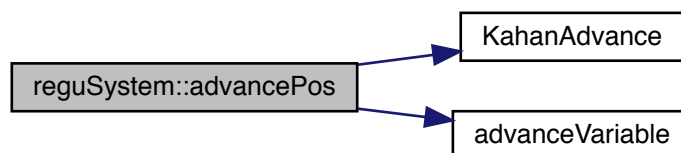
Advance position one step with current velocity.

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

Parameters

<i>timeStepSize</i>	Integration step size, will be transferred to physical time in the function.
---------------------	--

Here is the call graph for this function:



7.16.3.4 advanceVel()

```

template<typename Interaction , typename EvolvedData , typename Regularitor >
void reguSystem< Interaction, EvolvedData, Regularitor >::advanceVel (
    Scalar timeStepSize )
  
```

Advance velocity one step with current acceleration.

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

Parameters

<i>timeStepSize</i>	Integration step size, will be transfered to physical time in the function.
---------------------	---

7.16.3.5 kickAuxiVel()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void reguSystem< Interaction, EvolvedData, Regularitor >::kickAuxiVel (
    Scalar stepSize ) [private]
```

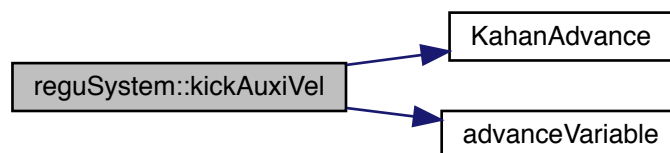
Advance auxiliar velocity with current acceleration.

Advance the auxiliar velocity with current total acceleration variable 'acc'.

Parameters

<i>stepSize</i>	Integration step size.
-----------------	------------------------

Here is the call graph for this function:



7.16.3.6 kickVel()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void reguSystem< Interaction, EvolvedData, Regularitor >::kickVel (
    Scalar stepSize ) [private]
```

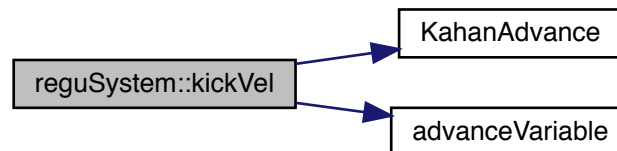
Advance velocity with current acceleration.

Advance the velocity with current total acceleration variable 'acc'.

Parameters

<i>stepSize</i>	Integration step size.
-----------------	------------------------

Here is the call graph for this function:



7.16.3.7 load()

```

template<typename Interaction , typename EvolvedData , typename Regularitor >
void reguSystem< Interaction, EvolvedData, Regularitor >::load (
    PlainArray & data )
  
```

Load data from a plain array.

Note

Overload base class `load()`.

Interface used by integrator and ODE iterator. Load data from a plain array processed by itegrator and iterator. Derived class could overload this function to additional process.

Parameters

<i>data</i>	Plain scalar array.
-------------	---------------------

7.16.3.8 operator=()

```

template<typename Interaction , typename EvolvedData , typename Regularitor >
const reguSystem< Interaction, EvolvedData, Regularitor > & reguSystem< Interaction, EvolvedData, Regularitor >::operator= (
    const reguSystem< Interaction, EvolvedData, Regularitor > & other )
  
```

Overload operator `=`.

7.16.3.9 read()

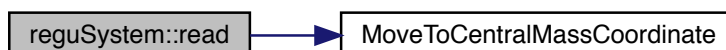
```
template<typename Interaction , typename EvolvedData , typename Regularitor >
std::istream & reguSystem< Interaction, EvolvedData, Regularitor >::read (
    std::istream & input )
```

Input data from standard c++ istream.

Note

Overload base class `read()`.

Implement of CRTP '>>' method. Here is the call graph for this function:



7.16.3.10 size()

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
static constexpr size_t reguSystem< Interaction, EvolvedData, Regularitor >::size ( ) [inline],
[static]
```

Get the number of the particles.

Returns

The particle number.

Note

Overload base class `size()`.

7.16.3.11 timeScale()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
EvolvedData::Scalar reguSystem< Interaction, EvolvedData, Regularitor >::timeScale (
    Scalar scale )
```

Interface to rescale the time.

Note

Overload base class `timeScale()`.

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

Returns

The physical time.

7.16.3.12 updateAccWith()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void reguSystem< Interaction, EvolvedData, Regularitor >::updateAccWith (
    VectorArray & velocity ) [private]
```

Update velocity dependent acceleration with given velocity. Then update the total acceleration.

Update the velocity dependent acceleration variable 'velDepAcc' with given velocity and velocity dependent pair force 'velDepForce'. Then update the total acceleration 'acc' by adding 'velIndepAcc' and 'velDepAcc'.

7.16.3.13 updateVelIndepAcc()

```
template<typename Interaction , typename EvolvedData , typename Regularitor >
void reguSystem< Interaction, EvolvedData, Regularitor >::updateVelIndepAcc ( ) [private]
```

Update velocity independent acceleration.

Update velocity independent acceleration 'velIndepAcc' with [Newtonian](#) interaction.

7.16.3.14 volume()

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
static constexpr size_t reguSystem< Interaction, EvolvedData, Regularitor >::volume ( ) [inline],
[static]
```

Get the total dynamic scalar number.

Returns

The dynamic scalar number.

Note

Overload base class [volume\(\)](#).

7.16.4 Member Data Documentation

7.16.4.1 regular

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
Regularitor reguSystem< Interaction, EvolvedData, Regularitor >::regular [private]
```

Regularization interface.

7.16.4.2 velDepAcc

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
VectorArray reguSystem< Interaction, EvolvedData, Regularitor >::velDepAcc [private]
```

Velocity dependent acceleration array.

7.16.4.3 velDepForce

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
Interaction reguSystem< Interaction, EvolvedData, Regularitor >::velDepForce [private]
```

Velocity dependent pair force functor.

7.16.4.4 velIndepAcc

```
template<typename Interaction, typename EvolvedData, typename Regularitor>
VectorArray reguSystem< Interaction, EvolvedData, Regularitor >::velIndepAcc [private]
```

Velocity independent acceleration array.

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

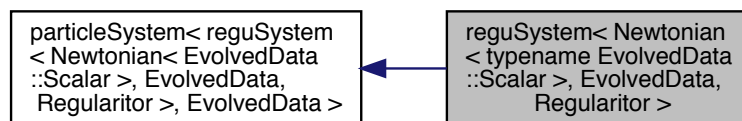
- particleSystem/reguSystem.h

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

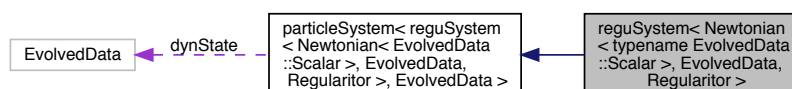
Partial specialization of [reguSystem](#) for velocity independent system.

```
#include <reguSystem.h>
```

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



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



Public Types

- typedef EvolvedData::Scalar [Scalar](#)
- typedef EvolvedData::Vector [Vector](#)
- typedef EvolvedData::VectorArray [VectorArray](#)
- typedef EvolvedData::ScalarArray [ScalarArray](#)
- typedef `std::array< Scalar, EvolvedData::volume()>` [PlainArray](#)

Public Member Functions

- void [advancePos](#) ([Scalar](#) timeStepSize)
Advance position one step with current velocity.
- void [advanceVel](#) ([Scalar](#) timeStepSize)
Advance velocity one step with current acceleration.
- `std::istream & read (std::istream &)`
Input data from standard c++ istream.
- void [load](#) ([PlainArray](#) &data)
Load data from a plain array.
- [Scalar](#) [timeScale](#) ([Scalar](#) scale)
Interface to rescale the time.

Static Public Member Functions

- static constexpr size_t [size](#) ()
Get the number of the particles.
- static constexpr size_t [volume](#) ()
Get the total dynamic scalar number.

Private Member Functions

- void [advanceOmega](#) ([Scalar](#) stepSize)
Advance regularization variable omega.
- void [updateVelIndepAcc](#) ()
Update velocity independent acceleration.

Private Attributes

- Regularitor [regular](#)
Regularization interface.

Additional Inherited Members

7.17.1 Detailed Description

```
template<typename EvolvedData, typename Regularitor>
class reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >
```

Partial specialization of [reguSystem](#) for velocity independent system.

7.17.2 Member Typedef Documentation

7.17.2.1 PlainArray

```
template<typename EvolvedData , typename Regularitor >
typedef std::array<Scalar, EvolvedData::volume()> reguSystem< Newtonian< typename EvolvedData::Scalar
>, EvolvedData, Regularitor >::PlainArray
```

7.17.2.2 Scalar

```
template<typename EvolvedData , typename Regularitor >
typedef EvolvedData::Scalar reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::Scalar
```

7.17.2.3 ScalarArray

```
template<typename EvolvedData , typename Regularitor >
typedef EvolvedData::ScalarArray reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::ScalarArray
```

7.17.2.4 Vector

```
template<typename EvolvedData , typename Regularitor >
typedef EvolvedData::Vector reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::Vector
```

7.17.2.5 VectorArray

```
template<typename EvolvedData , typename Regularitor >
typedef EvolvedData::VectorArray reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::VectorArray
```

7.17.3 Member Function Documentation

7.17.3.1 advanceOmega()

```
template<typename EvolvedData , typename Regularitor >
void reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::advanceOmega (
    Scalar stepSize ) [private]
```

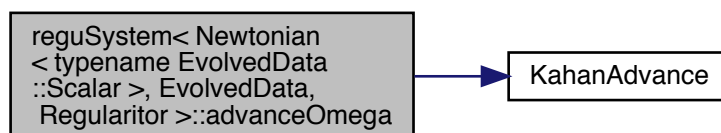
Advance regularization variable omega.

Advance omega with velocity independent acceleration and auxiliar velocity with physical time step size.

Parameters

<i>stepSize</i>	Physical time step.
-----------------	---------------------

Here is the call graph for this function:



7.17.3.2 advancePos()

```

template<typename EvolvedData , typename Regularitor >
void reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >↔
::advancePos (
    Scalar timeStepSize )
  
```

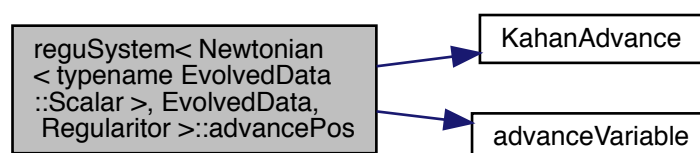
Advance position one step with current velocity.

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

Parameters

<i>timeStepSize</i>	Integration step size, will be transfered to physical time in the function.
---------------------	---

Here is the call graph for this function:



7.17.3.3 advanceVel()

```
template<typename EvolvedData , typename Regularitor >
void reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::
::advanceVel (
    Scalar timeStepSize )
```

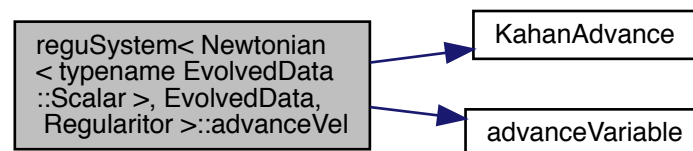
Advance velocity one step with current acceleration.

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

Parameters

<i>timeStepSize</i>	Integration step size, will be transfered to physical time in the function.
---------------------	---

Here is the call graph for this function:



7.17.3.4 load()

```
template<typename EvolvedData , typename Regularitor >
void reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >::load
(
    PlainArray & data )
```

Load data from a plain array.

Note

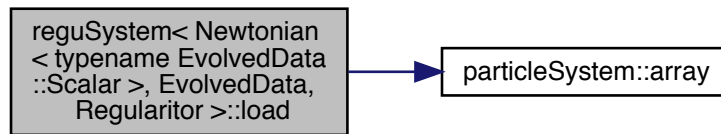
Overload base class `load()`.

Interface used by integrator and ODE iterator. Load data from a plain array processed by itegrator and iterator. Derived class could overload this function to additional process.

Parameters

<i>data</i>	Plain scalar array.
-------------	---------------------

Here is the call graph for this function:



7.17.3.5 read()

```

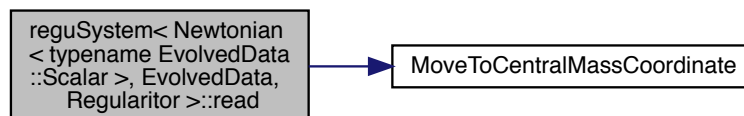
template<typename EvolvedData , typename Regularitor >
std::istream & reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor
>::read (
    std::istream & input )
  
```

Input data from standard c++ istream.

Note

Overload base class `read()`.

Implement of CRTP '>>' method. Here is the call graph for this function:



7.17.3.6 size()

```

template<typename EvolvedData , typename Regularitor >
static constexpr size_t reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData,
Regularitor >::size ( ) [inline], [static]
  
```

Get the number of the particles.

Returns

The particle number.

Note

Overload base class `size()`.

7.17.3.7 timeScale()

```
template<typename EvolvedData , typename Regularitor >
EvolvedData::Scalar reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor
>::timeScale (
    Scalar scale )
```

Interface to rescale the time.

Note

Overload base class [timeScale\(\)](#).

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

Returns

The physical time.

7.17.3.8 updateVelIndepAcc()

```
template<typename EvolvedData , typename Regularitor >
void reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >↔
::updateVelIndepAcc ( ) [private]
```

Update velocity independent acceleration.

Update velocity independent acceleration 'acc' with [Newtonian](#) interaction.

7.17.3.9 volume()

```
template<typename EvolvedData , typename Regularitor >
static constexpr size_t reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData,
Regularitor >::volume ( ) [inline], [static]
```

Get the total dynamic scalar number.

Returns

The dynamic scalar number.

Note

Overload base class [volume\(\)](#).

7.17.4 Member Data Documentation

7.17.4.1 regular

```
template<typename EvolvedData , typename Regularitor >
Regularitor reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor
>::regular [private]
```

Regularization interface.

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

- [particleSystem/reguSystem.h](#)

7.18 symplectic10th< ParticSys > Class Template Reference

Tenth order symplectic integrator.

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

Public Member Functions

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

Static Public Attributes

- static const int [order](#) {10}
Order of the integrator.

7.18.1 Detailed Description

```
template<typename ParticSys>
class symplectic10th< ParticSys >
```

Tenth order symplectic integrator.

7.18.2 Member Function Documentation

7.18.2.1 integrate()

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

Interface to integrate particle system.

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

Parameters

<i>particles</i>	Particle system need to be integrated.
<i>stepLength</i>	Step size for integration.

7.18.3 Member Data Documentation

7.18.3.1 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](#)

7.19 symplectic2th< ParticSys > Class Template Reference

Second order symplectic integrator.

```
#include <symplectic2th.h>
```

Public Member Functions

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

Static Public Attributes

- static const int [order](#) {2}
Order of the integrator.

Private Types

- typedef ParticSys::Scalar [Scalar](#)

7.19.1 Detailed Description

```
template<typename ParticSys>
class symplectic2th< ParticSys >
```

Second order symplectic integrator.

7.19.2 Member Typedef Documentation

7.19.2.1 Scalar

```
template<typename ParticSys >
typedef ParticSys::Scalar symplectic2th< ParticSys >::Scalar [private]
```

7.19.3 Member Function Documentation

7.19.3.1 [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

<i>particles</i>	Particle system need to be integrated.
<i>stepLength</i>	Step size for integration.

7.19.4 Member Data Documentation

7.19.4.1 [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](#)

7.20 [symplectic4th](#)< ParticSys > Class Template Reference

Fourth order symplectic integrator.

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

Public Member Functions

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

Static Public Attributes

- static const int [order](#) {4}
Order of the integrator.

7.20.1 Detailed Description

```
template<typename ParticSys>
class symplectic4th< ParticSys >
```

Fourth order symplectic integrator.

7.20.2 Member Function Documentation

7.20.2.1 integrate()

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

Interface to integrate particle system.

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

Parameters

<i>particles</i>	Particle system need to be integrated.
<i>stepLength</i>	Step size for integration.

7.20.3 Member Data Documentation

7.20.3.1 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

7.21 symplectic6th< ParticSys > Class Template Reference

Sixth order symplectic integrator.

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

Static Public Attributes

- static const int `order` {6}
Order of the integrator.

Private Member Functions

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

7.21.1 Detailed Description

```
template<typename ParticSys>
class symplectic6th< ParticSys >
```

Sixth order symplectic integrator.

7.21.2 Member Function Documentation

7.21.2.1 `integrate()`

```
template<typename ParticSys >
void symplectic6th< ParticSys >::integrate (
    ParticSys & particles,
    double stepLength ) [private]
```

Interface to integrate particle system.

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

Parameters

<i>particles</i>	Particle system need to be integrated.
<i>stepLength</i>	Step size for integration.

7.21.3 Member Data Documentation

7.21.3.1 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](#)

7.22 symplectic8th< ParticSys > Class Template Reference

Eighth order symplectic integrator.

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

Public Member Functions

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

Static Public Attributes

- static const int [order](#) {8}
Order of the integrator.

7.22.1 Detailed Description

```
template<typename ParticSys>
class symplectic8th< ParticSys >
```

Eighth order symplectic integrator.

7.22.2 Member Function Documentation

7.22.2.1 integrate()

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

Interface to integrate particle system.

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

Parameters

<i>particles</i>	Particle system need to be integrated.
<i>stepLength</i>	Step size for integration.

7.22.3 Member Data Documentation

7.22.3.1 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

7.23 TTL< DynamicState > Class Template Reference

Time Transform Leapfrog algorithmatic regularization interface.

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

Public Types

- typedef DynamicState::Scalar [Scalar](#)

Public Member Functions

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

Static Public Member Functions

- static constexpr size_t [size](#) ()

7.23.1 Detailed Description

```
template<typename DynamicState>
class TTL< DynamicState >
```

Time Transform Leapfrog algorithmatic regularization interface.

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

7.23.2 Member Typedef Documentation

7.23.2.1 Scalar

```
template<typename DynamicState >
typedef DynamicState::Scalar TTL< DynamicState >::Scalar
```

7.23.3 Member Function Documentation

7.23.3.1 getPhysicalPosTime()

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

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

Parameters

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

7.23.3.2 getPhysicalVelTime()

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

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

Parameters

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

7.23.3.3 size()

```
template<typename DynamicState >
static constexpr size_t TTL< DynamicState >::size ( ) [inline], [static]
```

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

- [particleSystem/regularization.h](#)

7.24 vec3< T > Struct Template Reference

Self 3D vector class.

```
#include <vector3.h>
```

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
Division by wise.
- [vec3 operator+](#) (const double c) const
Add scalar by wise.
- [vec3 operator-](#) (const double c) const
Subtract scalar by wise.
- [vec3 operator*](#) (const double c) const
Multiply scalar by wise.
- [vec3 operator/](#) (const double c) const
Divide scalar by wise.
- [vec3 operator-](#) () const
Opposite vector.
- [vec3 operator^](#) (const [vec3](#) &v) const
Cross product.
- [vec3 abs](#) () const
Absolute value by wise.
- const [vec3](#) & [operator+=](#) (const [vec3](#) &v)
- const [vec3](#) & [operator-=](#) (const [vec3](#) &v)
- const [vec3](#) & [operator/=](#) (const [vec3](#) &v)
- const [vec3](#) & [operator+=](#) (const double c)
- const [vec3](#) & [operator-=](#) (const double c)
- const [vec3](#) & [operator*=](#) (const double c)
- const [vec3](#) & [operator/=](#) (const double c)
- const [vec3](#) & [operator=](#) (const [vec3](#) &v)

- `double operator* (const vec3 &v) const`
Inner product.
- `double norm () const`
Calculate the norm.
- `double normSquare () const`
Calculate the square of the norm.
- `double reNorm () const`
Calculate the inverse of the norm.
- `void setZero ()`

Public Attributes

- `T x`
- `T y`
- `T z`

Friends

- `vec3 operator+ (const double c, const vec3 &v)`
- `vec3 operator- (const double c, const vec3 &v)`
- `vec3 operator* (const double c, const vec3 &v)`
- `std::ostream & operator<< (std::ostream &output, const vec3 &v)`
Output to ostream.
- `std::istream & operator>> (std::istream &input, vec3 &v)`
Input from istream.

7.24.1 Detailed Description

```
template<typename T>  
struct vec3< T >
```

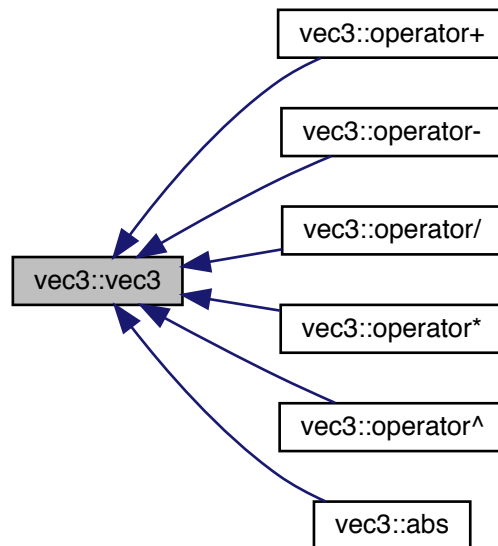
Self 3D vector class.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 `vec3()` [1/3]

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

Here is the caller graph for this function:



7.24.2.2 `vec3()` [2/3]

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

7.24.2.3 `vec3()` [3/3]

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

7.24.3 Member Function Documentation

7.24.3.1 `abs()`

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

Absolute value by wise.

Here is the call graph for this function:



7.24.3.2 `norm()`

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

Calculate the norm.

7.24.3.3 `normSquare()`

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

Calculate the square of the norm.

7.24.3.4 `operator*()` [1/2]

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

Multiply scalar by wise.

Here is the call graph for this function:



7.24.3.5 operator*() [2/2]

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

Inner product.

7.24.3.6 operator*=()

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

7.24.3.7 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:



7.24.3.8 operator+() [2/2]

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

Add scalar by wise.

Here is the call graph for this function:



7.24.3.9 `operator+=()` [1/2]

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

7.24.3.10 `operator+=()` [2/2]

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

7.24.3.11 `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:

7.24.3.12 `operator-()` [2/3]

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

Subtract scalar by wise.

Here is the call graph for this function:



7.24.3.13 operator-() [3/3]

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

Opposite vector.

Here is the call graph for this function:



7.24.3.14 operator-=() [1/2]

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

7.24.3.15 operator-=() [2/2]

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

7.24.3.16 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:



7.24.3.17 `operator/()` [2/2]

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

Divide scalar by wise.

Here is the call graph for this function:

7.24.3.18 `operator/=()` [1/2]

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

7.24.3.19 `operator/=()` [2/2]

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

7.24.3.20 `operator=()`

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

7.24.3.21 operator^()

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

Cross product.

Here is the call graph for this function:



7.24.3.22 reNorm()

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

Calculate the inverse of the norm.

Here is the caller graph for this function:



7.24.3.23 setZero()

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

7.24.4 Friends And Related Function Documentation

7.24.4.1 `operator*`

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

7.24.4.2 `operator+`

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

7.24.4.3 `operator-`

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

7.24.4.4 `operator<<`

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

Output to ostream.

7.24.4.5 `operator>>`

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

Input from istream.

7.24.5 Member Data Documentation

7.24.5.1 x

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

7.24.5.2 y

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

7.24.5.3 z

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

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

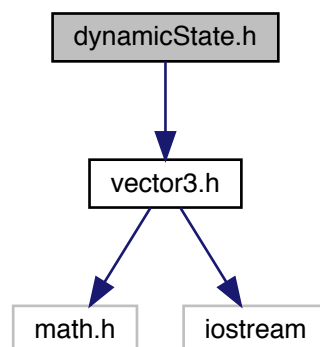
- [vector3.h](#)

8 File Documentation

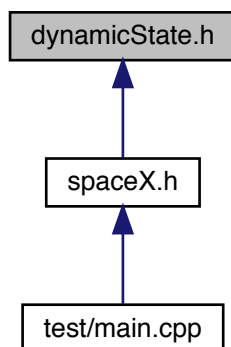
8.1 dynamicState.h File Reference

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

Include dependency graph for dynamicState.h:



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



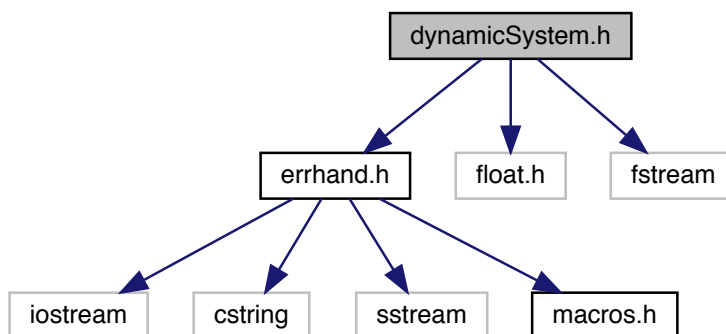
Classes

- class `dynamics< DataType, N >`
Class of dynamical variable.

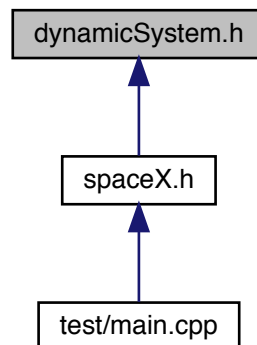
8.2 dynamicSystem.h File Reference

```
#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, Integrator, 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, Integrator< ParticSys >, ODEiterator< ParticSys, Integrator<`
`ParticSys > >>`

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

8.2.1 Typedef Documentation

8.2.1.1 `spaceX`

```

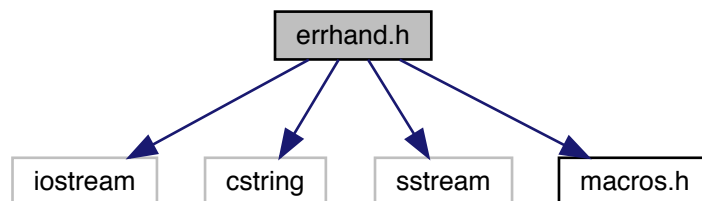
template<typename ParticSys , template< typename > class Integrator, template< typename,
typename > class ODEiterator>
using spaceX = dynamicSystem<ParticSys, Integrator<ParticSys>, ODEiterator<ParticSys, Integrator<Partic
Sys> >>
  
```

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

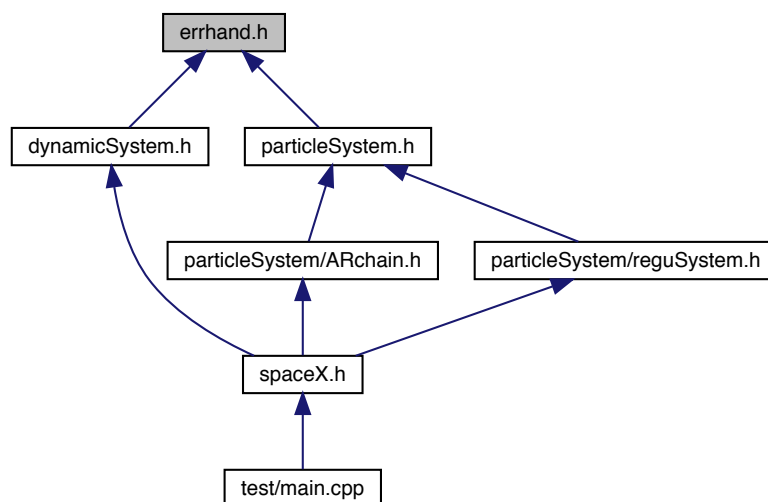
8.3 errhand.h File Reference

```
#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 = true`

8.3.1 Macro Definition Documentation

8.3.1.1 ANSI_COLOR_BLUE

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

8.3.1.2 ANSI_COLOR_CYAN

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

8.3.1.3 ANSI_COLOR_GREEN

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

8.3.1.4 ANSI_COLOR_MAGENTA

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

8.3.1.5 ANSI_COLOR_RED

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

8.3.1.6 ANSI_COLOR_RESET

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

8.3.1.7 ANSI_COLOR_YELLOW

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

8.3.1.8 NEWLINE

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

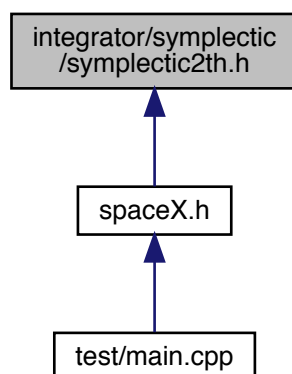
8.4 integrator/symplectic/symplectic10th.h File Reference

Classes

- class [symplectic10th< ParticSys >](#)
Tenth order symplectic integrator.

8.5 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.

8.6 integrator/symplectic/symplectic4th.h File Reference

Classes

- class `symplectic4th< ParticSys >`
Fourth order symplectic integrator.

8.7 integrator/symplectic/symplectic6th.h File Reference

Classes

- class `symplectic6th< ParticSys >`
Sixth order symplectic integrator.

8.8 integrator/symplectic/symplectic8th.h File Reference

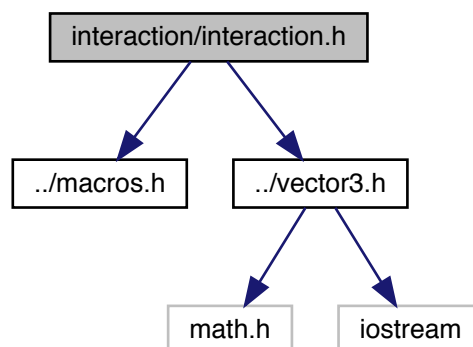
Classes

- class `symplectic8th< ParticSys >`
Eighth order symplectic integrator.

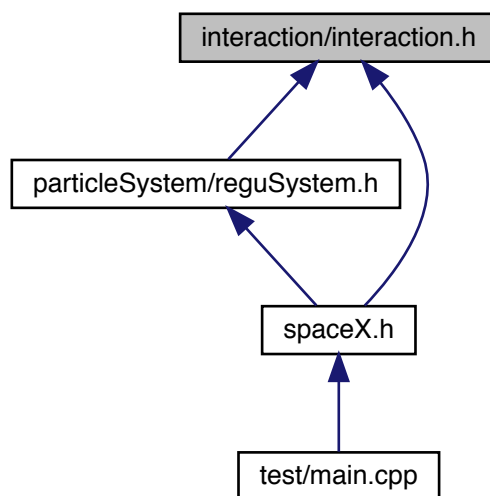
8.9 interaction/interaction.h File Reference

```
#include "../macros.h"  
#include "../vector3.h"
```

Include dependency graph for interaction.h:



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



Classes

- class [PN1th< Scalar >](#)
Post newtonian pair interaction functor(c++ std11)
- class [Newtonian< Scalar >](#)
Marker of None velocity dependent force functor(c++ std11)

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

8.9.1 Variable Documentation

8.9.1.1 INV_C

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

8.9.1.2 INV_C2

```
constexpr double INV_C2 = INV_C * INV_C
```

8.9.1.3 INV_C3

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

8.9.1.4 INV_C4

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

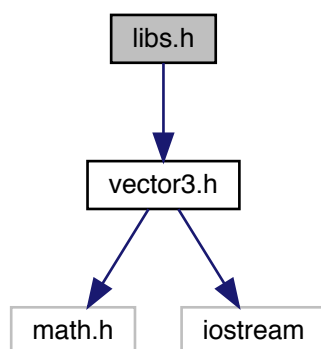
8.9.1.5 INV_C5

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

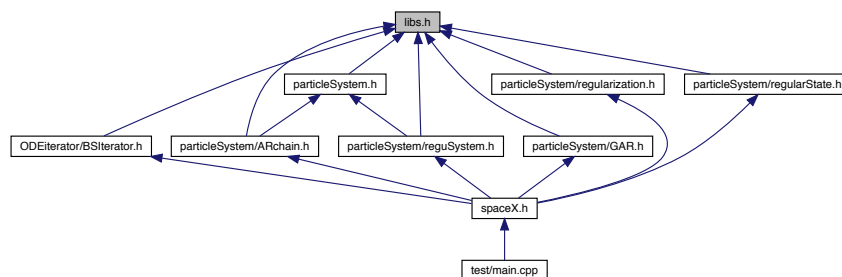
8.10 libs.h File Reference

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

Include dependency graph for libs.h:



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



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<class T >
void **swap** (T &a, T &b)

 Self swap()
- template<typename Scalar , size_t N>
void **KahanAdvance** (std::array< **vec3**< Scalar >, N > &var, const std::array< **vec3**< Scalar >, N >
&increase, std::array< **vec3**< Scalar >, N > &err, Scalar dt)

 Kahan Summation for Array.
- template<typename Scalar >
void **KahanAdvance** (Scalar &var, const Scalar increase, Scalar &err)

 Kahan Summation for Scalar.
- template<typename Scalar , size_t N>
void **advanceVariable** (std::array< **vec3**< Scalar >, N > &var, const std::array< **vec3**< Scalar >, N >
&add, Scalar dt)

 Normal Summation of two Arrays.
- template<typename Scalar , size_t N>
void **MoveToCentralMassCoordinate** (const std::array< Scalar, N > &mass, std::array< **vec3**< Scalar >, N
> &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

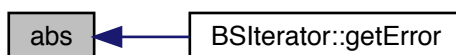
8.10.1 Function Documentation

8.10.1.1 abs()

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

Self [abs\(\)](#)

Here is the caller graph for this function:



8.10.1.2 advanceVariable()

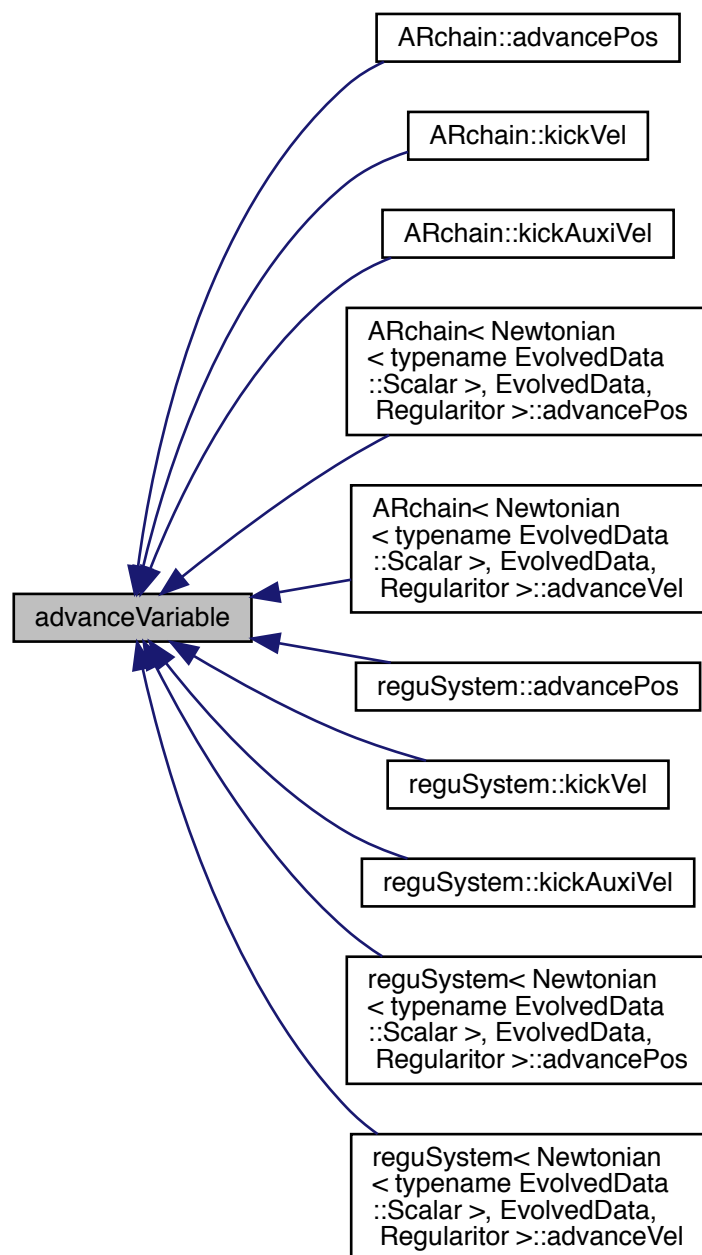
```
template<typename Scalar , size_t N>
void advanceVariable (
    std::array< vec3< Scalar >, N > & var,
    const std::array< vec3< Scalar >, N > & add,
    Scalar dt )
```

Normal Summation of two Arrays.

Parameters

<i>var</i>	Array of variable needs evolution.
<i>increase</i>	Array of increament.
<i>dt</i>	Step size of advance.

Here is the caller graph for this function:



8.10.1.3 getKineticEnergy()

```

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.

Parameters

<i>mass</i>	Array of mass.
<i>vel</i>	Array of velocity.

Returns

The kinetic energy.

Here is the caller graph for this function:



8.10.1.4 getPotentialEnergy()

```
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.

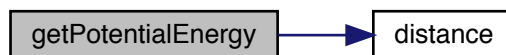
Parameters

<i>mass</i>	Array of mass.
<i>pos</i>	Array of position.

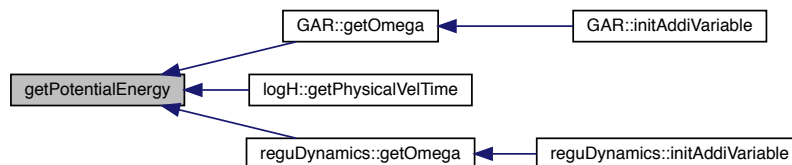
Returns

The potential energy.

Here is the call graph for this function:



Here is the caller graph for this function:



8.10.1.5 getTotalEnergy()

```

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 ) [inline]
  
```

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

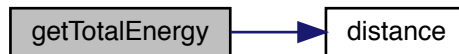
Parameters

<i>mass</i>	Array of mass.
<i>pos</i>	Array of position.
<i>vel</i>	Array of velocity.

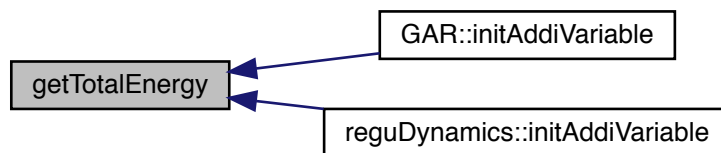
Returns

The total energy.

Here is the call graph for this function:



Here is the caller graph for this function:



8.10.1.6 KahanAdvance() [1/2]

```

template<typename Scalar , size_t N>
void KahanAdvance (
    std::array< vec3< Scalar >, N > & var,
    const std::array< vec3< Scalar >, N > & increase,
    std::array< vec3< Scalar >, N > & err,
    Scalar dt )
  
```

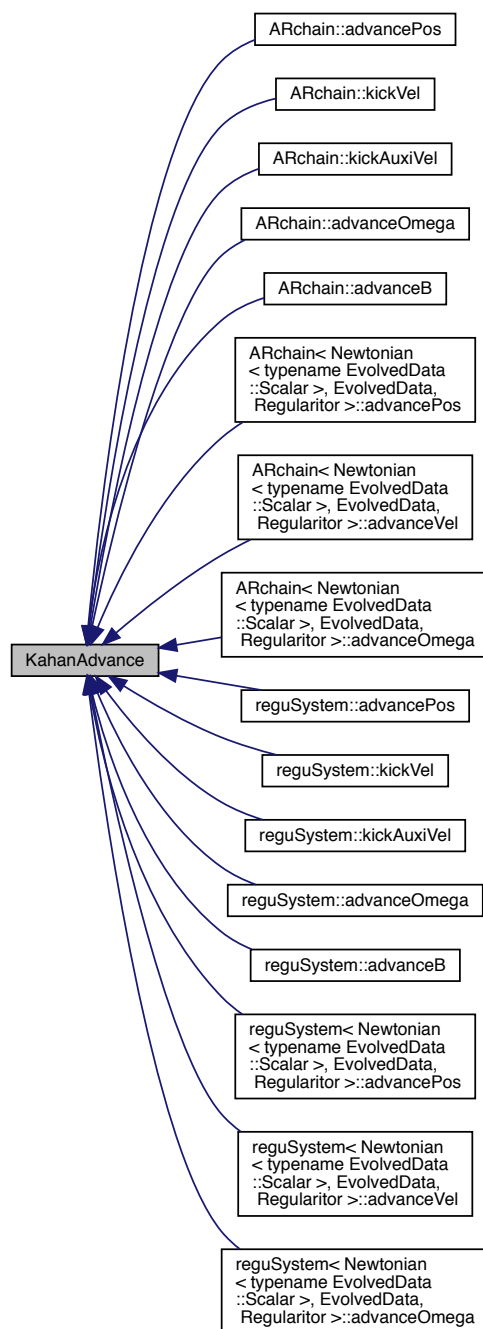
Kahan Summation for Array.

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

Parameters

<i>var</i>	Array of variable needs evolution.
<i>increase</i>	Array of increament.
<i>err</i>	Array of round off error from last addition.
<i>dt</i>	Step size of advance.

Here is the caller graph for this function:



8.10.1.7 KahanAdvance() [2/2]

```

template<typename Scalar >
void KahanAdvance (
    Scalar & var,

```

```
const Scalar increase,
Scalar & err )
```

Kahan Summation for Scalar.

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

Parameters

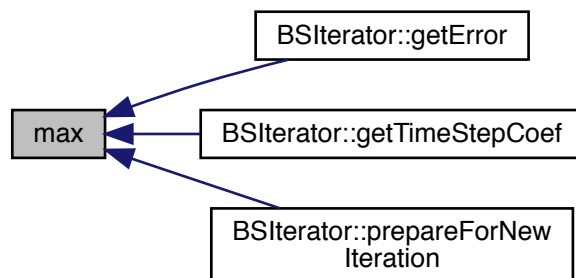
<i>var</i>	Scalar variable needs evolution.
<i>increase</i>	Scalar increament.
<i>err</i>	Scalar round off error from last addition.

8.10.1.8 max()

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

Self [max\(\)](#)

Here is the caller graph for this function:

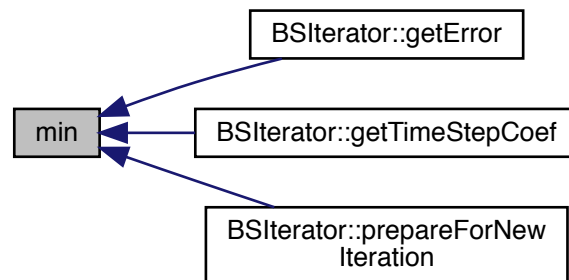


8.10.1.9 min()

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

Self [min\(\)](#)

Here is the caller graph for this function:



8.10.1.10 MoveToCentralMassCoordinate()

```

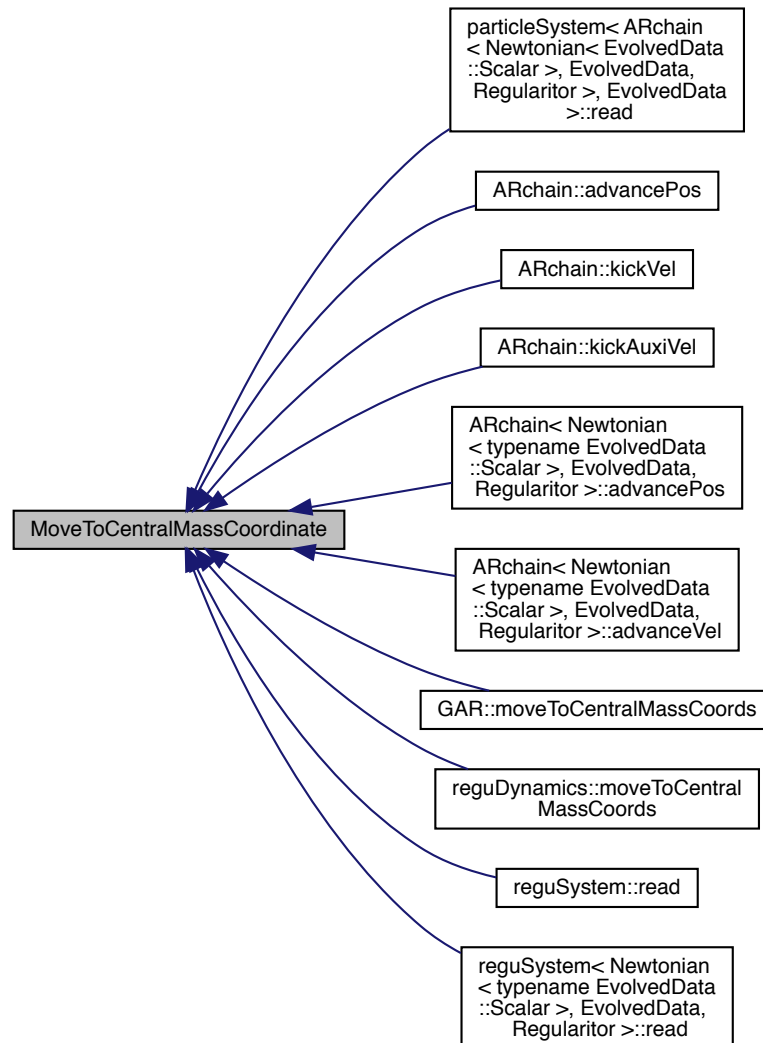
template<typename Scalar , size_t N>
void MoveToCentralMassCoordinate (
    const std::array< Scalar, N > & mass,
    std::array< vec3< Scalar >, N > & phyVar )
  
```

Move variables to central mass coordinates.

Parameters

<i>mass</i>	Array of mass.
<i>phyVar</i>	Array of variables need to be moved.

Here is the caller graph for this function:



8.10.1.11 print()

```

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

print an array. Used for debug

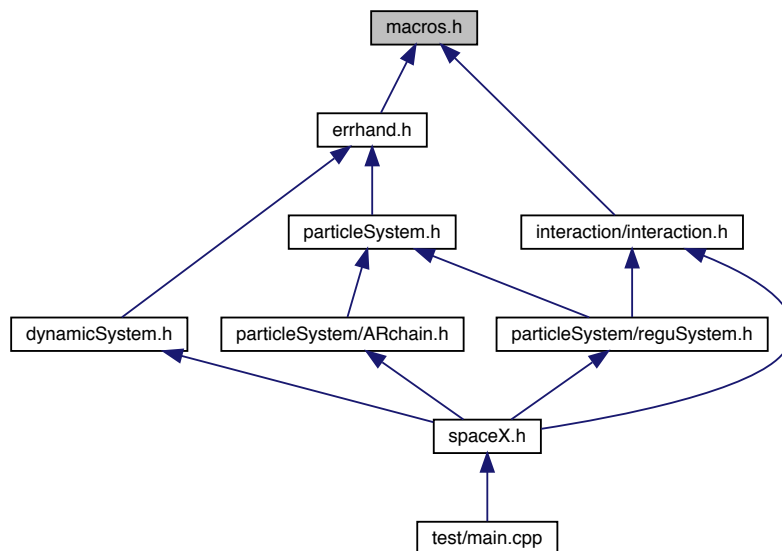
8.10.1.12 swap()

```
template<class T >
void swap (
    T & a,
    T & b )
```

Self [swap\(\)](#)

8.11 macros.h File Reference

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



Enumerations

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

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

8.11.1 Enumeration Type Documentation

8.11.1.1 EVENTTYPE

enum **EVENTTYPE**

Enumerator

TDE	
MERGE	
ESCAPE	
DISRUPTED	
UNEVENTFUL	
HVS	

8.11.1.2 INTEGRATORTYPE

enum **INTEGRATORTYPE** [strong]

Enumerator

DKDLEAPFROG	
KDKLEAPFROG	
SYM4	
PEFRL	
SYM6	
SYM8	
SYM10	

8.11.1.3 ITERTYPE

enum `ITERTYPE` [strong]

Enumerator

BSITER	
SEQITER	

8.11.1.4 PARTICTYPE

enum `PARTICTYPE`

Enumerator

NEUTRONSTAR	
STAR	
BLACKHOLE	
PSize_t	
NONE	

8.11.1.5 REGUTYPE

enum `REGUTYPE` [strong]

Enumerator

LOGH	
TTL	
NONE	

8.11.1.6 SYSTEMTYPE

enum `SYSTEMTYPE` [strong]

Enumerator

PLAIN	
CHAIN	

8.11.2 Variable Documentation

8.11.2.1 AU

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

8.11.2.2 C

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

8.11.2.3 DAY

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

8.11.2.4 G

```
constexpr double G = 1
```

8.11.2.5 KM

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

8.11.2.6 M_JUPITER

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

8.11.2.7 M_SOLAR

```
constexpr double M_SOLAR = 1
```

8.11.2.8 PC

```
constexpr double PC = 1
```

8.11.2.9 PI

```
constexpr double PI = 3.14159265358979323
```

8.11.2.10 R_SOLAR

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

8.11.2.11 V_UNIT

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

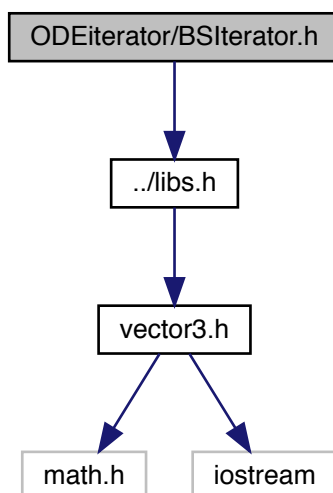
8.11.2.12 YEAR

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

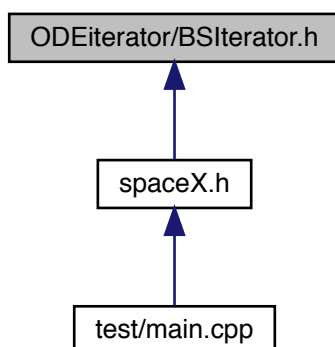
8.12 ODEiterator/BSIterator.h File Reference

```
#include "../libs.h"
```

Include dependency graph for BSIterator.h:



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

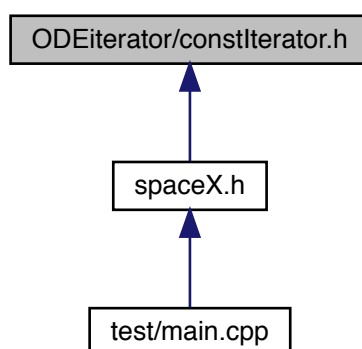


Classes

- class `BSIterator< ParticSys, Integrator >`
Bulirsch-Stoer extrapolation algorithm.

8.13 ODEiterator/constIterator.h File Reference

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



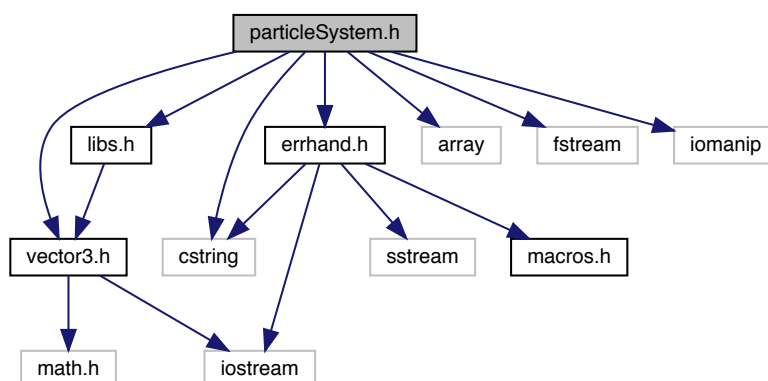
Classes

- class `constIterator< ParticSys, Integrator >`
Most common iterator.

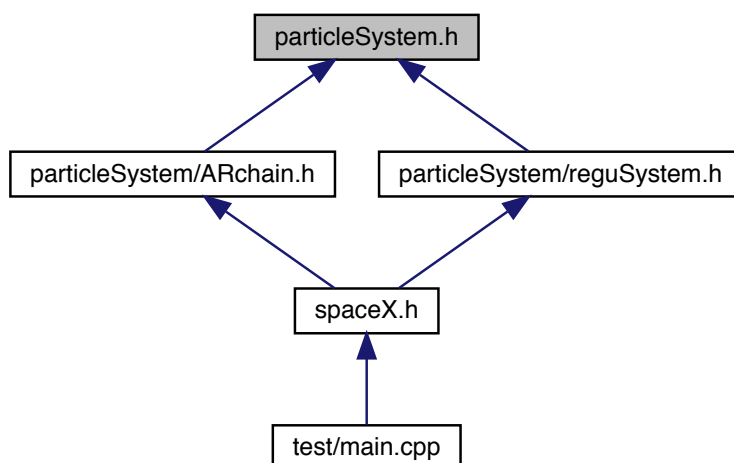
8.14 particleSystem.h File Reference

```
#include "vector3.h"
#include "libs.h"
#include "errhand.h"
#include <array>
#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< Derived, EvolvedData >](#)
Base class of particle System.

Functions

- `template<typename Derived , typename EvolvedData >`
`std::ostream & operator<< (std::ostream &os, const particleSystem< Derived, EvolvedData > &data)`
Overload operator <<.
- `template<typename Derived , typename EvolvedData >`
`std::istream & operator>> (std::istream &is, particleSystem< Derived, EvolvedData > &data)`
Overload operator >>.

8.14.1 Function Documentation

8.14.1.1 operator<<()

```
template<typename Derived , typename EvolvedData >
std::ostream& operator<< (
    std::ostream & os,
    const particleSystem< Derived, EvolvedData > & data )
```

Overload operator <<.

8.14.1.2 operator>>()

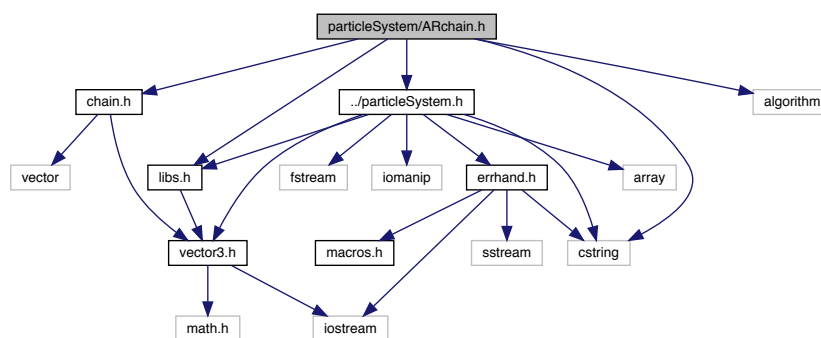
```
template<typename Derived , typename EvolvedData >
std::istream& operator>> (
    std::istream & is,
    particleSystem< Derived, EvolvedData > & data )
```

Overload operator >>.

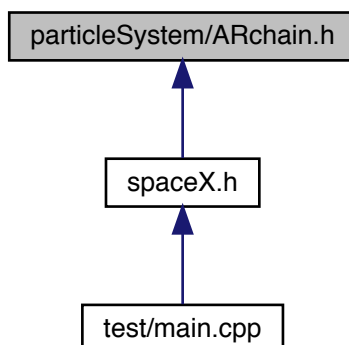
8.15 particleSystem/ARchain.h File Reference

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

Include dependency graph for ARchain.h:



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



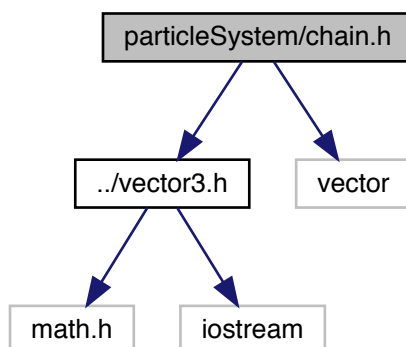
Classes

- class [ARchain](#)< [Interaction](#), [EvolvedData](#), [Regularitor](#) >
Algorithmic Regularization chain System.
- class [ARchain](#)< [Newtonian](#)< [typename EvolvedData::Scalar](#) >, [EvolvedData](#), [Regularitor](#) >
Partial specilization of template [ARchain](#).

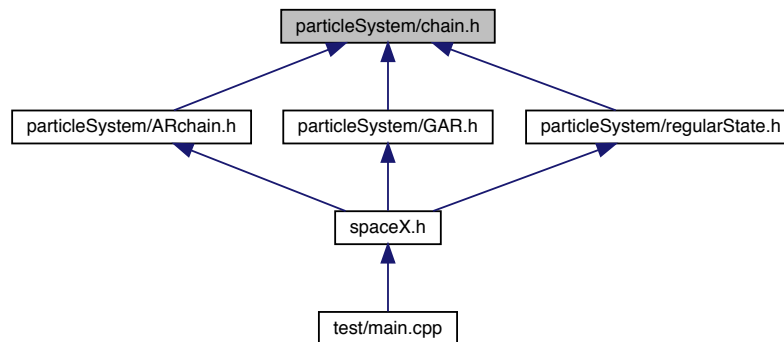
8.16 particleSystem/chain.h File Reference

```
#include "../vector3.h"
#include <vector>
```

Include dependency graph for chain.h:



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



Classes

- struct `chain::Node< Scalar >`
Struture to store the relative distance and index of two particles.

Namespaces

- `chain`

Typedefs

- `template<typename Scalar , size_t N>`
`using chain::VectorArray = std::array< vec3< Scalar >, N >`
- `template<size_t N>`
`using chain::IndexArray = std::array< size_t, N >`
- `template<typename Scalar , size_t N>`
`using chain::NodeArray = std::array< Node< Scalar >, N >`

Functions

- `template<typename Scalar , size_t N>`
`void chain::getChainIndex (const VectorArray< Scalar, N > &pos, IndexArray< N > &chainIndex)`
Calculate the mapping index from Cartesian coordinate to chain coordinate.
- `template<typename Scalar , size_t N>`
`void chain::createAdjMartix (const VectorArray< Scalar, N > &pos, NodeArray< Scalar, N *(N - 1)/2 > &AdjMatrix)`
Create the adjoint matrix for particle pairs.
- `template<typename Scalar , size_t N>`
`void chain::createChainIndex (NodeArray< Scalar, N *(N - 1)/2 > &AdjMatrix, IndexArray< N > &chainIndex)`
Create mapping index from adjoint matrix.
- `template<size_t N>`
`bool chain::IsDiff (const IndexArray< N > &Index1, const IndexArray< N > &Index2)`

Check if two mapping indexes are the same.

- template<typename Scalar , size_t N>
void [chain::updateChain](#) (VectorArray< Scalar, N > &pos, IndexArray< N > &chainIndex, IndexArray< N > &newIndex)

Update the position chain.

- template<typename Scalar , size_t N>
void [chain::synChain](#) (VectorArray< Scalar, N > &data, VectorArray< Scalar, N > &chainData, IndexArray< N > &chainIndex)

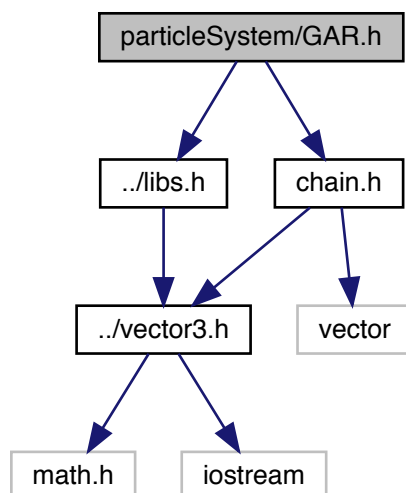
Calculate the chain data from Cartesian data and chain index mapping.

- template<typename Scalar , size_t N>
void [chain::synCartesian](#) (VectorArray< Scalar, N > &chainData, VectorArray< Scalar, N > &data, IndexArray< N > &chainIndex)

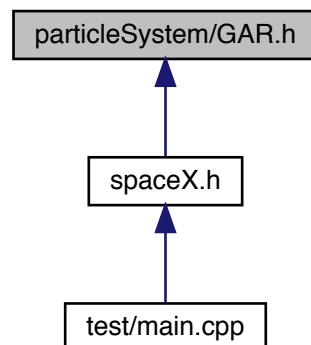
Calculate the Cartesian data from chain data and chain index mapping.

8.17 particleSystem/GAR.h File Reference

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



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



Classes

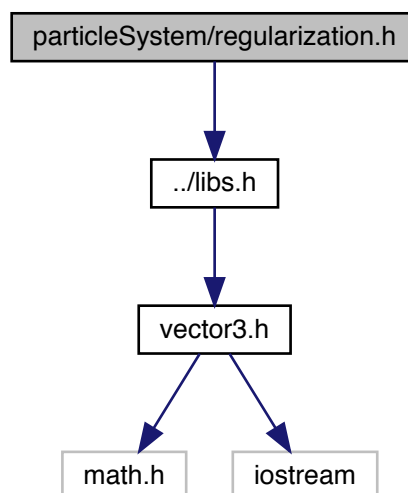
- class `GAR< DataType, N >`

Class of velocity dependent dynamical system with regularization variables.

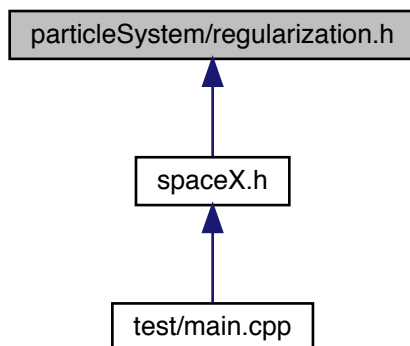
8.18 particleSystem/regularization.h File Reference

```
#include "../libs.h"
```

Include dependency graph for regularization.h:



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



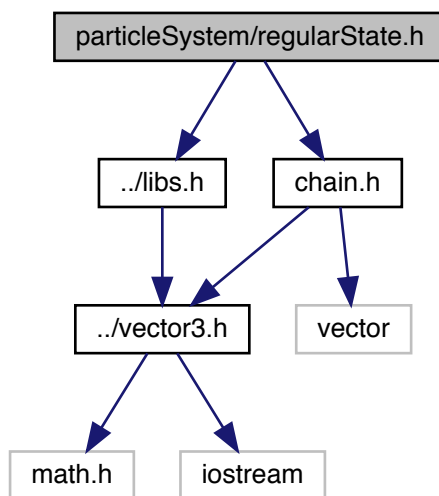
Classes

- class [logH< DynamicState >](#)
logH extention algorithmatic regularization interface
- class [TTL< DynamicState >](#)
Time Transform Leapfrog algorithmatic regularization interface.
- class [NoRegu< DynamicState >](#)
Ordinary algorithmatic regularization interface.

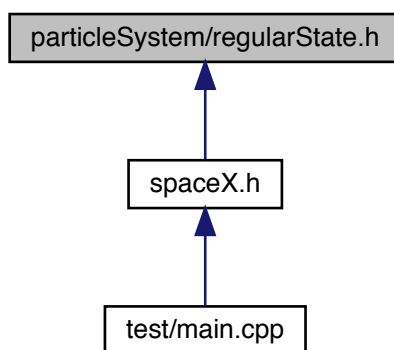
8.19 particleSystem/regularState.h File Reference

```
#include "chain.h"  
#include "../libs.h"
```

Include dependency graph for regularState.h:



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



Classes

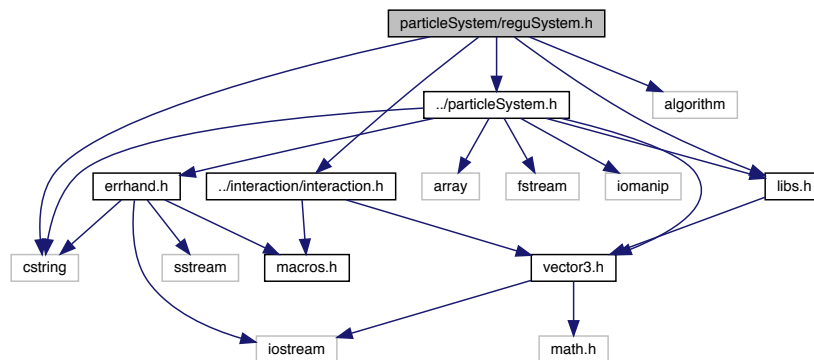
- class `reguDynamics< DataType, N >`

Class of dynamical system with regularization variables.

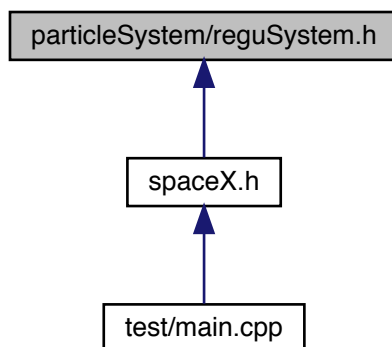
8.20 particleSystem/reguSystem.h File Reference

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

Include dependency graph for reguSystem.h:



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



Classes

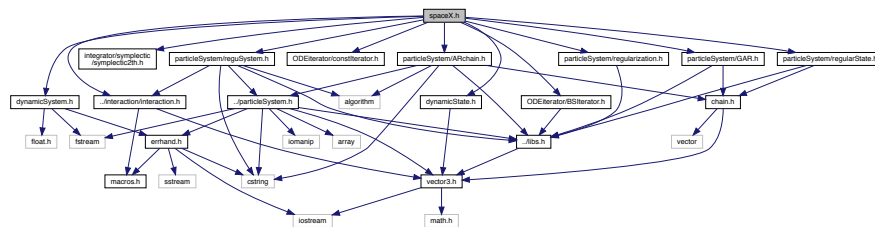
- class `reguSystem< Interaction, EvolvedData, Regularitor >`
Regularized particle System.
- class `reguSystem< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >`
Partial specialization of `reguSystem` for velocity independent system.

8.21 README.md File Reference

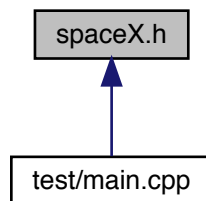
8.22 spaceX.h File Reference

```
#include "dynamicSystem.h"
#include "dynamicState.h"
#include "integrator/symplectic/symplectic2th.h"
#include "ODEiterator/BSIterator.h"
#include "ODEiterator/constIterator.h"
#include "particleSystem/reguSystem.h"
#include "particleSystem/ARchain.h"
#include "particleSystem/GAR.h"
#include "particleSystem/regularState.h"
#include "particleSystem/regularization.h"
#include "interaction/interaction.h"
```

Include dependency graph for spaceX.h:



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



Typedefs

- `template<size_t N, template< typename > class Regularitor = logH, typename Scalar = double, template< typename, size_t > class EvolvedData = reguDynamics, template< typename > class Interaction = Newtonian>`
using `NewtonianSystem = reguSystem< Interaction< Scalar >, EvolvedData< Scalar, N >, Regularitor< EvolvedData< Scalar, N >>>`
- `template<size_t N, template< typename > class Interaction, template< typename > class Regularitor = logH, typename Scalar = double, template< typename, size_t > class EvolvedData = GAR>`
using `VelDepSystem = reguSystem< Interaction< Scalar >, EvolvedData< Scalar, N >, Regularitor< EvolvedData< Scalar, N >>>`

- `template<size_t N, template< typename > class Regularitor = logH, typename Scalar = double, template< typename, size_t > class EvolvedData = reguDynamics, template< typename > class Interaction = Newtonian>`
`using AR_chain = ARchain< Interaction< Scalar >, EvolvedData< Scalar, N >, Regularitor< EvolvedData< Scalar, N > >>`
- `template<size_t N, template< typename > class Interaction, template< typename > class Regularitor = logH, typename Scalar = double, template< typename, size_t > class EvolvedData = GAR>`
`using VelARchain = ARchain< Interaction< Scalar >, EvolvedData< Scalar, N >, Regularitor< EvolvedData< Scalar, N > >>`

8.22.1 Typedef Documentation

8.22.1.1 AR_chain

```
template<size_t N, template< typename > class Regularitor = logH, typename Scalar = double,
template< typename, size_t > class EvolvedData = reguDynamics, template< typename > class
Interaction = Newtonian>
using AR_chain = ARchain<Interaction<Scalar>, EvolvedData<Scalar, N>, Regularitor<EvolvedData<Scalar, N> >>
```

8.22.1.2 NewtonianSystem

```
template<size_t N, template< typename > class Regularitor = logH, typename Scalar = double,
template< typename, size_t > class EvolvedData = reguDynamics, template< typename > class
Interaction = Newtonian>
using NewtonianSystem = reguSystem<Interaction<Scalar>, EvolvedData<Scalar, N>, Regularitor<EvolvedData<Scalar, N> >>
```

8.22.1.3 VelARchain

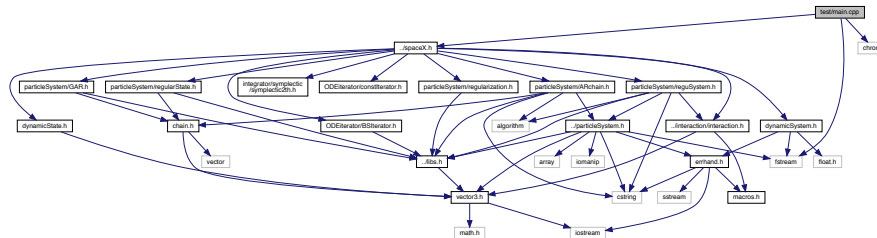
```
template<size_t N, template< typename > class Interaction, template< typename > class Regularitor
= logH, typename Scalar = double, template< typename, size_t > class EvolvedData = GAR>
using VelARchain = ARchain<Interaction<Scalar>, EvolvedData<Scalar, N>, Regularitor<EvolvedData<Scalar, N> >>
```

8.22.1.4 VelDepSystem

```
template<size_t N, template< typename > class Interaction, template< typename > class Regularitor
= logH, typename Scalar = double, template< typename, size_t > class EvolvedData = GAR>
using VelDepSystem = reguSystem<Interaction<Scalar>, EvolvedData<Scalar, N>, Regularitor<EvolvedData<Scalar, N> >>
```


8.23 test/main.cpp File Reference

```
#include "../spaceX.h"
#include <chrono>
#include <fstream>
Include dependency graph for main.cpp:
```



Typedefs

- typedef std::chrono::time_point< std::chrono::high_resolution_clock > [resolutionClock](#)

Functions

- int [main](#) (int argc, char **argv)

8.23.1 Typedef Documentation

8.23.1.1 resolutionClock

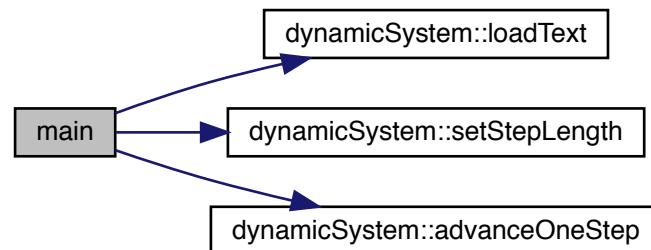
```
typedef std::chrono::time_point<std::chrono::high_resolution_clock> resolutionClock
```

8.23.2 Function Documentation

8.23.2.1 main()

```
int main (
    int argc,
    char ** argv )
```

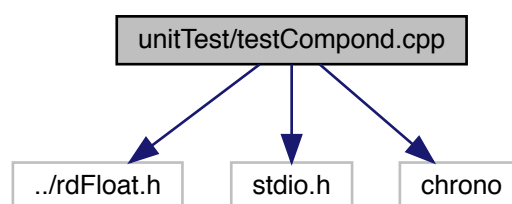
Here is the call graph for this function:



8.24 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)

8.24.1 Typedef Documentation

8.24.1.1 resolutionClock

```
typedef std::chrono::time_point<std::chrono::high_resolution_clock> resolutionClock
```

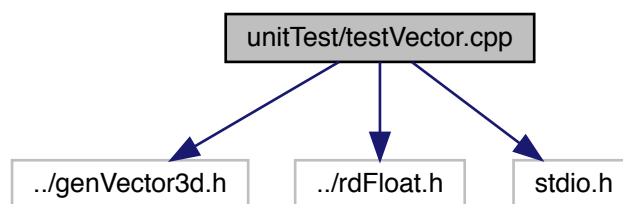
8.24.2 Function Documentation

8.24.2.1 main()

```
int main (  
    int argc,  
    char ** argv )
```

8.25 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)

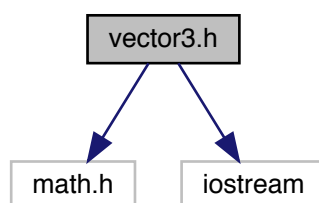
8.25.1 Function Documentation

8.25.1.1 main()

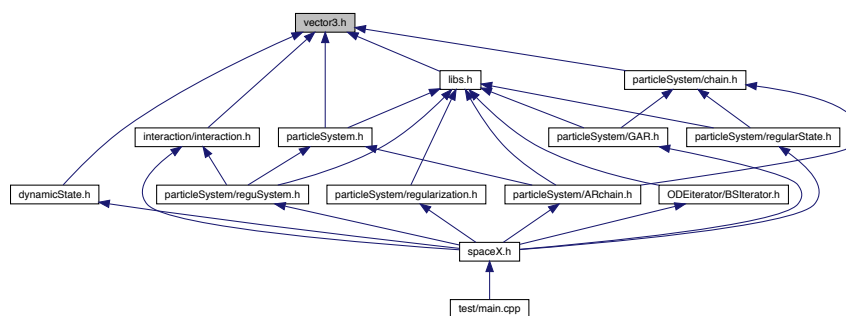
```
int main (
    int argc,
    char ** argv )
```

8.26 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.

8.26.1 Typedef Documentation

8.26.1.1 vec3c

```
typedef vec3<char> vec3c
```

8.26.1.2 vec3d

```
typedef vec3<double> vec3d
```

8.26.1.3 vec3f

```
typedef vec3<float> vec3f
```

8.26.1.4 vec3i

```
typedef vec3<int> vec3i
```

8.26.2 Function Documentation

8.26.2.1 distance()

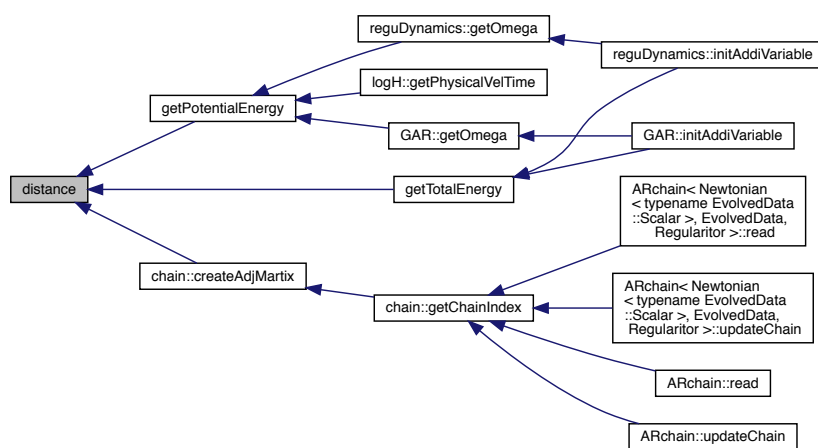
```

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

- ~dynamicSystem
 - dynamicSystem, [50](#)
- ~particleSystem
 - particleSystem, [74](#)
- ANSI_COLOR_BLUE
 - errhand.h, [124](#)
- ANSI_COLOR_CYAN
 - errhand.h, [124](#)
- ANSI_COLOR_GREEN
 - errhand.h, [124](#)
- ANSI_COLOR_MAGENTA
 - errhand.h, [124](#)
- ANSI_COLOR_RESET
 - errhand.h, [125](#)
- ANSI_COLOR_RED
 - errhand.h, [125](#)
- ANSI_COLOR_YELLOW
 - errhand.h, [125](#)
- AR_chain
 - spaceX.h, [155](#)
- ARchain
 - advanceOmega, [19](#)
 - advancePos, [19](#)
 - advanceVel, [20](#)
 - advanceB, [18](#)
 - array, [20](#)
 - chain, [25](#)
 - chainIndex, [25](#)
 - chainVelDepAcc, [25](#)
 - chainVelIndepAcc, [25](#)
 - IndexArray, [17](#)
 - kickAuxiVel, [20](#)
 - kickVel, [21](#)
 - load, [22](#)
 - operator=, [23](#)
 - PlainArray, [17](#)
 - read, [23](#)
 - regular, [25](#)
 - Scalar, [17](#)
 - ScalarArray, [17](#)
 - size, [23](#)
 - timeScale, [23](#)
 - updateAccWith, [24](#)
 - updateChain, [24](#)
 - updateVelIndepAcc, [24](#)
 - Vector, [18](#)
 - VectorArray, [18](#)
 - velDepAcc, [26](#)
 - velDepForce, [26](#)
 - velIndepAcc, [26](#)
- ARchain< Interaction, EvolvedData, Regularitor >, [15](#)
- ARchain< Newtonian< typename EvolvedData::Scalar >, EvolvedData, Regularitor >, [26](#)
 - advanceOmega, [29](#)
 - advancePos, [30](#)
 - advanceVel, [31](#)
 - array, [31](#)
 - chain, [34](#)
 - chainIndex, [34](#)
 - IndexArray, [28](#)
 - load, [31](#)
 - operator=, [32](#)
 - PlainArray, [28](#)
 - read, [32](#)
 - regular, [35](#)
 - Scalar, [28](#)
 - ScalarArray, [28](#)
 - size, [33](#)
 - timeScale, [33](#)
 - updateChain, [33](#)
 - updateVelIndepAcc, [34](#)
 - Vector, [29](#)
 - VectorArray, [29](#)
 - velIndepAcc, [35](#)
- abs
 - libs.h, [129](#)
 - vec3, [112](#)
- absoluteError
 - BSIterator, [41](#)
- acc
 - particleSystem, [76](#)
- advanceOmega
 - ARchain, [19](#)
 - ARchain< Newtonian< typename EvolvedData↵::Scalar >, EvolvedData, Regularitor >, [29](#)
 - reguSystem, [89](#)
 - reguSystem< Newtonian< typename Evolved↵Data::Scalar >, EvolvedData, Regularitor >, [97](#)
- advanceOneStep
 - dynamicSystem, [50](#)
- advancePos
 - ARchain, [19](#)
 - ARchain< Newtonian< typename EvolvedData↵::Scalar >, EvolvedData, Regularitor >, [30](#)
 - reguSystem, [90](#)
 - reguSystem< Newtonian< typename Evolved↵Data::Scalar >, EvolvedData, Regularitor >, [98](#)
- advanceVariable
 - libs.h, [130](#)
- advanceVel
 - ARchain, [20](#)
 - ARchain< Newtonian< typename EvolvedData↵::Scalar >, EvolvedData, Regularitor >, [31](#)
 - reguSystem, [90](#)
 - reguSystem< Newtonian< typename Evolved↵Data::Scalar >, EvolvedData, Regularitor >, [98](#)
- advanceB

- ARchain, 18
- reguSystem, 89
- array
 - ARchain, 20
 - ARchain< Newtonian< typename EvolvedData↔
::Scalar >, EvolvedData, Regularitor >, 31
 - dynamics, 47
 - GAR, 58
 - particleSystem, 74
 - reguDynamics, 81
- AU
 - macros.h, 141
- auxiVel
 - GAR, 63
- available
 - chain::Node, 68
- BSIterator
 - absoluteError, 41
 - BSIterator, 37
 - CC, 41
 - checkRejection, 37
 - copyDataToExtrapTab, 38
 - cost, 42
 - extrapTab, 42
 - extrapolate, 38
 - fmin, 42
 - getError, 38
 - getTimeStepCoef, 39
 - iterDepth, 42
 - iterate, 39
 - localSystem, 42
 - macroStepLength, 42
 - MaxDepth, 43
 - nSteps, 43
 - prepareForNewIteration, 40
 - relativeError, 43
 - s1, 43
 - s2, 43
 - s3, 43
 - s4, 44
 - Scalar, 37
 - scalarArray, 37
 - setAbsoluteError, 41
 - setRelativeError, 41
 - work, 44
- BSIterator< ParticSys, Integrator >, 35
- bindE
 - GAR, 63
 - reguDynamics, 85
- C
 - macros.h, 142
- CC
 - BSIterator, 41
- chain, 5
 - ARchain, 25
 - ARchain< Newtonian< typename EvolvedData↔
::Scalar >, EvolvedData, Regularitor >, 34
 - createAdjMartix, 7
 - createChainIndex, 7
 - getChainIndex, 8
 - IndexArray, 6
 - IsDiff, 9
 - NodeArray, 6
 - synCartesian, 10
 - synChain, 11
 - updateChain, 12
 - VectorArray, 6
- chain::Node
 - available, 68
 - i, 68
 - j, 68
 - Rij, 69
- chain::Node< Scalar >, 68
- chainIndex
 - ARchain, 25
 - ARchain< Newtonian< typename EvolvedData↔
::Scalar >, EvolvedData, Regularitor >, 34
- chainVelDepAcc
 - ARchain, 25
- chainVelIndepAcc
 - ARchain, 25
- checkRejection
 - BSIterator, 37
- constIterator
 - iterate, 45
 - Scalar, 44
- constIterator< ParticSys, Integrator >, 44
- copyDataToExtrapTab
 - BSIterator, 38
- cost
 - BSIterator, 42
- createAdjMartix
 - chain, 7
- createChainIndex
 - chain, 7
- DAY
 - macros.h, 142
- distance
 - vector3.h, 160
- dynState
 - particleSystem, 76
- dynamicState.h, 120
- dynamicSystem
 - ~dynamicSystem, 50
 - advanceOneStep, 50
 - getInitStepLength, 50
 - integrator, 52
 - iterator, 52
 - loadText, 51
 - particles, 52
 - setStepLength, 51
 - stepLength, 52
- dynamicSystem< ParticSys, Integrator, ODEiterator >, 49
- dynamicSystem.h, 121

- spaceX, 122
- dynamics
 - array, 47
 - initAddiVariable, 47
 - pos, 48
 - Scalar, 46
 - ScalarArray, 46
 - setZero, 47
 - size, 48
 - time, 48
 - Vector, 46
 - VectorArray, 46
 - vel, 49
 - volume, 48
- dynamics< DataType, N >, 45
- EVENTTYPE
 - macros.h, 140
- EraseLine
 - NOTICE, 13
- err_msg
 - errhand, 56
- errhand, 53
 - err_msg, 56
 - errhand, 53
 - file, 56
 - get_file, 53
 - get_line, 54
 - get_msg, 54
 - invoke_telegram_bot, 54
 - line, 56
 - print_to_stdout, 55
 - to_string_loc, 56
- errhand.h, 123
 - ANSI_COLOR_BLUE, 124
 - ANSI_COLOR_CYAN, 124
 - ANSI_COLOR_GREEN, 124
 - ANSI_COLOR_MAGENTA, 124
 - ANSI_COLOR_RESET, 125
 - ANSI_COLOR_RED, 125
 - ANSI_COLOR_YELLOW, 125
 - NEWLINE, 125
- extrapTab
 - BSIterator, 42
- extrapolate
 - BSIterator, 38
- file
 - errhand, 56
- fmin
 - BSIterator, 42
- G
 - macros.h, 142
- GAR< DataType, N >, 57
- GAR
 - array, 58
 - auxiVel, 63
 - bindE, 63
 - getOmega, 59
 - IndexArray, 58
 - initAddiVariable, 59
 - moveToCentralMassCoords, 60
 - omega, 63
 - pos, 63
 - Scalar, 58
 - ScalarArray, 58
 - setZero, 60
 - size, 61
 - time, 63
 - toCartesian, 61
 - toChain, 62
 - Vector, 58
 - VectorArray, 58
 - vel, 64
 - volume, 62
- get_file
 - errhand, 53
- get_line
 - errhand, 54
- get_msg
 - errhand, 54
- getChainIndex
 - chain, 8
- getError
 - BSIterator, 38
- getInitStepLength
 - dynamicSystem, 50
- getKineticEnergy
 - libs.h, 131
- getOmega
 - GAR, 59
 - reguDynamics, 81
- getPhysicalPosTime
 - logH, 65
 - NoRegu, 70
 - TTL, 109
- getPhysicalVelTime
 - logH, 65
 - NoRegu, 70
 - TTL, 109
- getPotentialEnergy
 - libs.h, 132
- getTimeStepCoef
 - BSIterator, 39
- getTotalEnergy
 - libs.h, 133
- i
 - chain::Node, 68
- INTEGRATORTYPE
 - macros.h, 140
- INV_C2
 - interaction.h, 127
- INV_C3
 - interaction.h, 128
- INV_C4
 - interaction.h, 128

- INV_C5
 - interaction.h, 128
- INV_C
 - interaction.h, 127
- ITERTYPE
 - macros.h, 140
- IndexArray
 - ARchain, 17
 - ARchain< Newtonian< typename EvolvedData↔
::Scalar >, EvolvedData, Regularitor >, 28
 - chain, 6
 - GAR, 58
 - reguDynamics, 80
- initAddiVariable
 - dynamics, 47
 - GAR, 59
 - reguDynamics, 82
- IntArray
 - particleSystem, 73
- integrate
 - symplectic10th, 102
 - symplectic2th, 104
 - symplectic4th, 105
 - symplectic6th, 106
 - symplectic8th, 107
- integrator
 - dynamicSystem, 52
- integrator/symplectic/symplectic10th.h, 125
- integrator/symplectic/symplectic2th.h, 125
- integrator/symplectic/symplectic4th.h, 126
- integrator/symplectic/symplectic6th.h, 126
- integrator/symplectic/symplectic8th.h, 126
- interaction.h
 - INV_C2, 127
 - INV_C3, 128
 - INV_C4, 128
 - INV_C5, 128
 - INV_C, 127
- interaction/interaction.h, 126
- invoke_telegram_bot
 - errhand, 54
- IsDiff
 - chain, 9
- iterDepth
 - BSIterator, 42
- iterate
 - BSIterator, 39
 - constIterator, 45
- iterator
 - dynamicSystem, 52
- j
 - chain::Node, 68
- KahanAdvance
 - libs.h, 134, 135
- kickAuxiVel
 - ARchain, 20
 - reguSystem, 91
- kickVel
 - ARchain, 21
 - reguSystem, 91
- KM
 - macros.h, 142
- libs.h, 128
 - abs, 129
 - advanceVariable, 130
 - getKineticEnergy, 131
 - getPotentialEnergy, 132
 - getTotalEnergy, 133
 - KahanAdvance, 134, 135
 - max, 136
 - min, 136
 - MoveToCentralMassCoordinate, 137
 - print, 138
 - swap, 138
- Line
 - NOTICE, 13
- line
 - errhand, 56
- load
 - ARchain, 22
 - ARchain< Newtonian< typename EvolvedData↔
::Scalar >, EvolvedData, Regularitor >, 31
 - particleSystem, 74
 - reguSystem, 92
 - reguSystem< Newtonian< typename Evolved↔
Data::Scalar >, EvolvedData, Regularitor >, 99
- loadText
 - dynamicSystem, 51
- localSystem
 - BSIterator, 42
- logH< DynamicState >, 64
- logH
 - getPhysicalPosTime, 65
 - getPhysicalVelTime, 65
 - Scalar, 65
 - size, 67
- M_JUPITER
 - macros.h, 142
- M_SOLAR
 - macros.h, 142
- macroStepLength
 - BSIterator, 42
- macros.h, 139
 - AU, 141
 - C, 142
 - DAY, 142
 - EVENTTYPE, 140
 - G, 142
 - INTEGRATORTYPE, 140
 - ITERTYPE, 140
 - KM, 142
 - M_JUPITER, 142
 - M_SOLAR, 142

- PARTICTYPE, 141
- PC, 142
- PI, 142
- R_SOLAR, 142
- REGUTYPE, 141
- SYSTEMTYPE, 141
- V_UNIT, 143
- YEAR, 143
- main
 - main.cpp, 156
 - testCompond.cpp, 158
 - testVector.cpp, 158
- main.cpp
 - main, 156
 - resolutionClock, 156
- mass
 - particleSystem, 76
- max
 - libs.h, 136
- MaxDepth
 - BSIterator, 43
- Message
 - NOTICE, 14
- min
 - libs.h, 136
- MoveToCentralMassCoordinate
 - libs.h, 137
- moveToCentralMassCoords
 - GAR, 60
 - reguDynamics, 82
- NEWLINE
 - errhand.h, 125
- NOTICE, 13
 - EraseLine, 13
 - Line, 13
 - Message, 14
 - RunInfo, 14
 - SubLine, 14
 - SubTitle, 14
 - Telegram, 14
 - Title, 14
 - WIDTH, 14
- nSteps
 - BSIterator, 43
- Newtonian
 - operator(), 68
- Newtonian< Scalar >, 67
- NewtonianSystem
 - spaceX.h, 155
- NoRegu
 - getPhysicalPosTime, 70
 - getPhysicalVelTime, 70
 - Scalar, 70
 - size, 70
- NoRegu< DynamicState >, 69
- NodeArray
 - chain, 6
- norm
 - vec3, 113
- normSquare
 - vec3, 113
- ODEiterator/BSIterator.h, 143
- ODEiterator/constIterator.h, 144
- omega
 - GAR, 63
 - reguDynamics, 85
- operator<<
 - particleSystem.h, 146
 - vec3, 119
- operator>>
 - particleSystem.h, 146
 - vec3, 119
- operator*
 - vec3, 113, 118
- operator*=
 - vec3, 114
- operator^
 - vec3, 117
- operator()
 - Newtonian, 68
 - PN1th, 78
- operator+
 - vec3, 114, 119
- operator+=
 - vec3, 114, 115
- operator-
 - vec3, 115, 119
- operator-=
 - vec3, 116
- operator/
 - vec3, 116
- operator/=
 - vec3, 117
- operator=
 - ARchain, 23
 - ARchain< Newtonian< typename EvolvedData↵
::Scalar >, EvolvedData, Regularitor >, 32
 - particleSystem, 75
 - reguSystem, 92
 - vec3, 117
- order
 - symplectic10th, 103
 - symplectic2th, 104
 - symplectic4th, 105
 - symplectic6th, 106
 - symplectic8th, 108
- PARTICTYPE
 - macros.h, 141
- PN1th
 - operator(), 78
 - Vector, 78
- PN1th< Scalar >, 77
- particleSystem
 - ~particleSystem, 74
 - acc, 76

- array, [74](#)
- dynState, [76](#)
- IntArray, [73](#)
- load, [74](#)
- mass, [76](#)
- operator=, [75](#)
- particleSystem, [73](#)
- PlainArray, [73](#)
- pos, [76](#)
- radius, [77](#)
- read, [75](#)
- Scalar, [73](#)
- ScalarArray, [73](#)
- size, [75](#)
- time, [77](#)
- timeScale, [75](#)
- type, [77](#)
- Vector, [73](#)
- VectorArray, [73](#)
- vel, [77](#)
- volume, [75](#)
- write, [76](#)
- particleSystem< Derived, EvolvedData >, [71](#)
- particleSystem.h, [145](#)
 - operator<<, [146](#)
 - operator>>, [146](#)
- particleSystem/ARchain.h, [146](#)
- particleSystem/GAR.h, [149](#)
- particleSystem/chain.h, [147](#)
- particleSystem/reguSystem.h, [153](#)
- particleSystem/regularState.h, [151](#)
- particleSystem/regularization.h, [150](#)
- particles
 - dynamicSystem, [52](#)
- PC
 - macros.h, [142](#)
- PI
 - macros.h, [142](#)
- PlainArray
 - ARchain, [17](#)
 - ARchain< Newtonian< typename EvolvedData↔
::Scalar >, EvolvedData, Regularitor >, [28](#)
 - particleSystem, [73](#)
 - reguSystem, [88](#)
 - reguSystem< Newtonian< typename Evolved↔
Data::Scalar >, EvolvedData, Regularitor >, [97](#)
- pos
 - dynamics, [48](#)
 - GAR, [63](#)
 - particleSystem, [76](#)
 - reguDynamics, [86](#)
- prepareForNewIteration
 - BSIterator, [40](#)
- print
 - libs.h, [138](#)
- print_to_stdout
 - errhand, [55](#)
- R_SOLAR
 - macros.h, [142](#)
- README.md, [154](#)
- REGUTYPE
 - macros.h, [141](#)
- radius
 - particleSystem, [77](#)
- reNorm
 - vec3, [118](#)
- read
 - ARchain, [23](#)
 - ARchain< Newtonian< typename EvolvedData↔
::Scalar >, EvolvedData, Regularitor >, [32](#)
 - particleSystem, [75](#)
 - reguSystem, [92](#)
 - reguSystem< Newtonian< typename Evolved↔
Data::Scalar >, EvolvedData, Regularitor >, [100](#)
- reguDynamics
 - array, [81](#)
 - bindE, [85](#)
 - getOmega, [81](#)
 - IndexArray, [80](#)
 - initAddiVariable, [82](#)
 - moveToCentralMassCoords, [82](#)
 - omega, [85](#)
 - pos, [86](#)
 - Scalar, [80](#)
 - ScalarArray, [80](#)
 - setZero, [83](#)
 - size, [83](#)
 - time, [86](#)
 - toCartesian, [83](#)
 - toChain, [84](#)
 - Vector, [81](#)
 - VectorArray, [81](#)
 - vel, [86](#)
 - volume, [85](#)
- reguDynamics< DataType, N >, [79](#)
- reguSystem
 - advanceOmega, [89](#)
 - advancePos, [90](#)
 - advanceVel, [90](#)
 - advanceB, [89](#)
 - kickAuxiVel, [91](#)
 - kickVel, [91](#)
 - load, [92](#)
 - operator=, [92](#)
 - PlainArray, [88](#)
 - read, [92](#)
 - regular, [94](#)
 - Scalar, [88](#)
 - ScalarArray, [88](#)
 - size, [93](#)
 - timeScale, [93](#)
 - updateAccWith, [93](#)
 - updateVelIndepAcc, [94](#)
 - Vector, [88](#)

- VectorArray, [89](#)
- velDepAcc, [94](#)
- velDepForce, [95](#)
- velIndepAcc, [95](#)
- volume, [94](#)
- reguSystem< Interaction, EvolvedData, Regularitor >, [86](#)
- reguSystem< Newtonian< typename EvolvedData::↵
Scalar >, EvolvedData, Regularitor >, [95](#)
 - advanceOmega, [97](#)
 - advancePos, [98](#)
 - advanceVel, [98](#)
 - load, [99](#)
 - PlainArray, [97](#)
 - read, [100](#)
 - regular, [101](#)
 - Scalar, [97](#)
 - ScalarArray, [97](#)
 - size, [100](#)
 - timeScale, [100](#)
 - updateVelIndepAcc, [101](#)
 - Vector, [97](#)
 - VectorArray, [97](#)
 - volume, [101](#)
- regular
 - ARchain, [25](#)
 - ARchain< Newtonian< typename EvolvedData↵
::Scalar >, EvolvedData, Regularitor >, [35](#)
 - reguSystem, [94](#)
 - reguSystem< Newtonian< typename Evolved↵
Data::Scalar >, EvolvedData, Regularitor >, [101](#)
- relativeError
 - BSIterator, [43](#)
- resolutionClock
 - main.cpp, [156](#)
 - testCompound.cpp, [158](#)
- Rij
 - chain::Node, [69](#)
- RunInfo
 - NOTICE, [14](#)
- s1
 - BSIterator, [43](#)
- s2
 - BSIterator, [43](#)
- s3
 - BSIterator, [43](#)
- s4
 - BSIterator, [44](#)
- SYSTEMTYPE
 - macros.h, [141](#)
- Scalar
 - ARchain, [17](#)
 - ARchain< Newtonian< typename EvolvedData↵
::Scalar >, EvolvedData, Regularitor >, [28](#)
 - BSIterator, [37](#)
 - constIterator, [44](#)
 - dynamics, [46](#)
 - GAR, [58](#)
 - logH, [65](#)
 - NoRegu, [70](#)
 - particleSystem, [73](#)
 - reguDynamics, [80](#)
 - reguSystem, [88](#)
 - reguSystem< Newtonian< typename Evolved↵
Data::Scalar >, EvolvedData, Regularitor >, [97](#)
 - symplectic2th, [104](#)
 - TTL, [109](#)
- ScalarArray
 - ARchain, [17](#)
 - ARchain< Newtonian< typename EvolvedData↵
::Scalar >, EvolvedData, Regularitor >, [28](#)
 - dynamics, [46](#)
 - GAR, [58](#)
 - particleSystem, [73](#)
 - reguDynamics, [80](#)
 - reguSystem, [88](#)
 - reguSystem< Newtonian< typename Evolved↵
Data::Scalar >, EvolvedData, Regularitor >, [97](#)
- scalarArray
 - BSIterator, [37](#)
- setAbsoluteError
 - BSIterator, [41](#)
- setRelativeError
 - BSIterator, [41](#)
- setStepLength
 - dynamicSystem, [51](#)
- setZero
 - dynamics, [47](#)
 - GAR, [60](#)
 - reguDynamics, [83](#)
 - vec3, [118](#)
- size
 - ARchain, [23](#)
 - ARchain< Newtonian< typename EvolvedData↵
::Scalar >, EvolvedData, Regularitor >, [33](#)
 - dynamics, [48](#)
 - GAR, [61](#)
 - logH, [67](#)
 - NoRegu, [70](#)
 - particleSystem, [75](#)
 - reguDynamics, [83](#)
 - reguSystem, [93](#)
 - reguSystem< Newtonian< typename Evolved↵
Data::Scalar >, EvolvedData, Regularitor >, [100](#)
 - TTL, [110](#)
- spaceX.h, [154](#)
 - AR_chain, [155](#)
 - NewtonianSystem, [155](#)
 - VelARchain, [155](#)
 - VelDepSystem, [155](#)
- spaceX
 - dynamicSystem.h, [122](#)

- stepLength
 - dynamicSystem, 52
- SubLine
 - NOTICE, 14
- SubTitle
 - NOTICE, 14
- swap
 - libs.h, 138
- symplectic10th
 - integrate, 102
 - order, 103
- symplectic10th< ParticSys >, 102
- symplectic2th
 - integrate, 104
 - order, 104
 - Scalar, 104
- symplectic2th< ParticSys >, 103
- symplectic4th
 - integrate, 105
 - order, 105
- symplectic4th< ParticSys >, 104
- symplectic6th
 - integrate, 106
 - order, 106
- symplectic6th< ParticSys >, 106
- symplectic8th
 - integrate, 107
 - order, 108
- symplectic8th< ParticSys >, 107
- synCartesian
 - chain, 10
- synChain
 - chain, 11
- TTL< DynamicState >, 108
- TTL
 - getPhysicalPosTime, 109
 - getPhysicalVelTime, 109
 - Scalar, 109
 - size, 110
- Telegram
 - NOTICE, 14
- test/main.cpp, 156
- testCompond.cpp
 - main, 158
 - resolutionClock, 158
- testVector.cpp
 - main, 158
- time
 - dynamics, 48
 - GAR, 63
 - particleSystem, 77
 - reguDynamics, 86
- timeScale
 - ARchain, 23
 - ARchain< Newtonian< typename EvolvedData←
::Scalar >, EvolvedData, Regularitor >, 33
 - particleSystem, 75
 - reguSystem, 93
 - reguSystem< Newtonian< typename EvolvedData←
Data::Scalar >, EvolvedData, Regularitor >, 100
- Title
 - NOTICE, 14
- to_string_loc
 - errhand, 56
- toCartesian
 - GAR, 61
 - reguDynamics, 83
- toChain
 - GAR, 62
 - reguDynamics, 84
- type
 - particleSystem, 77
- unitTest/testCompond.cpp, 157
- unitTest/testVector.cpp, 158
- updateAccWith
 - ARchain, 24
 - reguSystem, 93
- updateChain
 - ARchain, 24
 - ARchain< Newtonian< typename EvolvedData←
::Scalar >, EvolvedData, Regularitor >, 33
 - chain, 12
- updateVelIndepAcc
 - ARchain, 24
 - ARchain< Newtonian< typename EvolvedData←
::Scalar >, EvolvedData, Regularitor >, 34
 - reguSystem, 94
 - reguSystem< Newtonian< typename EvolvedData←
Data::Scalar >, EvolvedData, Regularitor >, 101
- V_UNIT
 - macros.h, 143
- vec3
 - abs, 112
 - norm, 113
 - normSquare, 113
 - operator<<, 119
 - operator>>, 119
 - operator*, 113, 118
 - operator*=, 114
 - operator^, 117
 - operator+, 114, 119
 - operator+=, 114, 115
 - operator-, 115, 119
 - operator-=, 116
 - operator/, 116
 - operator/=: 117
 - operator=, 117
 - reNorm, 118
 - setZero, 118
 - vec3, 111, 112
 - x, 119
 - y, 120
 - z, 120

- vec3< T >, 110
- vec3c
 - vector3.h, 160
- vec3d
 - vector3.h, 160
- vec3f
 - vector3.h, 160
- vec3i
 - vector3.h, 160
- Vector
 - ARchain, 18
 - ARchain< Newtonian< typename EvolvedData↔
::Scalar >, EvolvedData, Regularitor >, 29
 - dynamics, 46
 - GAR, 58
 - PN1th, 78
 - particleSystem, 73
 - reguDynamics, 81
 - reguSystem, 88
 - reguSystem< Newtonian< typename Evolved↔
Data::Scalar >, EvolvedData, Regularitor >, 97
- vector3.h, 159
 - distance, 160
 - vec3c, 160
 - vec3d, 160
 - vec3f, 160
 - vec3i, 160
- VectorArray
 - ARchain, 18
 - ARchain< Newtonian< typename EvolvedData↔
::Scalar >, EvolvedData, Regularitor >, 29
 - chain, 6
 - dynamics, 46
 - GAR, 58
 - particleSystem, 73
 - reguDynamics, 81
 - reguSystem, 89
 - reguSystem< Newtonian< typename Evolved↔
Data::Scalar >, EvolvedData, Regularitor >, 97
- vel
 - dynamics, 49
 - GAR, 64
 - particleSystem, 77
 - reguDynamics, 86
- VelARchain
 - spaceX.h, 155
- velDepAcc
 - ARchain, 26
 - reguSystem, 94
- velDepForce
 - ARchain, 26
 - reguSystem, 95
- VelDepSystem
 - spaceX.h, 155
- velIndepAcc
 - ARchain, 26
- ARchain< Newtonian< typename EvolvedData↔
::Scalar >, EvolvedData, Regularitor >, 35
- reguSystem, 95
- volume
 - dynamics, 48
 - GAR, 62
 - particleSystem, 75
 - reguDynamics, 85
 - reguSystem, 94
 - reguSystem< Newtonian< typename Evolved↔
Data::Scalar >, EvolvedData, Regularitor >, 101
- WIDTH
 - NOTICE, 14
- work
 - BSIterator, 44
- write
 - particleSystem, 76
- x
 - vec3, 119
- y
 - vec3, 120
- YEAR
 - macros.h, 143
- z
 - vec3, 120