

spacehub

Generated by Doxygen 1.8.14



# Contents

1	<a href="#">Template-SpaceX</a>	2
2	<a href="#">Namespace Index</a>	2
2.1	<a href="#">Namespace List</a>	2
3	<a href="#">Hierarchical Index</a>	2
3.1	<a href="#">Class Hierarchy</a>	2
4	<a href="#">Class Index</a>	3
4.1	<a href="#">Class List</a>	3
5	<a href="#">File Index</a>	4
5.1	<a href="#">File List</a>	4
6	<a href="#">Namespace Documentation</a>	5
6.1	<a href="#">chain Namespace Reference</a>	5
6.1.1	<a href="#">Typedef Documentation</a>	6
6.1.2	<a href="#">Function Documentation</a>	7
6.2	<a href="#">NOTICE Namespace Reference</a>	13
6.2.1	<a href="#">Function Documentation</a>	13
6.2.2	<a href="#">Variable Documentation</a>	14
7	<a href="#">Class Documentation</a>	15
7.1	<a href="#">ARchain&lt; Interaction, EvolvedData, Regularitor &gt; Class Template Reference</a>	15
7.1.1	<a href="#">Detailed Description</a>	17
7.1.2	<a href="#">Member Typedef Documentation</a>	17
7.1.3	<a href="#">Member Function Documentation</a>	18
7.1.4	<a href="#">Member Data Documentation</a>	25
7.2	<a href="#">ARchain&lt; Newtonian&lt; typename EvolvedData::Scalar &gt;, EvolvedData, Regularitor &gt; Class Template Reference</a>	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



## **Chapter 1**

# **Template-SpaceX**



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">NOTICE</a>	.....	<a href="#">13</a>
<a href="#">SpaceH</a>	.....	<a href="#">??</a>
<a href="#">SpaceH::chain</a>	.....	<a href="#">??</a>



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

array	
SpaceH::ArrayWrapper< T, S > . . . . .	??
BSIterator< ParticSys, Integrator > . . . . .	35
constIterator< ParticSys, Integrator > . . . . .	44
dicholterator< ParticSys, Integrator > . . . . .	??
dtype< T > . . . . .	??
dynamicSystem< ParticSys, ODEiterator > . . . . .	49
SpaceH::EmptyForce< Dtype, ArraySize > . . . . .	??
errhand . . . . .	53
SpaceH::Force< PairForce, Dtype, ArraySize > . . . . .	??
SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize > . . . . .	??
SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize > . . . . .	??
SpaceH::VelDepForce< PairForce, Dtype, ArraySize > . . . . .	??
SpaceH::VelIndepForce< PairForce, Dtype, ArraySize > . . . . .	??
SpaceH::get_value_type< T > . . . . .	??
SpaceH::Interaction< VelIndep, VelDep, ExtVelIndep, ExtVelDep > . . . . .	??
SpaceH::kahan< T > . . . . .	??
logH< Particles > . . . . .	64
SpaceH::NewtonForce< Dtype, ArraySize > . . . . .	??
SpaceH::chain::Node< Scalar > . . . . .	??
NoRegu< Particles > . . . . .	69
particles< Dtype, ArraySize > . . . . .	??
ReguParticles< Dtype, ArraySize > . . . . .	??
ChainParticles< Dtype, ArraySize > . . . . .	??
particleSystem< Particles, Interaction > . . . . .	71
ReguSystem< Particles, Interaction, Regularitor > . . . . .	??
ARchain< Particles, Interaction, Regularitor > . . . . .	15
SpaceH::PostNewtonian< Scalar > . . . . .	??
ProgressBar . . . . .	??
SpaceH::ProtoType< Dtype, Size > . . . . .	??
symplectic10th< ParticSys > . . . . .	102
symplectic2th< ParticSys > . . . . .	103
symplectic4th< ParticSys > . . . . .	104
symplectic6th< ParticSys > . . . . .	106

<code>symplectic8th&lt; ParticSys &gt;</code>	<a href="#">107</a>
<code>TTL&lt; Particles &gt;</code>	<a href="#">108</a>
<code>vec3&lt; T &gt;</code>	<a href="#">110</a>
<code>vector</code>	
<code>SpaceH::ArrayWrapper&lt; T, DYNAMICAL &gt;</code>	<a href="#">??</a>

## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">ARchain&lt; Particles, Interaction, Regularitor &gt;</a>	
Algorithmatic Regularization chain System . . . . .	15
<a href="#">SpaceH::ArrayWrapper&lt; T, S &gt;</a> . . . . .	??
<a href="#">SpaceH::ArrayWrapper&lt; T, DYNAMICAL &gt;</a> . . . . .	??
<a href="#">BSIterator&lt; ParticSys, Integrator &gt;</a>	
Bulirsch-Stoer extrapolation algorithm . . . . .	35
<a href="#">ChainParticles&lt; Dtype, ArraySize &gt;</a>	
Class of dynamical variable . . . . .	??
<a href="#">constIterator&lt; ParticSys, Integrator &gt;</a>	
Most common iterator . . . . .	44
<a href="#">dicholterator&lt; ParticSys, Integrator &gt;</a>	
Dichotomy iterator . . . . .	??
<a href="#">dtype&lt; T &gt;</a> . . . . .	??
<a href="#">dynamicSystem&lt; ParticSys, ODEiterator &gt;</a>	
A wrapper to make particle system, integrator and ODE iterator work together . . . . .	49
<a href="#">SpaceH::EmptyForce&lt; Dtype, ArraySize &gt;</a> . . . . .	??
<a href="#">errhand</a> . . . . .	53
<a href="#">SpaceH::ExtVelDepForce&lt; PairForce, Dtype, ArraySize &gt;</a> . . . . .	??
<a href="#">SpaceH::ExtVelIndepForce&lt; PairForce, Dtype, ArraySize &gt;</a> . . . . .	??
<a href="#">SpaceH::Force&lt; PairForce, Dtype, ArraySize &gt;</a> . . . . .	??
<a href="#">SpaceH::get_value_type&lt; T &gt;</a> . . . . .	??
<a href="#">SpaceH::Interaction&lt; VelIndep, VelDep, ExtVelIndep, ExtVelDep &gt;</a> . . . . .	??
<a href="#">SpaceH::kahan&lt; T &gt;</a>	
Kahan number . . . . .	??
<a href="#">logH&lt; Particles &gt;</a>	
LogH extention algorithmatic regularization interface . . . . .	64
<a href="#">SpaceH::NewtonForce&lt; Dtype, ArraySize &gt;</a> . . . . .	??
<a href="#">SpaceH::chain::Node&lt; Scalar &gt;</a>	
Struture to store the relative distance and index of two particles . . . . .	??
<a href="#">NoRegu&lt; Particles &gt;</a>	
Ordinary algorithmatic regularization interface . . . . .	69
<a href="#">particles&lt; Dtype, ArraySize &gt;</a>	
Class of dynamical variable . . . . .	??
<a href="#">particleSystem&lt; Particles, Interaction &gt;</a>	
Base class of particle System . . . . .	71

<a href="#">SpaceH::PostNewtonian&lt; Scalar &gt;</a>	
Post newtonian pair interaction functor(c++ std11)	??
<a href="#">ProgressBar</a>	??
<a href="#">SpaceH::ProtoType&lt; Dtype, Size &gt;</a>	??
<a href="#">ReguParticles&lt; Dtype, ArraySize &gt;</a>	
Class of dynamical system with regularization variables	??
<a href="#">ReguSystem&lt; Particles, Interaction, Regularitor &gt;</a>	
Regularized particle System	??
<a href="#">symplectic10th&lt; ParticSys &gt;</a>	
Eighth order symplectic integrator	102
<a href="#">symplectic2th&lt; ParticSys &gt;</a>	
Second order symplectic integrator	103
<a href="#">symplectic4th&lt; ParticSys &gt;</a>	
Fourth order symplectic integrator	104
<a href="#">symplectic6th&lt; ParticSys &gt;</a>	
Sixth order symplectic integrator	106
<a href="#">symplectic8th&lt; ParticSys &gt;</a>	
Eighth order symplectic integrator	107
<a href="#">TTL&lt; Particles &gt;</a>	
Time Transform Leapfrog algorithmatic regularization interface	108
<a href="#">vec3&lt; T &gt;</a>	
Self 3D vector class	110
<a href="#">SpaceH::VelDepForce&lt; PairForce, Dtype, ArraySize &gt;</a>	??
<a href="#">SpaceH::VelIndepForce&lt; PairForce, Dtype, ArraySize &gt;</a>	??



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">dynamicSystem.h</a>	121
<a href="#">errhand.h</a>	123
<a href="#">kahanNumber.h</a>	??
<a href="#">libs.h</a>	128
<a href="#">macros.h</a>	139
<a href="#">particles.h</a>	??
<a href="#">particleSystem.h</a>	145
<a href="#">protoType.h</a>	??
<a href="#">spaceX.h</a>	154
<a href="#">vector3.h</a>	159
<a href="#">integrator/symplectic/symplectic10th.h</a>	125
<a href="#">integrator/symplectic/symplectic2th.h</a>	125
<a href="#">integrator/symplectic/symplectic4th.h</a>	126
<a href="#">integrator/symplectic/symplectic6th.h</a>	126
<a href="#">integrator/symplectic/symplectic8th.h</a>	126
<a href="#">interaction/forces.h</a>	??
<a href="#">interaction/interaction.h</a>	126
<a href="#">ODEiterator/BSIterator.h</a>	143
<a href="#">ODEiterator/constIterator.h</a>	144
<a href="#">ODEiterator/dichotomy.h</a>	??
<a href="#">particleSystem/ARChain/ARchain.h</a>	146
<a href="#">particleSystem/ARChain/chain.h</a>	147
<a href="#">particleSystem/ARChain/dynamicChain.h</a>	??
<a href="#">particleSystem/reguSystem/regularization.h</a>	150
<a href="#">particleSystem/reguSystem/regularState.h</a>	151
<a href="#">particleSystem/reguSystem/reguSystem.h</a>	153
<a href="#">test/main.cpp</a>	156
<a href="#">test/typetest.cpp</a>	??
<a href="#">tools/timmer.h</a>	??
<a href="#">unitTest/testCompond.cpp</a>	157
<a href="#">unitTest/testVector.cpp</a>	158



## Chapter 6

# Namespace Documentation

### 6.1 NOTICE Namespace Reference

#### Functions

- void [Telegram](#) (const char \*host, const char \*msg)
- void [Title](#) (const char \*T)
- void [SubTitle](#) (const char \*T)
- void [EraseLine](#) ()
- void [Line](#) ()
- void [SubLine](#) ()
- void [RunInfo](#) (double timeLimit, double outputsize\_terval, double tolerance)

#### Variables

- constexpr size\_t [WIDTH](#) = 80
- bool [Message](#)

#### 6.1.1 Function Documentation

##### 6.1.1.1 EraseLine()

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

Here is the caller graph for this function:



#### 6.1.1.2 Line()

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

#### 6.1.1.3 RunInfo()

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

#### 6.1.1.4 SubLine()

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

#### 6.1.1.5 SubTitle()

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

#### 6.1.1.6 Telegram()

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

#### 6.1.1.7 Title()

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

### 6.1.2 Variable Documentation

## 6.1.2.1 Message

```
bool NOTICE::Message
```

## 6.1.2.2 WIDTH

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

## 6.2 SpaceH Namespace Reference

### Namespaces

- [chain](#)

### Classes

- struct [ArrayWrapper](#)
- struct [ArrayWrapper< T, DYNAMICAL >](#)
- struct [EmptyForce](#)
- struct [ExtVelDepForce](#)
- struct [ExtVelIndepForce](#)
- struct [Force](#)
- struct [get\\_value\\_type](#)
- class [Interaction](#)
- struct [kahan](#)  
*Kahan number.*
- struct [NewtonForce](#)
- class [PostNewtonian](#)  
*Post newtonian pair interaction functor(c++ std11)*
- struct [ProtoType](#)
- struct [VelDepForce](#)
- struct [VelIndepForce](#)

### Enumerations

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

## Functions

- `template<typename T1 , typename T2 >`  
`const T2 min (const T1 &x, const T2 &y)`  
*Self [min](#)()*
- `template<typename T1 , typename T2 >`  
`const T2 max (const T1 &x, const T2 &y)`  
*Self [max](#)()*
- `template<class T >`  
`const T abs (const T &x)`  
*Self [abs](#)()*
- `template<typename Scalar1 , typename Scalar2 >`  
`void advanceScalar (Scalar1 &var, Scalar2 increase)`  
*Self [swap](#)()*
- `template<typename Scalar , typename Vector1 , typename Vector2 >`  
`void advanceVector (Vector1 &var, const Vector2 &increase, Scalar stepSize)`
- `template<typename ScalarArray , typename VectorArray >`  
`void moveToCMCoord (const ScalarArray &mass, VectorArray &phyVar)`  
*Move variables to central mass coordinates.*
- `template<typename Scalar , size_t N>`  
`double getKineticEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &vel)`  
*Calculate the kinetic energy of particles.*
- `template<typename Scalar , size_t N>`  
`double getPotentialEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &pos)`  
*Calculate the potential energy of particles.*
- `template<typename Scalar , size_t N>`  
`double getTotalEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &pos, const std::array< vec3< Scalar >, N > &vel)`  
*Calculate the total(potential + kinetic) energy of particles.*
- `template<typename T >`  
`void print (T &var)`  
*print an array. Used for debug*

## Variables

- `constexpr double INV\_C = 1 / C`
- `constexpr double INV\_C2 = INV\_C * INV\_C`
- `constexpr double INV\_C3 = INV\_C2 * INV\_C`
- `constexpr double INV\_C4 = INV\_C3 * INV\_C`
- `constexpr double INV\_C5 = INV\_C4 * INV\_C`
- `constexpr size_t DYNAMICAL = 0`

### 6.2.1 Enumeration Type Documentation

#### 6.2.1.1 DATASTRUCT

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

## Enumerator

PLAIN	
CHAIN	

## 6.2.1.2 EVENTTYPE

```
enum SpaceH::EVENTTYPE
```

## Enumerator

TDE	
MERGE	
ESCAPE	
DISRUPTED	
UNEVENTFUL	
HVS	

## 6.2.1.3 INTEGRATORTYPE

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

## Enumerator

DKDLEAPFROG	
KDKLEAPFROG	
SYM4	
PEFRL	
SYM6	
SYM8	
SYM10	

## 6.2.1.4 ITERTYPE

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

## Enumerator

BSITER	
SEQITER	

### 6.2.1.5 PARTICTYPE

```
enum SpaceH::PARTICTYPE
```

#### Enumerator

NEUTRONSTAR	
STAR	
BLACKHOLE	
POINT	
NONE	

### 6.2.1.6 REGUTYPE

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

#### Enumerator

LOGH	
TTL	
NONE	

### 6.2.1.7 SYSTEMTYPE

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

#### Enumerator

PLAIN	
CHAIN	

## 6.2.2 Function Documentation

### 6.2.2.1 abs()

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



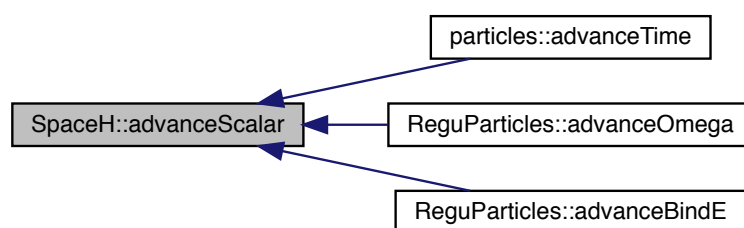
Self [abs\(\)](#)

### 6.2.2.2 advanceScalar()

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

Self [swap\(\)](#)

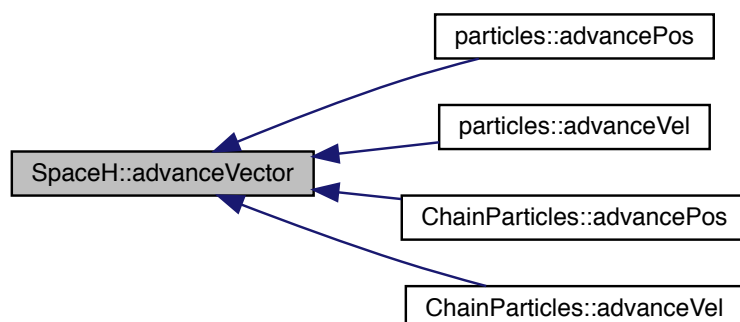
Here is the caller graph for this function:



### 6.2.2.3 advanceVector()

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

Here is the caller graph for this function:



#### 6.2.2.4 getKineticEnergy()

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

Calculate the kinetic energy of particles.

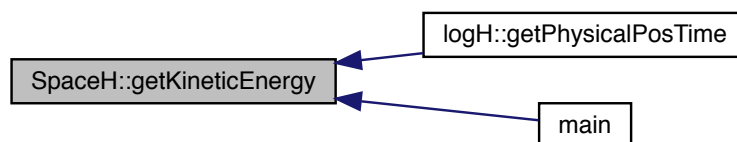
##### Parameters

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

##### Returns

The kinetic energy.

Here is the caller graph for this function:



#### 6.2.2.5 getPotentialEnergy()

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

Calculate the potential energy of particles.

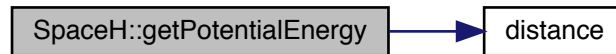
##### Parameters

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

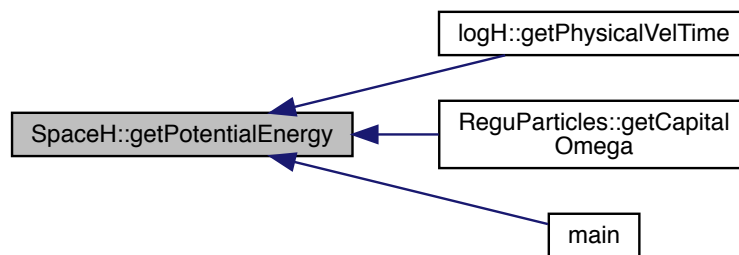
**Returns**

The potential energy.

Here is the call graph for this function:



Here is the caller graph for this function:

**6.2.2.6 getTotalEnergy()**

```

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

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

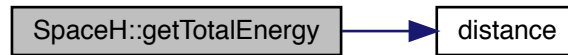
**Parameters**

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

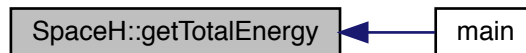
**Returns**

The total energy.

Here is the call graph for this function:



Here is the caller graph for this function:

**6.2.2.7 max()**

```

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

Self [max\(\)](#)

**6.2.2.8 min()**

```

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

Self [min\(\)](#)

**6.2.2.9 moveToCMCoord()**

```

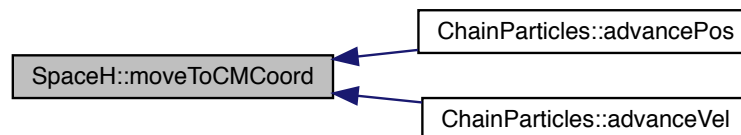
template<typename ScalarArray , typename VectorArray >
void SpaceH::moveToCMCoord (
    const ScalarArray & mass,
    VectorArray & phyVar )
  
```

Move variables to central mass coordinates.

## Parameters

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

Here is the caller graph for this function:



## 6.2.2.10 print()

```

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

print an array. Used for debug

## 6.2.3 Variable Documentation

## 6.2.3.1 DYNAMICAL

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

## 6.2.3.2 INV\_C

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

### 6.2.3.3 INV\_C2

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

### 6.2.3.4 INV\_C3

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

### 6.2.3.5 INV\_C4

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

### 6.2.3.6 INV\_C5

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

## 6.3 SpaceH::chain Namespace Reference

### Classes

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

### Functions

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

### 6.3.1 Function Documentation

#### 6.3.1.1 createAdjMartix()

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

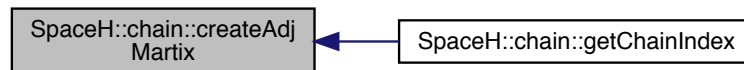
Create the adjoint matrix for particle pairs.

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

##### Parameters

<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 caller graph for this function:



#### 6.3.1.2 createChainIndex()

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

Create mapping index from adjoint matrix.

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

##### Parameters

<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:



### 6.3.1.3 getChainIndex()

```

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

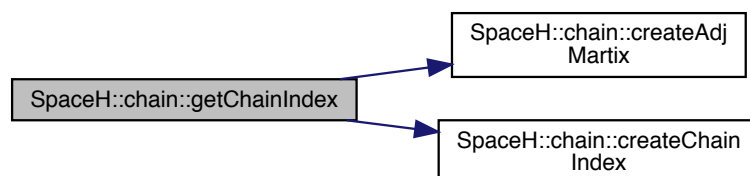
Calculate the mapping index from Cartesian coordinate to chain coordinate.

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

#### Parameters

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

Here is the call graph for this function:



### 6.3.1.4 IsDiff()

```

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



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

Check if two mapping indexes are the same.

Checking the identity of two chain index mappings.

#### Parameters

<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]

#### 6.3.1.5 synCartesian()

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

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

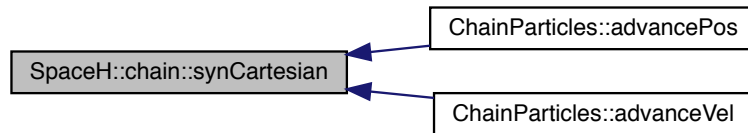
#### Parameters

<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:

**6.3.1.6 synChain()**

```

template<typename VectorArray , typename IndexArray >
void SpaceH::chain::synChain (
    VectorArray & data,
    VectorArray & chainData,
    IndexArray & 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\(\)](#).

**6.3.1.7 updateChain()**

```

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

Update the position chain.

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

## Parameters

<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.



## Chapter 7

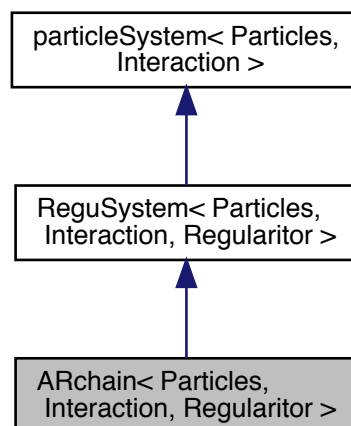
# Class Documentation

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

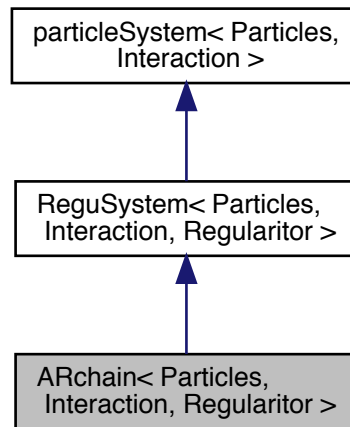
Algorithmatic Regularization chain System.

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

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



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



## Public Types

- using [Base](#) = [particleSystem](#)< Particles, Interaction >
- using [Scalar](#) = typename type::Scalar
- using [Vector](#) = typename type::Vector
- using [VectorArray](#) = typename type::VectorArray
- using [type](#) = typename Particles::type
- using [ActiveScalarArray](#) = typename Particles::ActiveScalarArray

## Public Member Functions

- void [kick](#) ([Scalar](#) stepSize)  
*Advance velocity one step with current acceleration.*

## Public Attributes

- Interaction [act](#)  
*Interaction class.*
- Particles [partc](#)  
*Particle class.*

## Additional Inherited Members

### 7.1.1 Detailed Description

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

Algorithmatic Regularization chain System.

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

## 7.1.2 Member Typedef Documentation

### 7.1.2.1 ActiveScalarArray

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

### 7.1.2.2 Base

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

### 7.1.2.3 Scalar

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

### 7.1.2.4 type

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

### 7.1.2.5 Vector

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

### 7.1.2.6 VectorArray

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

## 7.1.3 Member Function Documentation

### 7.1.3.1 kick()

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

Advance velocity one step with current acceleration.

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

## Parameters

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

## 7.1.4 Member Data Documentation

### 7.1.4.1 act

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

Interaction class.

### 7.1.4.2 partc

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

Particle class.

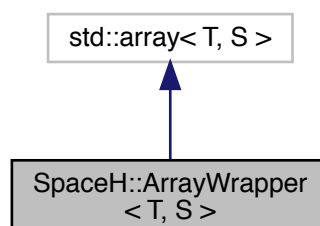
The documentation for this class was generated from the following file:

- particleSystem/ARChain/[ARchain.h](#)

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

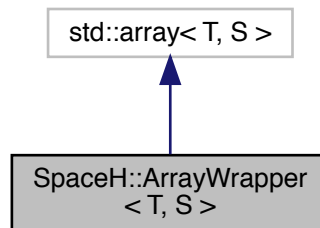
```
#include <protoType.h>
```

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





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



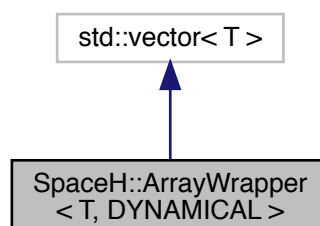
The documentation for this struct was generated from the following file:

- [protoType.h](#)

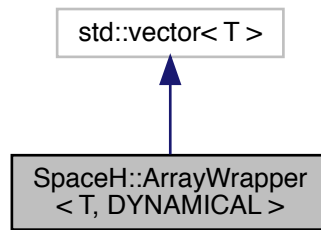
## 7.3 SpaceH::ArrayWrapper< T, DYNAMICAL > Struct Template Reference

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

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



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



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

- [protoType.h](#)

## 7.4 BSIterator< ParticSys, Integrator > Class Template Reference

Bulirsch-Stoer extrapolation algorithm.

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

### Public Types

- using [type](#) = typename ParticSys::type
- using [Scalar](#) = typename type::Scalar
- using [ActiveScalarArray](#) = typename ParticSys::ActiveScalarArray
- template<typename T, size\_t S>  
using [Container](#) = typename type::template [Container](#)< T, S >

### Public Member Functions

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

### 7.4.1 Detailed Description

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

Bulirsch-Stoer extrapolation algorithm.

### 7.4.2 Member Typedef Documentation

#### 7.4.2.1 ActiveScalarArray

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

#### 7.4.2.2 Container

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

#### 7.4.2.3 Scalar

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

#### 7.4.2.4 type

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

### 7.4.3 Constructor & Destructor Documentation

#### 7.4.3.1 BSiterator()

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

Constructor for initializing cost, nSteps, fmin and CC.

### 7.4.4 Member Function Documentation

#### 7.4.4.1 iterate()

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

Interface of ODE iterator.

#### Note

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

#### Parameters

<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.

#### 7.4.4.2 setAbsoluteError()

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

Set the local absolute error.

## 7.4.4.3 setRelativeError()

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

Set the local relative error.

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

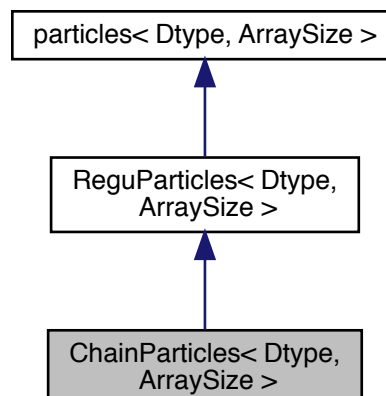
- ODEiterator/[BSIterator.h](#)

## 7.5 ChainParticles&lt; Dtype, ArraySize &gt; Class Template Reference

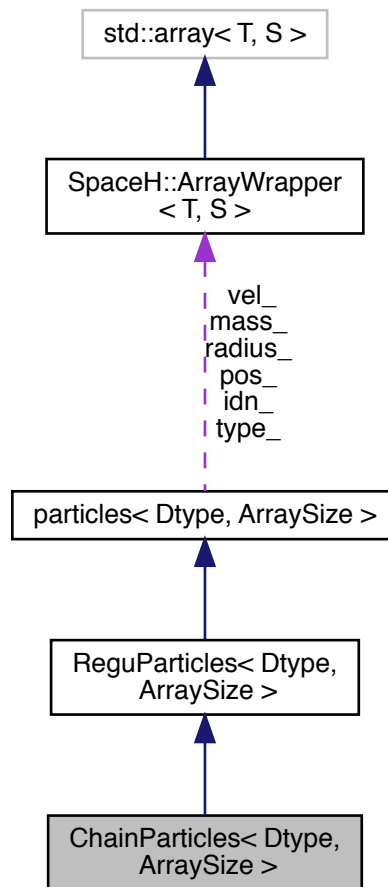
Class of dynamical variable.

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

Inheritance diagram for ChainParticles< Dtype, ArraySize >:



Collaboration diagram for ChainParticles< Dtype, ArraySize >:



## Public Types

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

## Public Member Functions

- `const VectorArray & chainPos () const`  
Position array const interface. Reference to `pos_`.
- `const VectorArray & chainVel () const`  
Velocity array const interface. Reference to `vel_`.
- `const IndexArray & chainIndex () const`

- Index array const interface. Reference to ch\_index\_.*
- const [Vector](#) & [chainPos](#) (size\_t i) const
- Position vector const interface. Reference to pos\_[i].*
- const [Vector](#) & [chainVel](#) (size\_t i) const
- Velocity vecotr const interface. Reference to vel\_[i].*
- const size\_t [chainIndex](#) (size\_t i) const
- Index const interface. Reference to ch\_index\_[i].*
- void [advancePos](#) ([Scalar](#) stepSize)
- Advance the position array with internal velocity array.*
- void [advanceVel](#) (const [VectorArray](#) &acc, [Scalar](#) stepSize)
- Advance the velocity array with given acceleration array.*

### Static Public Attributes

- static constexpr [SpaceH::DATASTRUCT](#) dataStruct {[SpaceH::DATASTRUCT::CHAIN](#)}

### Friends

- std::istream & [operator>>](#) (std::istream &is, [ChainParticles](#) &partc)
- Input(Initialize) variables with istream.*
- [ActiveScalarArray](#) & [operator>>](#) ([ActiveScalarArray](#) &data, [ChainParticles](#) &partc)
- Input variables with plain scalar array.*
- [ActiveScalarArray](#) & [operator<<](#) ([ActiveScalarArray](#) &data, const [ChainParticles](#) &partc)
- Output variables to plain scalar array.*

### Additional Inherited Members

#### 7.5.1 Detailed Description

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

Class of dynamical variable.

#### 7.5.2 Member Typedef Documentation

##### 7.5.2.1 ActiveScalarArray

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

### 7.5.2.2 Base

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

### 7.5.2.3 IndexArray

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

### 7.5.2.4 Scalar

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

### 7.5.2.5 Vector

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

### 7.5.2.6 VectorArray

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

## 7.5.3 Member Function Documentation

### 7.5.3.1 advancePos()

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

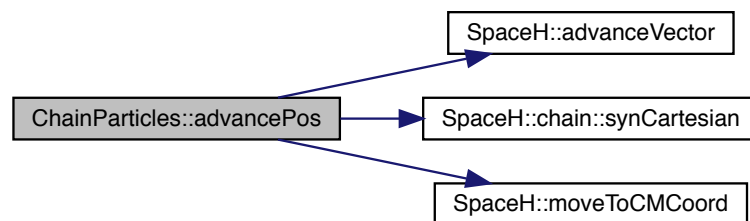
Advance the position array with internal velocity array.



## Parameters

<i>stepSize</i>	The advance step size.
-----------------	------------------------

Here is the call graph for this function:



## 7.5.3.2 advanceVel()

```

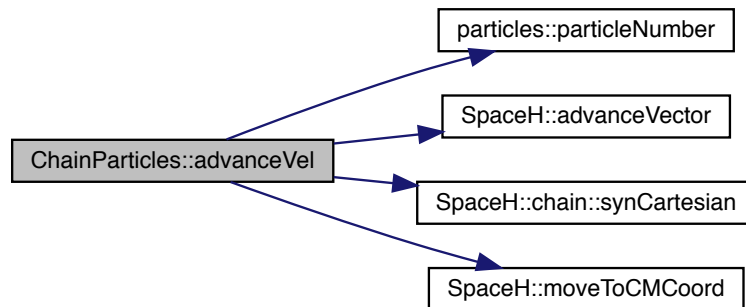
template<typename Dtype , size_t ArraySize>
void ChainParticles< Dtype, ArraySize >::advanceVel (
    const VectorArray & acc,
    Scalar stepSize ) [inline]
  
```

Advance the velocity array with given acceleration array.

## Parameters

<i>stepSize</i>	The advance step size.
<i>acc</i>	The acceleration array.

Here is the call graph for this function:



#### 7.5.3.3 chainIndex() [1/2]

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

Index array const interface. Reference to `ch_index_`.

#### 7.5.3.4 chainIndex() [2/2]

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

Index const interface. Reference to `ch_index_[i]`.

#### 7.5.3.5 chainPos() [1/2]

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

Position array const interface. Reference to `pos_`.

**7.5.3.6 chainPos()** [2/2]

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

Position vector const interface. Reference to pos\_[i].

**7.5.3.7 chainVel()** [1/2]

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

Velocity array const interface. Reference to vel\_.

**7.5.3.8 chainVel()** [2/2]

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

Velocity vecotr const interface. Reference to vel\_[i].

**7.5.4 Friends And Related Function Documentation****7.5.4.1 operator<<**

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

Output variables to plain scalar array.

**7.5.4.2 operator>>** [1/2]

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

Input(Initialize) variables with istream.

### 7.5.4.3 operator>> [2/2]

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

Input variables with plain scalar array.

## 7.5.5 Member Data Documentation

### 7.5.5.1 dataStruct

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

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

- particleSystem/ARChain/[dynamicChain.h](#)

## 7.6 constIterator< ParticSys, Integrator > Class Template Reference

Most common iterator.

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

### Public Types

- using [type](#) = typename ParticSys::type
- using [Scalar](#) = typename type::Scalar

### Public Member Functions

- [Scalar iterate](#) (ParticSys &[particles](#), [Scalar](#) stepLength)  
*interface to iterate particle system for one step*

### 7.6.1 Detailed Description

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

Most common iterator.

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

## 7.6.2 Member Typedef Documentation

### 7.6.2.1 Scalar

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

### 7.6.2.2 type

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

## 7.6.3 Member Function Documentation

### 7.6.3.1 iterate()

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

interface to iterate particle system for one step

#### Parameters

<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.

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

- ODEiterator/[constiterator.h](#)

## 7.7 dicholterator< ParticSys, Integrator > Class Template Reference

Dichotomy iterator.

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

## Public Types

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

## Public Member Functions

- [Scalar](#) [iterate](#) (ParticSys &[particles](#), Integrator &[integrator](#), [Scalar](#) [stepLength](#))  
*interface to iterate particle system for one step*
- void [setRelativeError](#) ([Scalar](#) [relError](#))  
*Set the local relative error.*
- void [setAbsoluteError](#) ([Scalar](#) [absError](#))  
*Set the local absolute error.*

### 7.7.1 Detailed Description

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

Dichotomy iterator.

Dichotomy iterator, use dichotomy to adjust the step size.

### 7.7.2 Member Typedef Documentation

#### 7.7.2.1 PlainArray

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

#### 7.7.2.2 Scalar

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

### 7.7.3 Member Function Documentation

#### 7.7.3.1 iterate()

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

interface to iterate particle system for one step

## Parameters

<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.

## 7.7.3.2 setAbsoluteError()

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

Set the local absolute error.

## 7.7.3.3 setRelativeError()

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

Set the local relative error.

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

- ODEiterator/[dichotomy.h](#)

## 7.8 dtype&lt; T &gt; Struct Template Reference

## Public Types

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

## Static Public Member Functions

- template<typename U >  
static U::value\_type [check](#) (typename U::value\_type)
- template<typename U >  
static U [check](#) (U)

## 7.8.1 Member Typedef Documentation

### 7.8.1.1 type

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

## 7.8.2 Member Function Documentation

### 7.8.2.1 check() [1/2]

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

### 7.8.2.2 check() [2/2]

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

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

- [test/typetest.cpp](#)

## 7.9 dynamicSystem< ParticSys, ODEiterator > Class Template Reference

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

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

### Public Types

- using [type](#) = typename ParticSys::type
- using [Scalar](#) = typename type::Scalar



## Public Member Functions

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

## Public Attributes

- Scalar [stepLength](#) {0.0}  
*Macro step size for ODE iterator.*
- int [steps](#) {0}  
*Steps.*
- ParticSys [particles](#)  
*Particle system.*
- ODEiterator [iterator](#)  
*ODE Iterator.*

### 7.9.1 Detailed Description

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

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

### 7.9.2 Member Typedef Documentation

#### 7.9.2.1 Scalar

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

#### 7.9.2.2 type

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

### 7.9.3 Constructor & Destructor Documentation

#### 7.9.3.1 ~dynamicSystem()

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

Default destructor, virtualize for inherent class.

### 7.9.4 Member Function Documentation

#### 7.9.4.1 advanceOneStep()

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

Advance the particle system for one step.

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

#### 7.9.4.2 loadText()

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

Load particle system initial condition from file.

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

#### Parameters

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

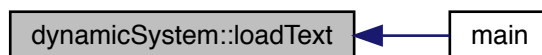
#### Exceptions

<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 call graph for this function:



Here is the caller graph for this function:



#### 7.9.4.3 setStepLength()

```

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

Set the step length.

### 7.9.5 Member Data Documentation

#### 7.9.5.1 iterator

```

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

ODE Iterator.

### 7.9.5.2 particles

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

Particle system.

### 7.9.5.3 stepLength

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

Macro step size for ODE iterator.

### 7.9.5.4 steps

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

Steps.

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

- [dynamicSystem.h](#)

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

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

### Public Types

- using [type](#) = [SpaceH::ProtoType](#)< Dtype, ArraySize >
- using [Scalar](#) = typename [type::Scalar](#)
- using [Vector](#) = typename [type::Vector](#)
- using [VectorArray](#) = typename [type::VectorArray](#)
- using [ScalarArray](#) = typename [type::ScalarArray](#)
- using [IndexArray](#) = typename [type::IndexArray](#)

## Public Member Functions

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

## Static Public Attributes

- static constexpr bool `isVelDep` {false}

## 7.10.1 Member Typedef Documentation

### 7.10.1.1 IndexArray

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

### 7.10.1.2 Scalar

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

### 7.10.1.3 ScalarArray

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

### 7.10.1.4 type

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

### 7.10.1.5 Vector

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

### 7.10.1.6 VectorArray

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

## 7.10.2 Member Function Documentation

### 7.10.2.1 acc() [1/2]

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

### 7.10.2.2 acc() [2/2]

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

### 7.10.2.3 addTotal()

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

### 7.10.2.4 calcuAcc() [1/4]

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

7.10.2.5 `calcuAcc()` [2/4]

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

7.10.2.6 `calcuAcc()` [3/4]

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

7.10.2.7 `calcuAcc()` [4/4]

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

## 7.10.3 Member Data Documentation

7.10.3.1 `isVelDep`

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

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

- [interaction/forces.h](#)

## 7.11 errhand Class Reference

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

## Public Member Functions

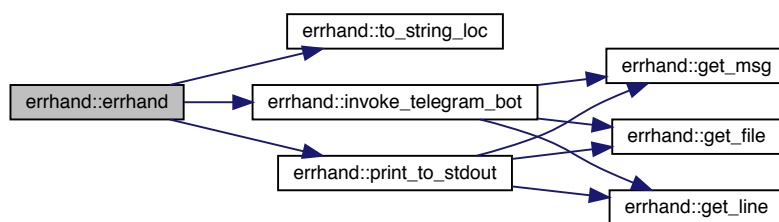
- [errhand](#) (std::string err\_msg\_input, const char \*file\_input, size\_t line\_input)
- std::string [get\\_msg](#) () const
- std::string [get\\_file](#) () const
- size\_t [get\\_line](#) () const
- std::string [to\\_string\\_loc](#) (const char \*obj)
- void [invoke\\_telegram\\_bot](#) ()
- void [print\\_to\\_stdout](#) ()

### 7.11.1 Constructor & Destructor Documentation

#### 7.11.1.1 errhand()

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

Here is the call graph for this function:

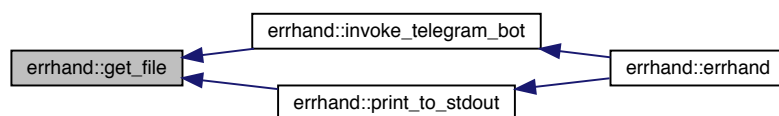


### 7.11.2 Member Function Documentation

#### 7.11.2.1 get\_file()

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

Here is the caller graph for this function:

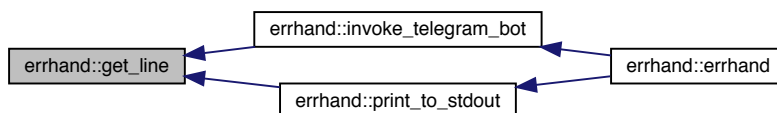




### 7.11.2.2 get\_line()

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

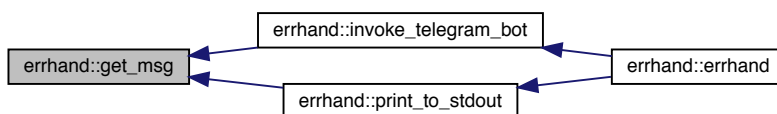
Here is the caller graph for this function:



### 7.11.2.3 get\_msg()

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

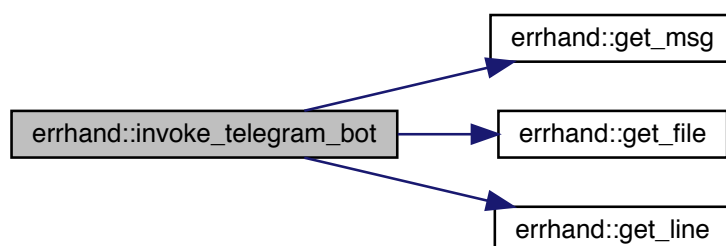
Here is the caller graph for this function:



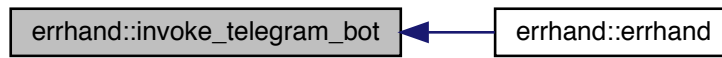
### 7.11.2.4 invoke\_telegram\_bot()

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

Here is the call graph for this function:



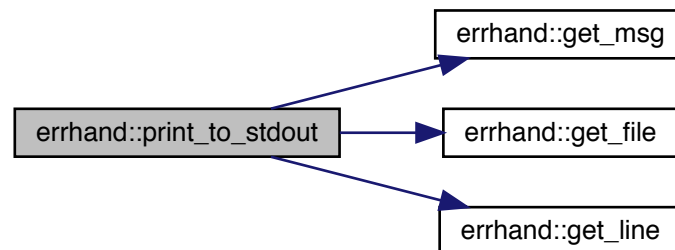
Here is the caller graph for this function:



#### 7.11.2.5 `print_to_stdout()`

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

Here is the call graph for this function:



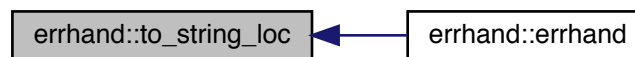
Here is the caller graph for this function:



## 7.11.2.6 to\_string\_loc()

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

Here is the caller graph for this function:



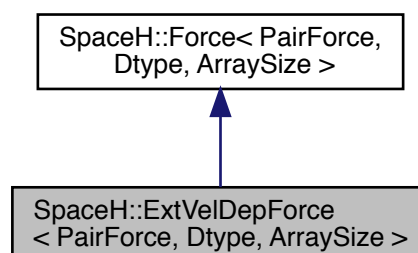
The documentation for this class was generated from the following file:

- [errhand.h](#)

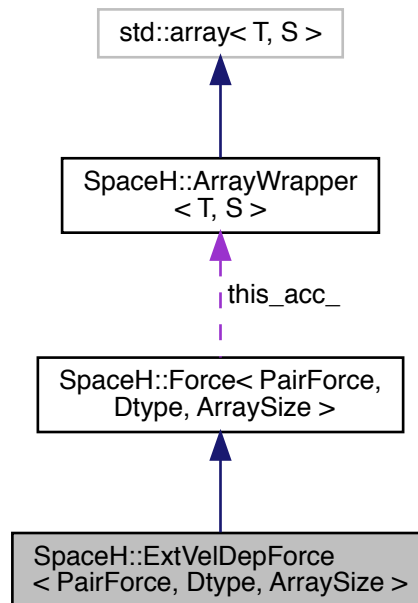
## 7.12 SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize > Struct Template Reference

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

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



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



## Public Types

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

## Public Member Functions

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

## Public Attributes

- PairForce `force_`
- `VectorArray` `this_acc_`

## Static Public Attributes

- static constexpr bool `isVelDep` {true}

## Additional Inherited Members

### 7.12.1 Member Typedef Documentation

#### 7.12.1.1 Base

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

#### 7.12.1.2 ScalarArray

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

#### 7.12.1.3 type

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

#### 7.12.1.4 VectorArray

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

### 7.12.2 Member Function Documentation

#### 7.12.2.1 calcuAcc()

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

### 7.12.3 Member Data Documentation

#### 7.12.3.1 force\_

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

#### 7.12.3.2 isVelDep

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

#### 7.12.3.3 this\_acc\_

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

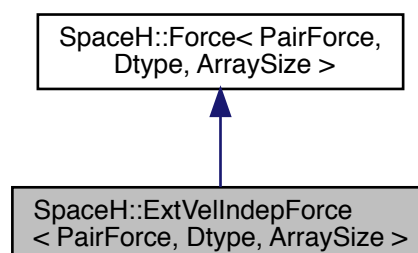
The documentation for this struct was generated from the following file:

- [interaction/forces.h](#)

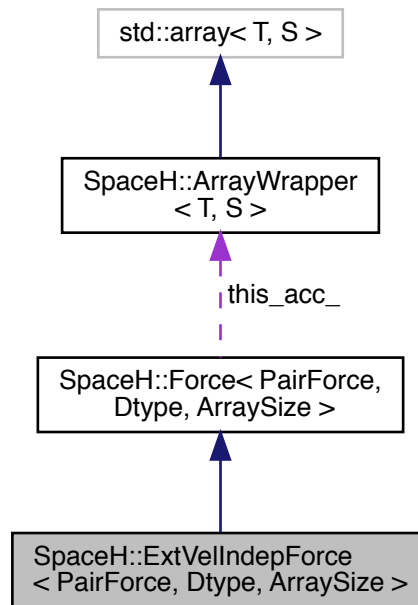
## 7.13 SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize > Struct Template Reference

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

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



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



## Public Types

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

## Public Member Functions

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

## Public Attributes

- `PairForce` `force_`
- `VectorArray` `this_acc_`

## Static Public Attributes

- static constexpr bool `isVelDep` {false}

## Additional Inherited Members

### 7.13.1 Member Typedef Documentation

#### 7.13.1.1 Base

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

#### 7.13.1.2 ScalarArray

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

#### 7.13.1.3 type

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

#### 7.13.1.4 VectorArray

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

### 7.13.2 Member Function Documentation

#### 7.13.2.1 calcuAcc()

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



### 7.13.3 Member Data Documentation

#### 7.13.3.1 force\_

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

#### 7.13.3.2 isVelDep

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

#### 7.13.3.3 this\_acc\_

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

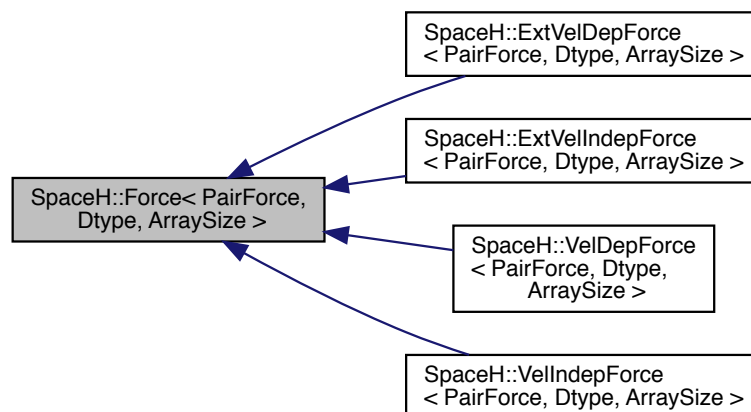
The documentation for this struct was generated from the following file:

- [interaction/forces.h](#)

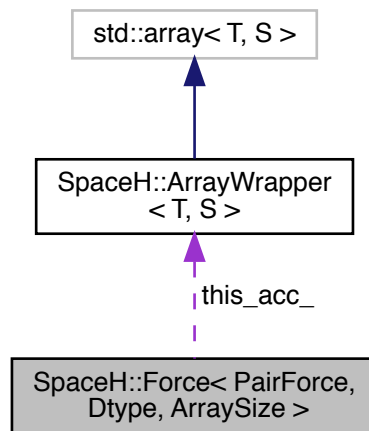
## 7.14 SpaceH::Force< PairForce, Dtype, ArraySize > Struct Template Reference

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

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



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



## Public Types

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

## Public Member Functions

- void `addTotal` (`VectorArray` &`acc`)
- const `VectorArray` &`acc` ()
- const `Vector` &`acc` (size\_t `i`)

## Protected Attributes

- `VectorArray` `this_acc_`
- `PairForce` `force_`

## 7.14.1 Member Typedef Documentation

### 7.14.1.1 `type`

```

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

```

## 7.14.1.2 Vector

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

## 7.14.1.3 VectorArray

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

## 7.14.2 Member Function Documentation

## 7.14.2.1 acc() [1/2]

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

Here is the caller graph for this function:



## 7.14.2.2 acc() [2/2]

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

### 7.14.2.3 addTotal()

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

Here is the call graph for this function:



## 7.14.3 Member Data Documentation

### 7.14.3.1 force\_

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

### 7.14.3.2 this\_acc\_

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

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

- [interaction/forces.h](#)

## 7.15 SpaceH::get\_value\_type< T > Struct Template Reference

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

### Public Types

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

## 7.15.1 Member Typedef Documentation

### 7.15.1.1 type

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

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

- [protoType.h](#)

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

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

### Public Types

- using [type](#) = typename VelIndep::type
- using [Scalar](#) = typename type::Scalar
- using [Vector](#) = typename type::Vector
- using [VectorArray](#) = typename type::VectorArray
- using [ScalarArray](#) = typename type::ScalarArray

### Public Member Functions

- const [VectorArray](#) & [totalAcc](#) ()
- const [VectorArray](#) & [velIndepAcc](#) ()
- const [VectorArray](#) & [velDepAcc](#) ()
- const [VectorArray](#) & [extVelIndepAcc](#) ()
- const [VectorArray](#) & [extVelDepAcc](#) ()
- const [Vector](#) & [totalAcc](#) (size\_t i)
- const [Vector](#) & [velIndepAcc](#) (size\_t i)
- const [Vector](#) & [velDepAcc](#) (size\_t i)
- const [Vector](#) & [extVelIndepAcc](#) (size\_t i)
- const [Vector](#) & [extVelDepAcc](#) (size\_t i)
- template<typename Particles >  
std::enable\_if< Particles::dataStruct==SpaceH::DATASTRUCT::PLAIN >::type [calcuVelIndepAcc](#) (const Particles &partc)
- template<typename Particles >  
std::enable\_if< Particles::dataStruct==SpaceH::DATASTRUCT::CHAIN >::type [calcuVelIndepAcc](#) (const Particles &partc)
- template<typename Particles >  
std::enable\_if< Particles::dataStruct==SpaceH::DATASTRUCT::PLAIN >::type [calcuVelDepAcc](#) (const Particles &partc)

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

## Static Public Attributes

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

## 7.16.1 Member Typedef Documentation

### 7.16.1.1 Scalar

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

### 7.16.1.2 ScalarArray

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

### 7.16.1.3 type

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

#### 7.16.1.4 Vector

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

#### 7.16.1.5 VectorArray

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

### 7.16.2 Member Function Documentation

#### 7.16.2.1 calcuAuxiVelDepAcc() [1/2]

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

#### 7.16.2.2 calcuAuxiVelDepAcc() [2/2]

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

#### 7.16.2.3 calcuExtAuxiVelDepAcc()

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

**7.16.2.4 `calcuExtVelDepAcc()`**

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

**7.16.2.5 `calcuExtVelIndepAcc()`**

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

**7.16.2.6 `calcuTotalAcc()`**

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

**7.16.2.7 `calcuVelDepAcc()` [1/2]**

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

**7.16.2.8 `calcuVelDepAcc()` [2/2]**

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



**7.16.2.9** `calcuVelIndepAcc()` [1/2]

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

**7.16.2.10** `calcuVelIndepAcc()` [2/2]

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

**7.16.2.11** `extVelDepAcc()` [1/2]

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

**7.16.2.12** `extVelDepAcc()` [2/2]

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

**7.16.2.13** `extVelIndepAcc()` [1/2]

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

**7.16.2.14** `extVelIndepAcc()` [2/2]

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

**7.16.2.15 totalAcc()** [1/2]

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

**7.16.2.16 totalAcc()** [2/2]

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

**7.16.2.17 velDepAcc()** [1/2]

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

**7.16.2.18 velDepAcc()** [2/2]

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

**7.16.2.19 velIndepAcc()** [1/2]

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

**7.16.2.20 velIndepAcc()** [2/2]

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

## 7.16.2.21 zeroTotalAcc()

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

## 7.16.3 Member Data Documentation

## 7.16.3.1 isVelDep

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

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

- interaction/[interaction.h](#)

## 7.17 SpaceH::kahan&lt; T &gt; Struct Template Reference

Kahan number.

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

## Public Types

- using [value\\_type](#) = T

## Public Member Functions

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

## Public Attributes

- T [real](#)
- T [err](#)

## Friends

- `kahan operator-` (const `kahan` &hs)
- const `kahan` & `operator+=` (`kahan` &lhs, const `kahan` &rhs)
- const `kahan` & `operator-=` (`kahan` &lhs, const `kahan` &rhs)
- const `kahan` & `operator/=` (`kahan` &lhs, const `kahan` &rhs)
- const `kahan` & `operator*=` (`kahan` &lhs, const `kahan` &rhs)
- `std::ostream` & `operator<<` (`std::ostream` &output, const `kahan` &v)  
*Output to ostream.*
- `std::istream` & `operator>>` (`std::istream` &input, `kahan` &v)  
*Input from istream.*

### 7.17.1 Detailed Description

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

Kahan number.

A way to reduce the round off error when adding a small number to a big one. See details in [https://en.wikipedia.org/wiki/Kahan\\_summation\\_algorithm](https://en.wikipedia.org/wiki/Kahan_summation_algorithm)

### 7.17.2 Member Typedef Documentation

#### 7.17.2.1 value\_type

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

### 7.17.3 Constructor & Destructor Documentation

#### 7.17.3.1 kahan() [1/3]

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

#### 7.17.3.2 kahan() [2/3]

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

### 7.17.3.3 kahan() [3/3]

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

## 7.17.4 Member Function Documentation

### 7.17.4.1 operator T() [1/2]

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

### 7.17.4.2 operator T() [2/2]

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

### 7.17.4.3 operator=()

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

### 7.17.4.4 zeroErr()

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

## 7.17.5 Friends And Related Function Documentation

#### 7.17.5.1 operator\*==

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

#### 7.17.5.2 operator+=

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

#### 7.17.5.3 operator-

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

#### 7.17.5.4 operator-=

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

#### 7.17.5.5 operator/=

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

#### 7.17.5.6 operator<<

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

Output to ostream.

## 7.17.5.7 operator&gt;&gt;

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

Input from istream.

## 7.17.6 Member Data Documentation

## 7.17.6.1 err

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

## 7.17.6.2 real

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

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

- [kahanNumber.h](#)

## 7.18 logH&lt; Particles &gt; Class Template Reference

[logH](#) extention algorithmatic regularization interface

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

## Public Types

- using [type](#) = typename Particles::type
- using [Scalar](#) = typename type::Scalar

## Public Member Functions

- [Scalar getPhysicalPosTime](#) (Particles &partc, [Scalar](#) stepSize)  
*Calculate the physical time for position advance from integration step size.*
- [Scalar getPhysicalVelTime](#) (Particles &partc, [Scalar](#) stepSize)  
*Calculate the physical time for velocity advance from integration step size.*

### 7.18.1 Detailed Description

```
template<typename Particles>
class logH< Particles >
```

`logH` extention algorithmatic regularization interface

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

### 7.18.2 Member Typedef Documentation

#### 7.18.2.1 Scalar

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

#### 7.18.2.2 type

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

### 7.18.3 Member Function Documentation

#### 7.18.3.1 getPhysicalPosTime()

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

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

#### Parameters

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



Here is the call graph for this function:



### 7.18.3.2 getPhysicalVelTime()

```

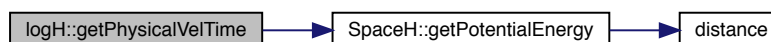
template<typename Particles >
Scalar logH< Particles >::getPhysicalVelTime (
    Particles & partc,
    Scalar stepSize ) [inline]
  
```

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

#### Parameters

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

Here is the call graph for this function:



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

- `particleSystem/reguSystem/regularization.h`

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

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

## Public Types

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

## Public Member Functions

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

## 7.19.1 Member Typedef Documentation

### 7.19.1.1 Scalar

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

### 7.19.1.2 ScalarArray

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

### 7.19.1.3 type

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

### 7.19.1.4 Vector

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

## 7.19.1.5 VectorArray

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

## 7.19.2 Member Function Documentation

## 7.19.2.1 operator&gt;()

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

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

- [interaction/forces.h](#)

## 7.20 SpaceH::chain::Node&lt; Scalar &gt; Struct Template Reference

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

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

## Public Attributes

- Scalar [Rij](#)
- [size\\_t i](#)
- [size\\_t j](#)
- bool [available](#)

## 7.20.1 Detailed Description

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

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

## 7.20.2 Member Data Documentation

### 7.20.2.1 available

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

State of node. If this node can be chained.

### 7.20.2.2 i

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

Particle index.

### 7.20.2.3 j

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

Particle index.

### 7.20.2.4 Rij

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

Relative distance of two particles.

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

- particleSystem/ARChain/[chain.h](#)

## 7.21 NoRegu< Particles > Class Template Reference

Ordinary algorithmatic regularization interface.

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

### Public Types

- using [type](#) = typename Particles::type
- using [Scalar](#) = typename type::Scalar

## Public Member Functions

- [Scalar getPhysicalPosTime](#) (Particles &partc, [Scalar](#) stepSize)  
*Calculate the physical time for position advance from integration step size.*
- [Scalar getPhysicalVelTime](#) (Particles &partc, [Scalar](#) stepSize)  
*Calculate the physical time for velocity advance from integration step size.*

### 7.21.1 Detailed Description

```
template<typename Particles>
class NoRegu< Particles >
```

Ordinary algorithmatic regularization interface.

No regularization.

### 7.21.2 Member Typedef Documentation

#### 7.21.2.1 Scalar

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

#### 7.21.2.2 type

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

### 7.21.3 Member Function Documentation

#### 7.21.3.1 getPhysicalPosTime()

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

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

## Parameters

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

7.21.3.2 `getPhysicalVelTime()`

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

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

## Parameters

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

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

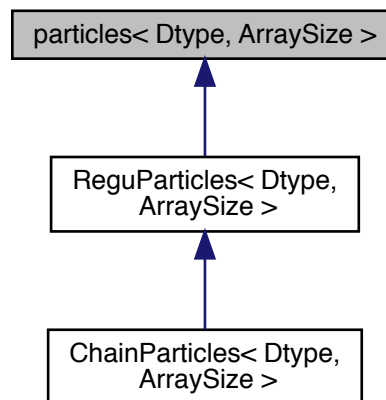
- `particleSystem/reguSystem/regularization.h`

7.22 `particles< Dtype, ArraySize >` Class Template Reference

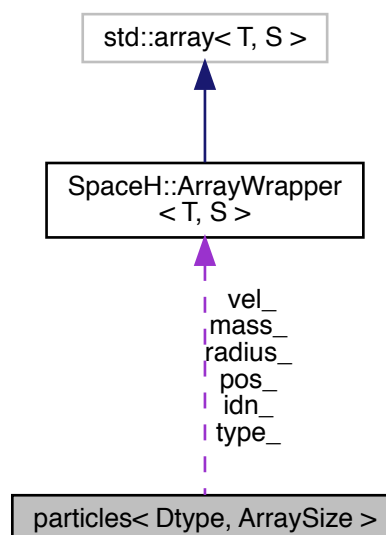
Class of dynamical variable.

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

Inheritance diagram for particles< Dtype, ArraySize >:



Collaboration diagram for particles< Dtype, ArraySize >:



## Public Types

- using `type` = `SpaceH::ProtoType< Dtype, ArraySize >`
- template<typename T, size\_t S>  
using `Container` = typename type::template `Container`< T, S >

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

## Public Member Functions

- `size_t particleNumber ()` const  
*Get the number of the particles.*
- `const Scalar & time ()` const  
*Physical time scalar const interface. Reference to state.time.*
- `const VectorArray & pos ()` const  
*Position array const interface. Reference to state.pos.*
- `const VectorArray & vel ()` const  
*Velocity array const interface. Reference to state.vel.*
- `const ScalarArray & mass ()` const  
*Mass array const interface. Reference to attribute.mass.*
- `const ScalarArray & radius ()` const  
*Radius array const interface. Reference to attribute.radius.*
- `const IntArray & kind ()` const  
*Particle type array const interface. Reference to attribute.type.*
- `const IntArray & idn ()` const  
*Particle id array const interface. Reference to attribute.type.*
- `const Vector & pos (size_t i)` const  
*Position vector const interface. Reference to state.pos[i].*
- `const Vector & vel (size_t i)` const  
*Velocity vecotr const interface. Reference to state.vel[i].*
- `const Scalar & mass (size_t i)` const  
*Mass const interface. Reference to attribute.mass[i].*
- `const Scalar & radius (size_t i)` const  
*Radius const interface. Reference to attribute.radius[i].*
- `const int & kind (size_t i)` const  
*Particle type const interface. Reference to attribute.type[i].*
- `const int & idn (size_t i)` const  
*Particle id const interface. Reference to attribute.type[i].*
- `void advanceTime (Scalar dt)`  
*Advance the time.*
- `void advancePos (Scalar stepSize)`  
*Advance the position array with internal velocity array.*
- `void advanceVel (const VectorArray &acc, Scalar stepSize)`  
*Advance the velocity array with given acceleration array.*

## Static Public Attributes

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



## Protected Attributes

- [VectorArray pos\\_](#)  
*Position array of the particles. Element is 3D vector.*
- [VectorArray vel\\_](#)  
*Velocity array of the particles. Element is 3D vector.*
- [ScalarArray mass\\_](#)  
*Mass array of the particles. Element is Scalar.*
- [ScalarArray radius\\_](#)  
*Radius array of the particles. Element is Scalar.*
- [IntArray type\\_](#)  
*Type Array of the particles. Element is int.*
- [IntArray idn\\_](#)  
*Id Array of the particles. Element is int.*
- [Scalar time\\_](#)  
*The physical time of the dynamic system.*
- [Scalar totalMass\\_](#)  
*The total mass of the system.*

## Friends

- `std::istream & operator>> (std::istream &is, particles &partc)`  
*Input(Initialize) variables with istream.*
- `std::ostream & operator<< (std::ostream &os, const particles &partc)`  
*Output variables to ostream.*
- `ActiveScalarArray & operator>> (ActiveScalarArray &data, particles &partc)`  
*Input variables with plain scalar array.*
- `ActiveScalarArray & operator<< (ActiveScalarArray &data, const particles &partc)`  
*Output variables to plain scalar array.*

### 7.22.1 Detailed Description

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

Class of dynamical variable.

### 7.22.2 Member Typedef Documentation

#### 7.22.2.1 ActiveScalarArray

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

#### 7.22.2.2 Container

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

#### 7.22.2.3 IntArray

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

#### 7.22.2.4 Scalar

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

#### 7.22.2.5 ScalarArray

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

#### 7.22.2.6 type

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

#### 7.22.2.7 Vector

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

#### 7.22.2.8 VectorArray

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

### 7.22.3 Member Function Documentation

#### 7.22.3.1 advancePos()

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

Advance the position array with internal velocity array.

## Parameters

<i>stepSize</i>	The advance step size.
-----------------	------------------------

Here is the call graph for this function:



## 7.22.3.2 advanceTime()

```

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

Advance the time.

## Parameters

<i>dt</i>	Time increament.
-----------	------------------

Here is the call graph for this function:



## 7.22.3.3 advanceVel()

```

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

Advance the velocity array with given acceleration array.

## Parameters

<i>stepSize</i>	The advance step size.
<i>acc</i>	The acceleration array.

Here is the call graph for this function:



## 7.22.3.4 idn() [1/2]

```

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

```

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

## 7.22.3.5 idn() [2/2]

```

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

```

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

## 7.22.3.6 kind() [1/2]

```

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

```

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

### 7.22.3.7 kind() [2/2]

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

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

### 7.22.3.8 mass() [1/2]

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

Mass array const interface. Reference to attribute.mass.

Here is the caller graph for this function:



### 7.22.3.9 mass() [2/2]

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

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

## 7.22.3.10 particleNumber()

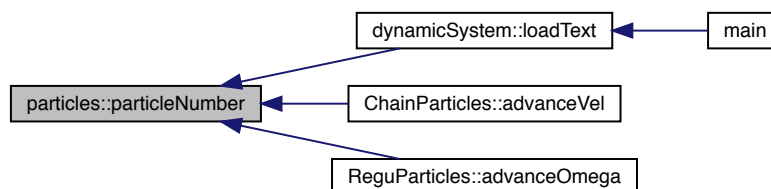
```
template<typename Dtype , size_t ArraySize>
size_t particles< Dtype, ArraySize >::particleNumber ( ) const [inline]
```

Get the number of the particles.

## Returns

The particle number.

Here is the caller graph for this function:



## 7.22.3.11 pos() [1/2]

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

Position array const interface. Reference to state.pos.

Here is the caller graph for this function:



**7.22.3.12 pos()** [2/2]

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

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

**7.22.3.13 radius()** [1/2]

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

Radius array const interface. Reference to attribute.radius.

**7.22.3.14 radius()** [2/2]

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

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

**7.22.3.15 time()**

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

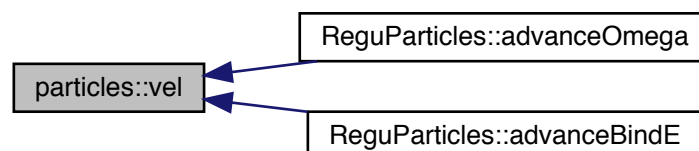
Physical time scalar const interface. Reference to state.time.

**7.22.3.16 vel()** [1/2]

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

Velocity array const interface. Reference to state.vel.

Here is the caller graph for this function:





## 7.22.3.17 vel() [2/2]

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

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

## 7.22.4 Friends And Related Function Documentation

## 7.22.4.1 operator&lt;&lt; [1/2]

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

Output variables to ostream.

## 7.22.4.2 operator&lt;&lt; [2/2]

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

Output variables to plain scalar array.

## 7.22.4.3 operator&gt;&gt; [1/2]

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

Input(Initialize) variables with istream.

#### 7.22.4.4 operator>> [2/2]

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

Input variables with plain scalar array.

### 7.22.5 Member Data Documentation

#### 7.22.5.1 activeScalar

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

#### 7.22.5.2 dataStruct

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

#### 7.22.5.3 idn\_

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

Id Array of the particles. Element is int.

#### 7.22.5.4 mass\_

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

Mass array of the particles. Element is Scalar.

#### 7.22.5.5 pos\_

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

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

#### 7.22.5.6 radius\_

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

Radius array of the particles. Element is Scalar.

#### 7.22.5.7 time\_

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

The physical time of the dynamic system.

#### 7.22.5.8 totalMass\_

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

The total mass of the system.

#### 7.22.5.9 type\_

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

Type Array of the particles. Element is int.

## 7.22.5.10 vel\_

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

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

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

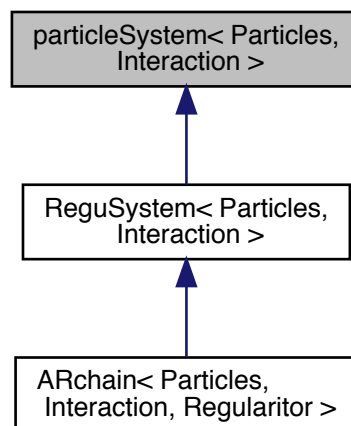
- [particles.h](#)

## 7.23 particleSystem&lt; Particles, Interaction &gt; Class Template Reference

Base class of particle System.

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

Inheritance diagram for particleSystem< Particles, Interaction >:



## Public Types

- using [type](#) = typename Particles::type
- using [Scalar](#) = typename type::Scalar
- using [Vector](#) = typename type::Vector
- using [VectorArray](#) = typename type::VectorArray
- using [ScalarArray](#) = typename type::ScalarArray
- using [IntArray](#) = typename type::IntArray
- using [ActiveScalarArray](#) = typename Particles::ActiveScalarArray

## Public Member Functions

- `size_t particleNumber ()`  
*Get the number of the particles.*
- `const Scalar & time () const`  
*Physical time scalar const interface. Reference to state.time.*
- `const VectorArray & pos () const`  
*Position array const interface. Reference to state.pos.*
- `const VectorArray & vel () const`  
*Velocity array const interface. Reference to state.vel.*
- `const ScalarArray & mass () const`  
*Mass array const interface. Reference to attribute.mass.*
- `const ScalarArray & radius () const`  
*Radius array const interface. Reference to attribute.radius.*
- `const IntArray & kind () const`  
*Particle type array const interface. Reference to attribute.type.*
- `const IntArray & idn () const`  
*Particle id array const interface. Reference to attribute.type.*
- `const Vector & pos (size_t i) const`  
*Position vector const interface. Reference to state.pos[i].*
- `const Vector & vel (size_t i) const`  
*Velocity vecotr const interface. Reference to state.vel[i].*
- `const Scalar & mass (size_t i) const`  
*Mass const interface. Reference to attribute.mass[i].*
- `const Scalar & radius (size_t i) const`  
*Radius const interface. Reference to attribute.radius[i].*
- `const int & kind (size_t i) const`  
*Particle type const interface. Reference to attribute.type[i].*
- `const int & idn (size_t i) const`  
*Particle id const interface. Reference to attribute.type[i].*
- `Scalar timeScale (Scalar scale)`  
*Interface to rescale the time.*
- `void drift (Scalar stepSize)`  
*Advance position one step with current velocity.*
- `void kick (Scalar stepSize)`  
*Advance velocity one step with current acceleration.*
- `void preIterProcess ()`  
*Preprocess before iteration.*
- `void afterIterProcess ()`  
*After process after iteration.*
- `virtual ~particleSystem ()`  
*Virtualize default destructor.*

## Static Public Attributes

- `static constexpr size_t arraySize {type::arraySize}`

## Protected Attributes

- Particles [partc](#)  
*Particle class.*
- Interaction [act](#)  
*Interaction class.*

## Friends

- `std::ostream & operator<< (std::ostream &os, const particleSystem &sys)`  
*Overload operator <<.*
- `std::istream & operator>> (std::istream &is, particleSystem &sys)`  
*Input from istream.*
- `ActiveScalarArray & operator>> (ActiveScalarArray &data, particleSystem &sys)`  
*Input variables with plain scalar array.*
- `ActiveScalarArray & operator<< (ActiveScalarArray &data, const particleSystem &sys)`  
*Output variables to plain scalar array.*

### 7.23.1 Detailed Description

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

Base class of particle System.

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

### 7.23.2 Member Typedef Documentation

#### 7.23.2.1 ActiveScalarArray

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

#### 7.23.2.2 IntArray

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

#### 7.23.2.3 Scalar

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

#### 7.23.2.4 ScalarArray

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

#### 7.23.2.5 type

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

#### 7.23.2.6 Vector

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

#### 7.23.2.7 VectorArray

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

### 7.23.3 Constructor & Destructor Documentation

#### 7.23.3.1 ~particleSystem()

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

Virtualize default destructor.

## 7.23.4 Member Function Documentation

### 7.23.4.1 afterIterProcess()

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

After process after iteration.

### 7.23.4.2 drift()

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

Advance position one step with current velocity.

### 7.23.4.3 idn() [1/2]

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

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

### 7.23.4.4 idn() [2/2]

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

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

### 7.23.4.5 kick()

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

Advance velocity one step with current acceleration.



**7.23.4.6 kind()** [1/2]

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

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

**7.23.4.7 kind()** [2/2]

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

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

**7.23.4.8 mass()** [1/2]

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

Mass array const interface. Reference to attribute.mass.

**7.23.4.9 mass()** [2/2]

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

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

**7.23.4.10 particleNumber()**

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

Get the number of the particles.

**Returns**

The particle number.

**7.23.4.11 pos()** [1/2]

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

Position array const interface. Reference to state.pos.

**7.23.4.12 pos()** [2/2]

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

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

**7.23.4.13 preIterProcess()**

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

Preprocess before iteration.

**7.23.4.14 radius()** [1/2]

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

Radius array const interface. Reference to attribute.radius.

**7.23.4.15 radius()** [2/2]

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

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

## 7.23.4.16 time()

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

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

## 7.23.4.17 timeScale()

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

Interface to rescale the time.

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

## Returns

The phsyical time.

## 7.23.4.18 vel() [1/2]

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

Velocity array const interface. Reference to state.vel.

## 7.23.4.19 vel() [2/2]

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

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

## 7.23.5 Friends And Related Function Documentation

#### 7.23.5.1 operator<< [1/2]

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

Overload operator <<.

#### 7.23.5.2 operator<< [2/2]

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

Output variables to plain scalar array.

#### 7.23.5.3 operator>> [1/2]

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

Input from istream.

#### 7.23.5.4 operator>> [2/2]

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

Input variables with plain scalar array.

### 7.23.6 Member Data Documentation

## 7.23.6.1 act

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

Interaction class.

## 7.23.6.2 arraySize

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

## 7.23.6.3 partc

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

Particle class.

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

- [particleSystem.h](#)

## 7.24 SpaceH::PostNewtonian&lt; Scalar &gt; Class Template Reference

Post newtonian pair interaction functor(c++ std11)

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

## Public Member Functions

- void [operator\(\)](#) (Scalar m1, Scalar m2, [Vector](#) &dr, [Vector](#) &dv, [Vector](#) &v1, [Vector](#) &v2, [Vector](#) &acc1, [Vector](#) &acc2)  
*Update the velocity dependent acceleration of particle 1 and 2.*

## 7.24.1 Detailed Description

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

Post newtonian pair interaction functor(c++ std11)

## 7.24.2 Member Function Documentation

### 7.24.2.1 operator()

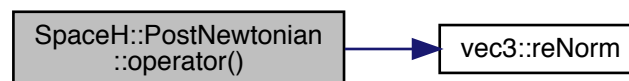
```
template<typename Scalar >
void SpaceH::PostNewtonian< Scalar >::operator() (
    Scalar m1,
    Scalar m2,
    Vector & dr,
    Vector & dv,
    Vector & v1,
    Vector & v2,
    Vector & acc1,
    Vector & acc2 ) [inline]
```

Update the velocity dependent acceleration of particle 1 and 2.

#### Parameters

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

Here is the call graph for this function:



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

- [interaction/forces.h](#)

## 7.25 ProgressBar Class Reference

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

## Public Types

- typedef std::chrono::high\_resolution\_clock [Clock](#)
- typedef std::chrono::time\_point< [Clock](#) > [ClockTime](#)
- typedef std::chrono::milliseconds [ms](#)

## Public Member Functions

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

## 7.25.1 Member Typedef Documentation

### 7.25.1.1 Clock

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

### 7.25.1.2 ClockTime

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

### 7.25.1.3 ms

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

## 7.25.2 Constructor & Destructor Documentation

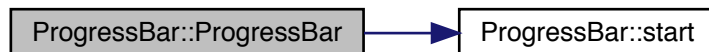
### 7.25.2.1 ProgressBar() [1/2]

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

### 7.25.2.2 ProgressBar() [2/2]

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

Here is the call graph for this function:



## 7.25.3 Member Function Documentation

### 7.25.3.1 autoShow()

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

Here is the call graph for this function:



### 7.25.3.2 getTime()

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



## 7.25.3.3 reset()

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

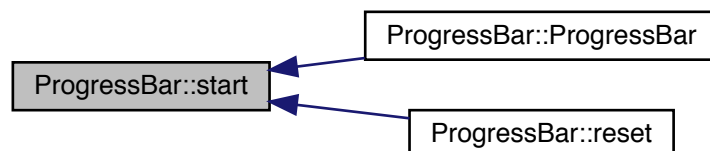
Here is the call graph for this function:



## 7.25.3.4 start()

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

Here is the caller graph for this function:



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

- [tools/timmer.h](#)

## 7.26 SpaceH::ProtoType&lt; Dtype, Size &gt; Struct Template Reference

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

## Public Types

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

## Static Public Attributes

- `static constexpr size_t arraySize {Size}`

## 7.26.1 Member Typedef Documentation

### 7.26.1.1 Container

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

### 7.26.1.2 IndexArray

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

### 7.26.1.3 IntArray

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

### 7.26.1.4 Scalar

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

#### 7.26.1.5 ScalarArray

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

#### 7.26.1.6 SizeArray

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

#### 7.26.1.7 Vector

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

#### 7.26.1.8 VectorArray

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

### 7.26.2 Member Data Documentation

#### 7.26.2.1 arraySize

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

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

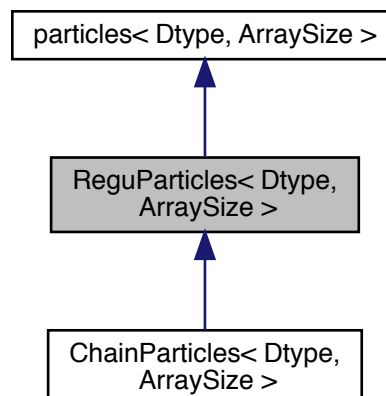
- [protoType.h](#)

## 7.27 ReguParticles< Dtype, ArraySize > Class Template Reference

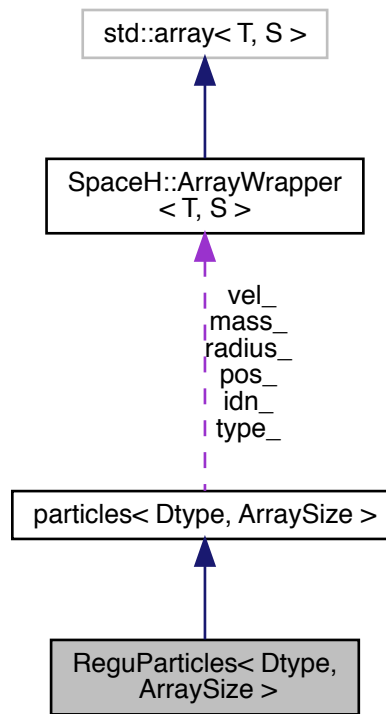
Class of dynamical system with regularization variables.

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

Inheritance diagram for ReguParticles< Dtype, ArraySize >:



Collaboration diagram for ReguParticles< Dtype, ArraySize >:



## Public Types

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

## Public Member Functions

- const `Scalar` & `omega` () const  
*Omega scalar const interface. Reference to state.time.*
- const `Scalar` & `bindE` () const  
*BindE scalar const interface. Reference to state.time.*
- void `advanceOmega` (const `VectorArray` &velIndepAcc, const `VectorArray` &vel, `Scalar` stepSize)  
*Advance the Omega.*
- void `advanceBindE` (const `VectorArray` &velDepAcc, const `VectorArray` &vel, `Scalar` stepSize)  
*Advance the bindE.*

## Static Public Attributes

- static constexpr size\_t `activeScalar` {6\*`type::arraySize` + 3}

## Protected Member Functions

- `Scalar getCapitalOmega ()`  
*Calculate the regularized variable Omega.*

## Protected Attributes

- `Scalar omega_`
- `Scalar bindE_`

## Friends

- `std::istream & operator>> (std::istream &is, ReguParticles &partc)`  
*Input(Initialize) variables with istream.*
- `ActiveScalarArray & operator>> (ActiveScalarArray &data, ReguParticles &partc)`  
*Input variables with plain scalar array.*
- `ActiveScalarArray & operator<< (ActiveScalarArray &data, const ReguParticles &partc)`  
*Output variables to plain scalar array.*

### 7.27.1 Detailed Description

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

Class of dynamical system with regularization variables.

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

### 7.27.2 Member Typedef Documentation

#### 7.27.2.1 ActiveScalarArray

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

### 7.27.2.2 Base

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

### 7.27.2.3 Container

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

### 7.27.2.4 Scalar

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

### 7.27.2.5 type

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

### 7.27.2.6 Vector

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

### 7.27.2.7 VectorArray

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

## 7.27.3 Member Function Documentation

### 7.27.3.1 advanceBindE()

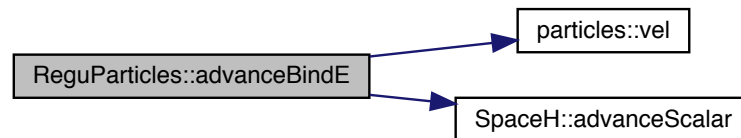
```
template<typename Dtype , size_t ArraySize>
void ReguParticles< Dtype, ArraySize >::advanceBindE (
    const VectorArray & velDepAcc,
    const VectorArray & vel,
    Scalar stepSize ) [inline]
```

Advance the bindE.

## Parameters

<i>velDepAcc</i>	Velocity dependent acceleration array.
<i>vel</i>	Velocity array.
<i>stepSize</i>	Time stepSize.

Here is the call graph for this function:

7.27.3.2 `advanceOmega()`

```

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

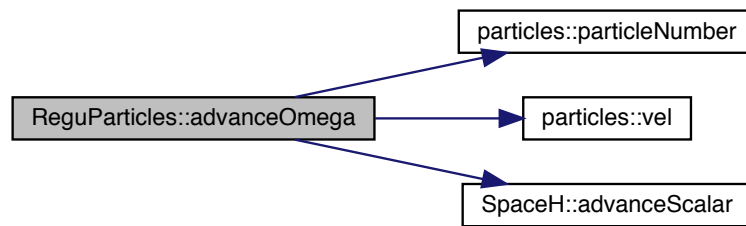
Advance the Omega.

## Parameters

<i>velIndepAcc</i>	Velocity independent acceleration array.
<i>vel</i>	Velocity array.
<i>stepSize</i>	Time stepSize.



Here is the call graph for this function:



### 7.27.3.3 bindE()

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

BindE scalar const interface. Reference to state.time.

### 7.27.3.4 getCapitalOmega()

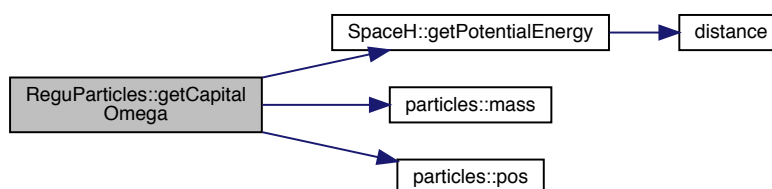
```
template<typename Dtype , size_t ArraySize>
Scalar ReguParticles< Dtype, ArraySize >::getCapitalOmega ( ) [inline], [protected]
```

Calculate the regularized variable Omega.

#### Returns

The value of capital omega.

Here is the call graph for this function:



### 7.27.3.5 omega()

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

Omega scalar const interface. Reference to state.time.

## 7.27.4 Friends And Related Function Documentation

### 7.27.4.1 operator<<

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

Output variables to plain scalar array.

### 7.27.4.2 operator>> [1/2]

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

Input(Initialize) variables with istream.

### 7.27.4.3 operator>> [2/2]

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

Input variables with plain scalar array.

## 7.27.5 Member Data Documentation

## 7.27.5.1 activeScalar

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

## 7.27.5.2 bindE\_

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

## 7.27.5.3 omega\_

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

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

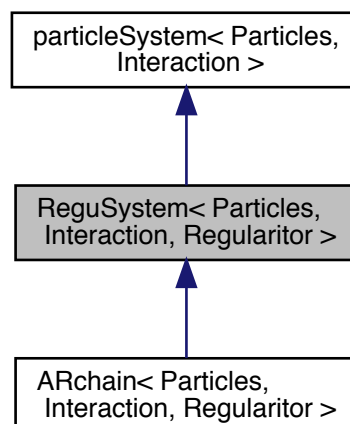
- [particleSystem/reguSystem/regularState.h](#)

## 7.28 ReguSystem&lt; Particles, Interaction, Regularitor &gt; Class Template Reference

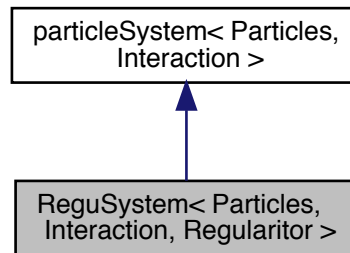
Regularized particle System.

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

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



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



## Public Types

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

## Public Member Functions

- `Scalar` & `omega` ()  
*Omega interface. Reference to `partc.omega`.*
- `Scalar` & `bindE` ()  
*Bindine energy interface. Reference to `partc.bindE`.*
- void `drift` (`Scalar` `stepSize`)  
*Advance position one step with current velocity.*
- void `kick` (`Scalar` `stepSize`)  
*Advance velocity one step with current acceleration.*
- `Scalar` `timeScale` (`Scalar` `scale`)  
*Interface to rescale the time.*

## Public Attributes

- Interaction `act`  
*Interaction class.*
- Particles `partc`  
*Particle class.*

## Additional Inherited Members

### 7.28.1 Detailed Description

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

Regularized particle System.

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

### 7.28.2 Member Typedef Documentation

#### 7.28.2.1 ActiveScalarArray

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

#### 7.28.2.2 Base

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

#### 7.28.2.3 Scalar

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

#### 7.28.2.4 type

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

### 7.28.2.5 Vector

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

### 7.28.2.6 VectorArray

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

## 7.28.3 Member Function Documentation

### 7.28.3.1 bindE()

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

Bindine energy interface. Reference to partc.bindE.

### 7.28.3.2 drift()

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

Advance position one step with current velocity.

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

#### Parameters

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

### 7.28.3.3 kick()

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

Advance velocity one step with current acceleration.

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

#### Parameters

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

#### 7.28.3.4 omega()

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

Omega interface. Reference to partc.omega.

#### 7.28.3.5 timeScale()

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

Interface to rescale the time.

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

#### Returns

The phsyical time.

### 7.28.4 Member Data Documentation

#### 7.28.4.1 act

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

Interaction class.

#### 7.28.4.2 `partc`

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

Particle class.

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

- `particleSystem/reguSystem/reguSystem.h`

## 7.29 `symplectic10th< ParticSys >` Class Template Reference

Eighth order symplectic integrator.

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

### Public Types

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

### Public Member Functions

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

### Static Public Attributes

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

#### 7.29.1 Detailed Description

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

Eighth order symplectic integrator.

#### 7.29.2 Member Typedef Documentation



### 7.29.2.1 Scalar

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

### 7.29.2.2 type

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

## 7.29.3 Member Function Documentation

### 7.29.3.1 integrate()

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

Interface to integrate particle system.

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

#### Parameters

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

## 7.29.4 Member Data Documentation

### 7.29.4.1 order

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

Order of the integrator.

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

- integrator/symplectic/[symplectic10th.h](#)

## 7.30 symplectic2th< ParticSys > Class Template Reference

Second order symplectic integrator.

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

### Public Types

- using [type](#) = typename ParticSys::type
- using [Scalar](#) = typename type::Scalar

### Public Member Functions

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

### Static Public Attributes

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

#### 7.30.1 Detailed Description

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

Second order symplectic integrator.

#### 7.30.2 Member Typedef Documentation

##### 7.30.2.1 Scalar

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

##### 7.30.2.2 type

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

### 7.30.3 Member Function Documentation

#### 7.30.3.1 integrate()

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

Interface to integrate particle system.

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

##### Parameters

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

### 7.30.4 Member Data Documentation

#### 7.30.4.1 order

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

Order of the integrator.

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

- [integrator/symplectic/symplectic2th.h](#)

## 7.31 symplectic4th< ParticSys > Class Template Reference

Fourth order symplectic integrator.

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

### Public Types

- using [type](#) = typename ParticSys::type
- using [Scalar](#) = typename type::Scalar

## Public Member Functions

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

## Static Public Attributes

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

### 7.31.1 Detailed Description

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

Fourth order symplectic integrator.

### 7.31.2 Member Typedef Documentation

#### 7.31.2.1 `Scalar`

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

#### 7.31.2.2 `type`

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

### 7.31.3 Member Function Documentation

#### 7.31.3.1 `integrate()`

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

Interface to integrate particle system.

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

## Parameters

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

## 7.31.4 Member Data Documentation

## 7.31.4.1 order

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

Order of the integrator.

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

- [integrator/symplectic/symplectic4th.h](#)

## 7.32 symplectic6th&lt; ParticSys &gt; Class Template Reference

Sixth order symplectic integrator.

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

## Public Types

- using [type](#) = typename ParticSys::type
- using [Scalar](#) = typename type::Scalar

## Public Member Functions

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

## Static Public Attributes

- static const int [order](#) {6}  
*Order of the integrator.*

### 7.32.1 Detailed Description

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

Sixth order symplectic integrator.

### 7.32.2 Member Typedef Documentation

#### 7.32.2.1 Scalar

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

#### 7.32.2.2 type

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

### 7.32.3 Member Function Documentation

#### 7.32.3.1 integrate()

```
template<typename ParticSys >
void symplectic6th< ParticSys >::integrate (
    ParticSys & particles,
    Scalar stepLength )
```

Interface to integrate particle system.

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

#### Parameters

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

### 7.32.4 Member Data Documentation

#### 7.32.4.1 order

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

Order of the integrator.

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

- [integrator/symplectic/symplectic6th.h](#)

## 7.33 symplectic8th< ParticSys > Class Template Reference

Eighth order symplectic integrator.

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

### Public Types

- using [type](#) = typename ParticSys::type
- using [Scalar](#) = typename type::Scalar

### Public Member Functions

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

### Static Public Attributes

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

### 7.33.1 Detailed Description

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

Eighth order symplectic integrator.

## 7.33.2 Member Typedef Documentation

### 7.33.2.1 Scalar

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

### 7.33.2.2 type

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

## 7.33.3 Member Function Documentation

### 7.33.3.1 integrate()

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

Interface to integrate particle system.

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

#### Parameters

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

## 7.33.4 Member Data Documentation

### 7.33.4.1 order

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



Order of the integrator.

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

- [integrator/symplectic/symplectic8th.h](#)

## 7.34 TTL< Particles > Class Template Reference

Time Transform Leapfrog algorithmatic regularization interface.

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

### Public Types

- using [type](#) = typename Particles::type
- using [Scalar](#) = typename type::Scalar

### Public Member Functions

- [Scalar getPhysicalPosTime](#) (Particles &partc, [Scalar](#) stepSize)  
*Calculate the physical time for position advance from integration step size.*
- [Scalar getPhysicalVelTime](#) (Particles &partc, [Scalar](#) stepSize)  
*Calculate the physical time for velocity advance from integration step size.*

#### 7.34.1 Detailed Description

```
template<typename Particles>
class TTL< Particles >
```

Time Transform Leapfrog algorithmatic regularization interface.

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

#### 7.34.2 Member Typedef Documentation

##### 7.34.2.1 Scalar

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

### 7.34.2.2 type

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

## 7.34.3 Member Function Documentation

### 7.34.3.1 getPhysicalPosTime()

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

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

#### Parameters

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

### 7.34.3.2 getPhysicalVelTime()

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

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

#### Parameters

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

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

- `particleSystem/reguSystem/regularization.h`

## 7.35 vec3< T > Struct Template Reference

Self 3D vector class.

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

### Public Types

- using `value_type` = T

### Public Member Functions

- `vec3` ()
- `vec3` (T vx, T vy, T vz)
- `vec3` (const `vec3` &v)
- `vec3 operator+` (const `vec3` &v) const  
*Addition by wise.*
- `vec3 operator-` (const `vec3` &v) const  
*Subtraction by wise.*
- `vec3 operator/` (const `vec3` &v) const  
*Divition by wise.*
- `vec3 operator+` (const T c) const  
*Add scalar by wise.*
- `vec3 operator-` (const T c) const  
*Subtract scalar by wise.*
- `vec3 operator*` (const T c) const  
*Multiply scalar by wise.*
- `vec3 operator/` (const T c) const  
*Divide scalar by wise.*
- `vec3 operator-` () const  
*Opposite vector.*
- `vec3 operator^` (const `vec3` &v) const  
*Cross product.*
- `vec3 abs` () const  
*Absolute value by wise.*
- const `vec3` & `operator+=` (const `vec3` &v)
- const `vec3` & `operator-=` (const `vec3` &v)
- const `vec3` & `operator/=` (const `vec3` &v)
- const `vec3` & `operator+=` (const T c)
- const `vec3` & `operator-=` (const T c)
- const `vec3` & `operator*=` (const T c)
- const `vec3` & `operator/=` (const T c)
- const `vec3` & `operator=` (const `vec3` &v)
- T `operator*` (const `vec3` &v) const  
*Inner product.*
- T `norm` () const  
*Calculate the norm.*
- T `normSquare` () const  
*Calcualte the square of the norm.*
- T `reNorm` () const  
*Calculate the inverse of the norm.*
- void `setZero` ()

## Public Attributes

- [T x](#)
- [T y](#)
- [T z](#)

## Friends

- [vec3 operator+](#) (const T c, const [vec3](#) &v)
- [vec3 operator-](#) (const T c, const [vec3](#) &v)
- [vec3 operator\\*](#) (const T c, const [vec3](#) &v)
- std::ostream & [operator<<](#) (std::ostream &output, const [vec3](#) &v)  
*Output to ostream.*
- std::istream & [operator>>](#) (std::istream &input, [vec3](#) &v)  
*Input from istream.*

### 7.35.1 Detailed Description

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

Self 3D vector class.

### 7.35.2 Member Typedef Documentation

#### 7.35.2.1 value\_type

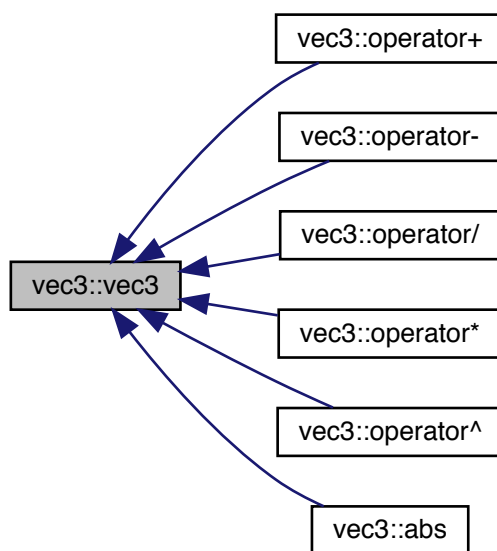
```
template<typename T>  
using vec3< T >::value_type = T
```

### 7.35.3 Constructor & Destructor Documentation

## 7.35.3.1 vec3() [1/3]

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

Here is the caller graph for this function:



## 7.35.3.2 vec3() [2/3]

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

## 7.35.3.3 vec3() [3/3]

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

## 7.35.4 Member Function Documentation

#### 7.35.4.1 abs()

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

Absolute value by wise.

Here is the call graph for this function:



#### 7.35.4.2 norm()

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

Calculate the norm.

#### 7.35.4.3 normSquare()

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

Calcualte the square of the norm.

#### 7.35.4.4 operator\*() [1/2]

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

Multiply scalar by wise.

Here is the call graph for this function:



**7.35.4.5 operator\*()** [2/2]

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

Inner product.

**7.35.4.6 operator\*=( )**

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

**7.35.4.7 operator+()** [1/2]

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

Addition by wise.

Here is the call graph for this function:

**7.35.4.8 operator+()** [2/2]

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

Add scalar by wise.

Here is the call graph for this function:



**7.35.4.9 operator+=( )** [1/2]

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

**7.35.4.10 operator+=( )** [2/2]

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

**7.35.4.11 operator-( )** [1/3]

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

Subtraction by wise.

Here is the call graph for this function:

**7.35.4.12 operator-( )** [2/3]

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

Subtract scalar by wise.

Here is the call graph for this function:





**7.35.4.13 operator-()** [3/3]

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

Opposite vector.

Here is the call graph for this function:

**7.35.4.14 operator-=()** [1/2]

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

**7.35.4.15 operator-=()** [2/2]

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

**7.35.4.16 operator/()** [1/2]

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

Division by wise.

Here is the call graph for this function:



#### 7.35.4.17 operator/() [2/2]

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

Divide scalar by wise.

Here is the call graph for this function:



#### 7.35.4.18 operator/=() [1/2]

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

#### 7.35.4.19 operator/=() [2/2]

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

#### 7.35.4.20 operator=()

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

## 7.35.4.21 operator^()

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

Cross product.

Here is the call graph for this function:

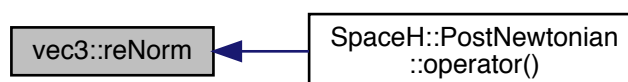


## 7.35.4.22 reNorm()

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

Calculate the inverse of the norm.

Here is the caller graph for this function:



## 7.35.4.23 setZero()

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

## 7.35.5 Friends And Related Function Documentation

#### 7.35.5.1 operator\*

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

#### 7.35.5.2 operator+

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

#### 7.35.5.3 operator-

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

#### 7.35.5.4 operator<<

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

Output to ostream.

#### 7.35.5.5 operator>>

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

Input from istream.

### 7.35.6 Member Data Documentation

## 7.35.6.1 x

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

## 7.35.6.2 y

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

## 7.35.6.3 z

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

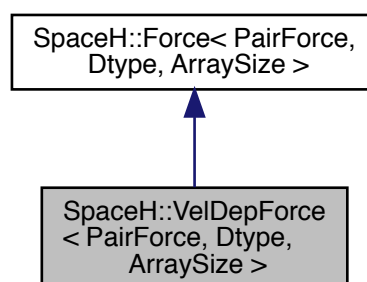
The documentation for this struct was generated from the following file:

- [vector3.h](#)

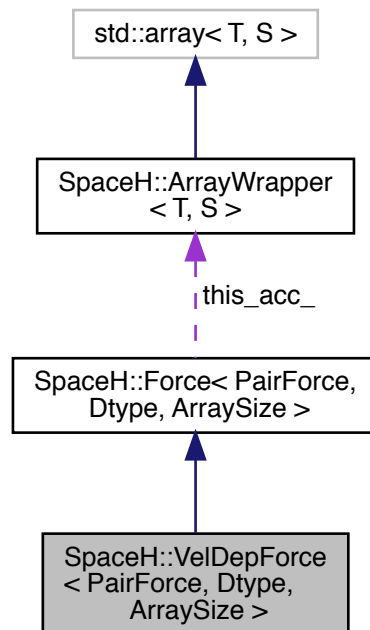
## 7.36 SpaceH::VelDepForce< PairForce, Dtype, ArraySize > Struct Template Reference

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

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



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



## Public Types

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

## Public Member Functions

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

## Public Attributes

- PairForce `force_`
- `VectorArray` `this_acc_`

## Static Public Attributes

- static constexpr bool `isVelDep` {true}

## Additional Inherited Members

### 7.36.1 Member Typedef Documentation

#### 7.36.1.1 Base

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

#### 7.36.1.2 IndexArray

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

#### 7.36.1.3 Scalar

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

#### 7.36.1.4 ScalarArray

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

#### 7.36.1.5 type

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

#### 7.36.1.6 Vector

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

### 7.36.1.7 VectorArray

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

## 7.36.2 Member Function Documentation

### 7.36.2.1 calcuAcc() [1/2]

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

### 7.36.2.2 calcuAcc() [2/2]

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

## 7.36.3 Member Data Documentation

### 7.36.3.1 force\_

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

### 7.36.3.2 isVelDep

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



## 7.36.3.3 this\_acc\_

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

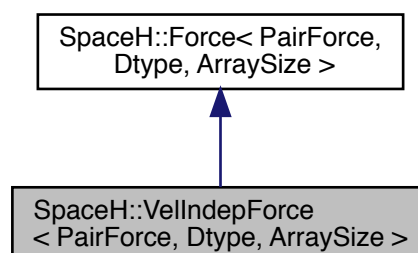
The documentation for this struct was generated from the following file:

- [interaction/forces.h](#)

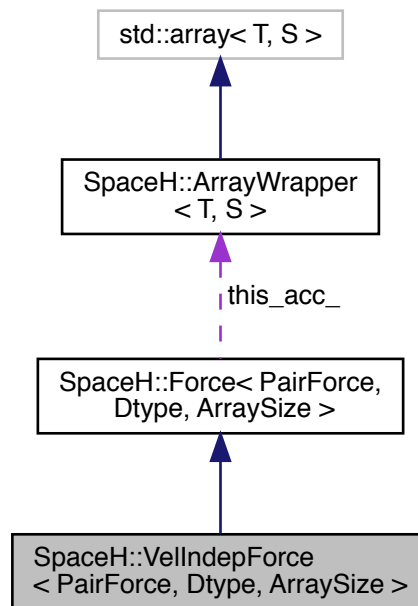
## 7.37 SpaceH::VelIndepForce&lt; PairForce, Dtype, ArraySize &gt; Struct Template Reference

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

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



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



## Public Types

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

## Public Member Functions

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

## Public Attributes

- PairForce `force_`
- `VectorArray` `this_acc_`

## Static Public Attributes

- static constexpr bool `isVelDep` {false}

## Additional Inherited Members

### 7.37.1 Member Typedef Documentation

#### 7.37.1.1 Base

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

#### 7.37.1.2 IndexArray

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

#### 7.37.1.3 Scalar

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

#### 7.37.1.4 ScalarArray

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

#### 7.37.1.5 type

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

#### 7.37.1.6 Vector

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

### 7.37.1.7 VectorArray

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

## 7.37.2 Member Function Documentation

### 7.37.2.1 calcuAcc() [1/2]

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

### 7.37.2.2 calcuAcc() [2/2]

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

## 7.37.3 Member Data Documentation

### 7.37.3.1 force\_

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

### 7.37.3.2 isVelDep

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

### 7.37.3.3 this\_acc\_

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

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

- [interaction/forces.h](#)

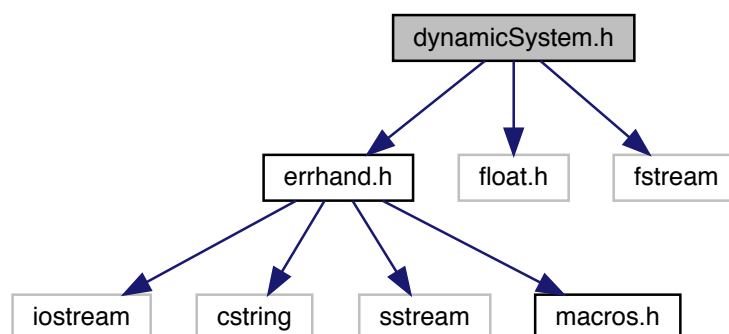
## Chapter 8

# File Documentation

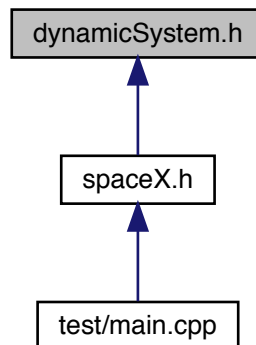
### 8.1 dynamicSystem.h File Reference

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

Include dependency graph for dynamicSystem.h:



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



## Classes

- class `dynamicSystem< ParticSys, ODEiterator >`  
*A wrapper to make particle system, integrator and ODE iterator work together.*

## Typedefs

- `template<typename ParticSys , template< typename > class Integrator, template< typename, typename > class ODEiterator>`  
`using spaceX = dynamicSystem< ParticSys, ODEiterator< ParticSys, Integrator< ParticSys > >>`  
*Alias of template name, linking the particle system, integrator and ODE iterator.*

### 8.1.1 Typedef Documentation

#### 8.1.1.1 spaceX

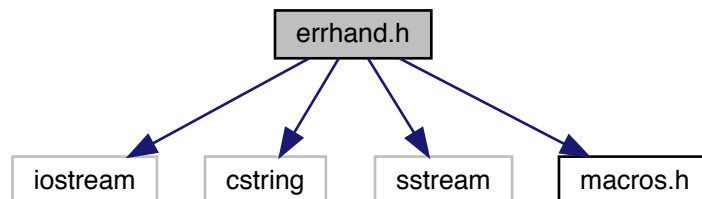
```

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

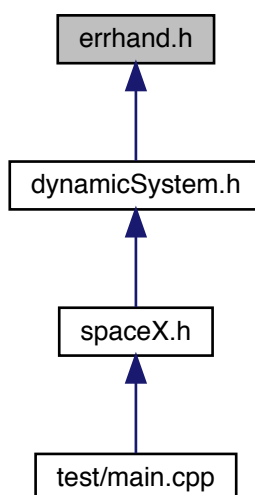
Alias of template name, linking the particle system, integrator and ODE iterator.

## 8.2 errhand.h File Reference

```
#include <iostream>
#include <cstring>
#include <sstream>
#include "macros.h"
Include dependency graph for errhand.h:
```



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



### Classes

- class [errhand](#)

### Namespaces

- [NOTICE](#)

## Macros

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

## Functions

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

## Variables

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

### 8.2.1 Macro Definition Documentation

#### 8.2.1.1 ANSI\_COLOR\_BLUE

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

#### 8.2.1.2 ANSI\_COLOR\_CYAN

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

#### 8.2.1.3 ANSI\_COLOR\_GREEN

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



#### 8.2.1.4 ANSI\_COLOR\_MAGENTA

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

#### 8.2.1.5 ANSI\_COLOR\_RED

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

#### 8.2.1.6 ANSI\_COLOR\_RESET

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

#### 8.2.1.7 ANSI\_COLOR\_YELLOW

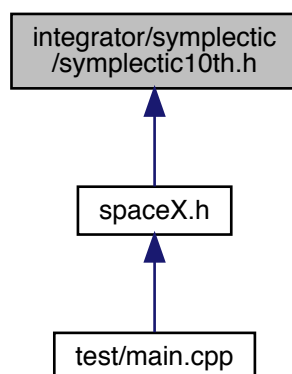
```
#define ANSI_COLOR_YELLOW "\x1b[33m"
```

#### 8.2.1.8 NEWLINE

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

## 8.3 integrator/symplectic/symplectic10th.h File Reference

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



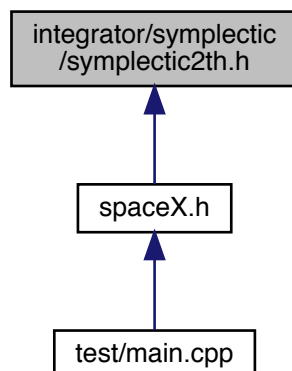
## Classes

- class [symplectic10th](#)< [ParticSys](#) >

*Eighth order symplectic integrator.*

## 8.4 integrator/symplectic/symplectic2th.h File Reference

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



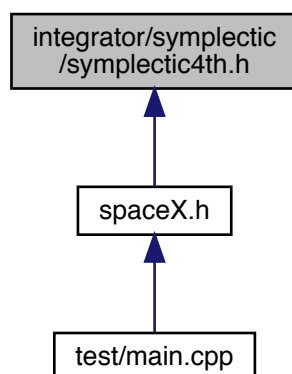
## Classes

- class [symplectic2th](#)< [ParticSys](#) >

*Second order symplectic integrator.*

## 8.5 integrator/symplectic/symplectic4th.h File Reference

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

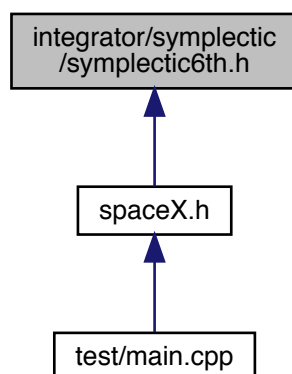


### Classes

- class [symplectic4th](#)< [ParticSys](#) >  
*Fourth order symplectic integrator.*

## 8.6 integrator/symplectic/symplectic6th.h File Reference

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

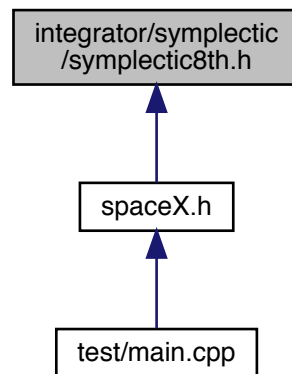


## Classes

- class [symplectic6th](#)< [ParticSys](#) >  
*Sixth order symplectic integrator.*

## 8.7 integrator/symplectic/symplectic8th.h File Reference

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



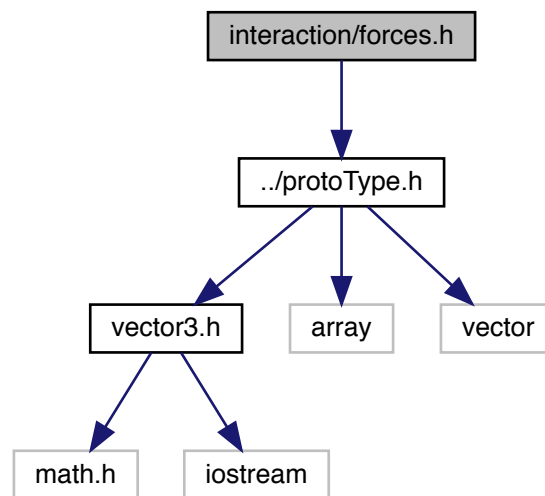
## Classes

- class [symplectic8th](#)< [ParticSys](#) >  
*Eighth order symplectic integrator.*

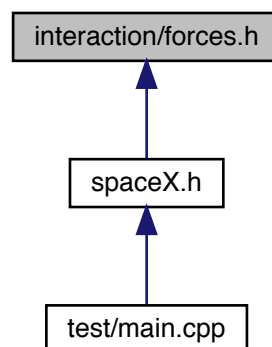
## 8.8 interaction/forces.h File Reference

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

Include dependency graph for forces.h:



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



## Classes

- struct [SpaceH::EmptyForce< Dtype, ArraySize >](#)
- struct [SpaceH::Force< PairForce, Dtype, ArraySize >](#)
- struct [SpaceH::VelIndepForce< PairForce, Dtype, ArraySize >](#)
- struct [SpaceH::ExtVelIndepForce< PairForce, Dtype, ArraySize >](#)
- struct [SpaceH::VelDepForce< PairForce, Dtype, ArraySize >](#)
- struct [SpaceH::ExtVelDepForce< PairForce, Dtype, ArraySize >](#)

- struct [SpaceH::NewtonForce< Dtype, ArraySize >](#)
- class [SpaceH::PostNewtonian< Scalar >](#)

*Post newtonian pair interaction functor(c++ std11)*

## Namespaces

- [SpaceH](#)

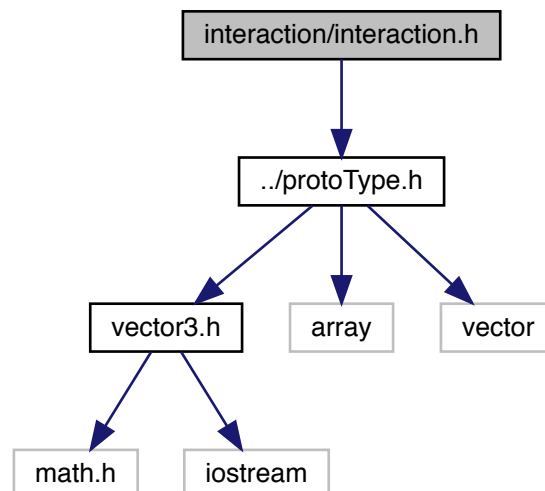
## Variables

- constexpr double [SpaceH::INV\\_C](#) = 1 / C
- constexpr double [SpaceH::INV\\_C2](#) = INV\_C \* INV\_C
- constexpr double [SpaceH::INV\\_C3](#) = INV\_C2 \* INV\_C
- constexpr double [SpaceH::INV\\_C4](#) = INV\_C3 \* INV\_C
- constexpr double [SpaceH::INV\\_C5](#) = INV\_C4 \* INV\_C

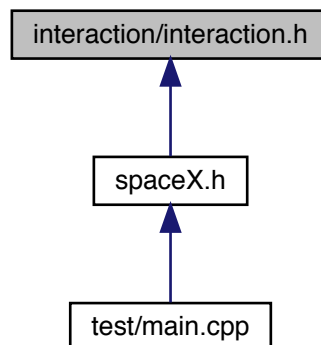
## 8.9 interaction/interaction.h File Reference

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

Include dependency graph for interaction.h:



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



### Classes

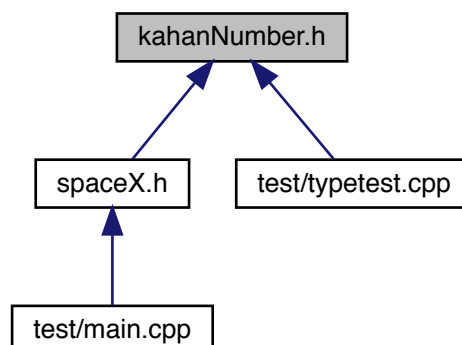
- class [SpaceH::Interaction](#)< [VelIndep](#), [VelDep](#), [ExtVelIndep](#), [ExtVelDep](#) >

### Namespaces

- [SpaceH](#)

## 8.10 kahanNumber.h File Reference

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



## Classes

- struct [SpaceH::kahan< T >](#)  
*Kahan number.*

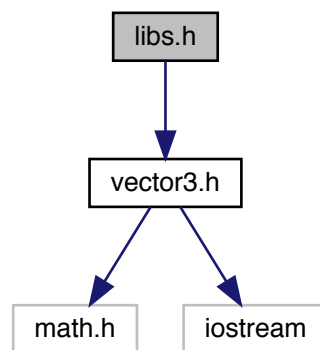
## Namespaces

- [SpaceH](#)

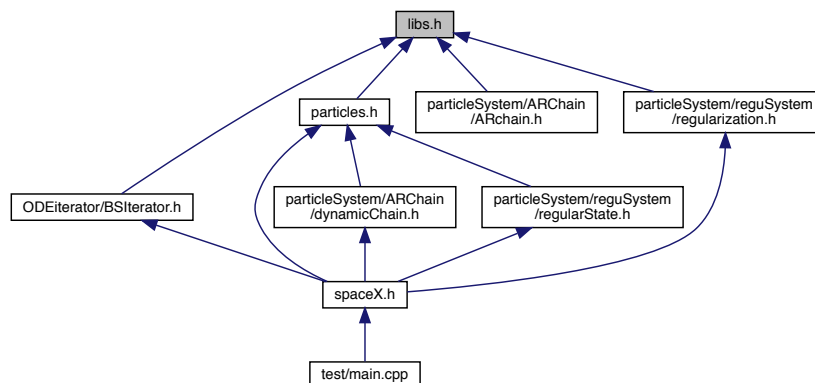
## 8.11 libs.h File Reference

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

Include dependency graph for libs.h:



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





## Namespaces

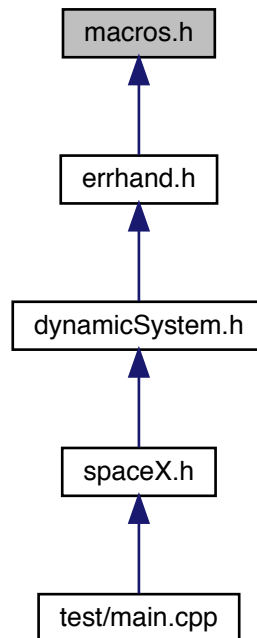
- [SpaceH](#)

## Functions

- `template<typename T1 , typename T2 >`  
`const T2 SpaceH::min (const T1 &x, const T2 &y)`  
*Self [min\(\)](#)*
- `template<typename T1 , typename T2 >`  
`const T2 SpaceH::max (const T1 &x, const T2 &y)`  
*Self [max\(\)](#)*
- `template<class T >`  
`const T SpaceH::abs (const T &x)`  
*Self [abs\(\)](#)*
- `template<typename Scalar1 , typename Scalar2 >`  
`void SpaceH::advanceScalar (Scalar1 &var, Scalar2 increase)`  
*Self [swap\(\)](#)*
- `template<typename Scalar , typename Vector1 , typename Vector2 >`  
`void SpaceH::advanceVector (Vector1 &var, const Vector2 &increase, Scalar stepSize)`
- `template<typename ScalarArray , typename VectorArray >`  
`void SpaceH::moveToCMCoord (const ScalarArray &mass, VectorArray &phyVar)`  
*Move variables to central mass coordinates.*
- `template<typename Scalar , size_t N>`  
`double SpaceH::getKineticEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &vel)`  
*Calculate the kinetic energy of particles.*
- `template<typename Scalar , size_t N>`  
`double SpaceH::getPotentialEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &pos)`  
*Calculate the potential energy of particles.*
- `template<typename Scalar , size_t N>`  
`double SpaceH::getTotalEnergy (const std::array< Scalar, N > &mass, const std::array< vec3< Scalar >, N > &pos, const std::array< vec3< Scalar >, N > &vel)`  
*Calculate the total(potential + kinetic) energy of particles.*
- `template<typename T >`  
`void SpaceH::print (T &var)`  
*print an array. Used for debug*

## 8.12 macros.h File Reference

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



### Variables

- constexpr double **PI** = 3.14159265358979323
- constexpr double **AU** = (**PI** / 648000)
- constexpr double **PC** = 1
- constexpr double **M\_SOLAR** = 1
- constexpr double **M\_JUPITER** = 0.9547919E-3
- constexpr double **R\_SOLAR** = 2.25461E-8
- constexpr double **YEAR** = 6.694685210039141E-08
- constexpr double **DAY** = **YEAR** / 365.25636042
- constexpr double **G** = 1
- constexpr double **V\_UNIT** = 6.54589713446219E-2
- constexpr double **C** = 299792.458 / **V\_UNIT**
- constexpr double **KM** = 3.2407557442395564e-14

### 8.12.1 Variable Documentation

### 8.12.1.1 AU

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

### 8.12.1.2 C

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

### 8.12.1.3 DAY

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

### 8.12.1.4 G

```
constexpr double G = 1
```

### 8.12.1.5 KM

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

### 8.12.1.6 M\_JUPITER

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

### 8.12.1.7 M\_SOLAR

```
constexpr double M_SOLAR = 1
```

### 8.12.1.8 PC

```
constexpr double PC = 1
```

#### 8.12.1.9 PI

```
constexpr double PI = 3.14159265358979323
```

#### 8.12.1.10 R\_SOLAR

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

#### 8.12.1.11 V\_UNIT

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

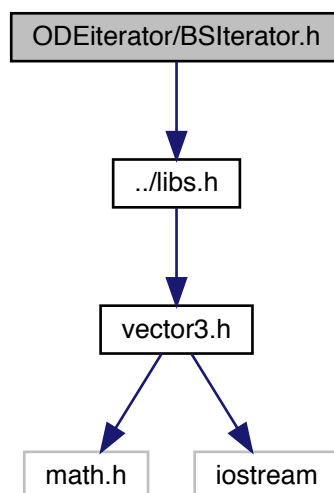
#### 8.12.1.12 YEAR

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

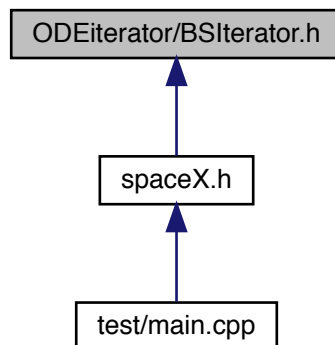
### 8.13 ODEiterator/BSIterator.h File Reference

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

Include dependency graph for BSIterator.h:



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

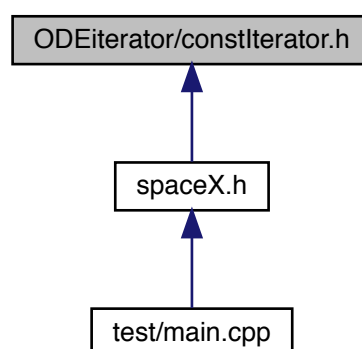


### Classes

- class [BSliterator](#)< [ParticSys](#), [Integrator](#) >  
*Bulirsch-Stoer extrapolation algorithm.*

## 8.14 ODEiterator/constliterator.h File Reference

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



### Classes

- class [constliterator](#)< [ParticSys](#), [Integrator](#) >  
*Most common iterator.*

## 8.15 ODEiterator/dichotomy.h File Reference

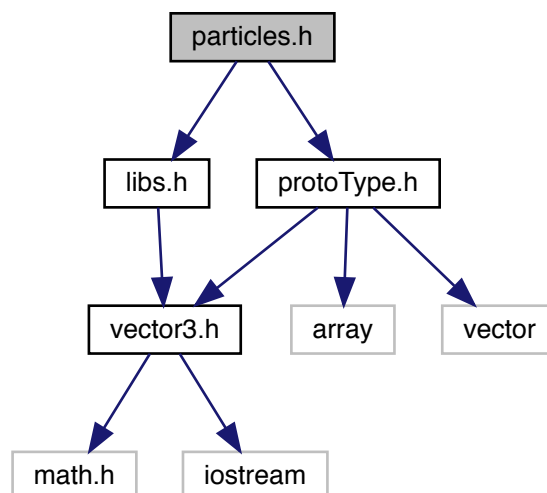
### Classes

- class [dicholterator](#)< [ParticSys](#), [Integrator](#) >

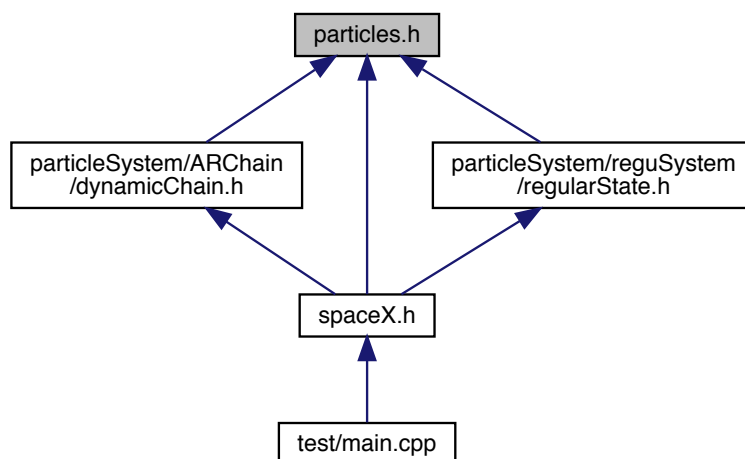
*Dichotomy iterator.*

## 8.16 particles.h File Reference

```
#include "protoType.h"  
#include "libs.h"  
Include dependency graph for particles.h:
```



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



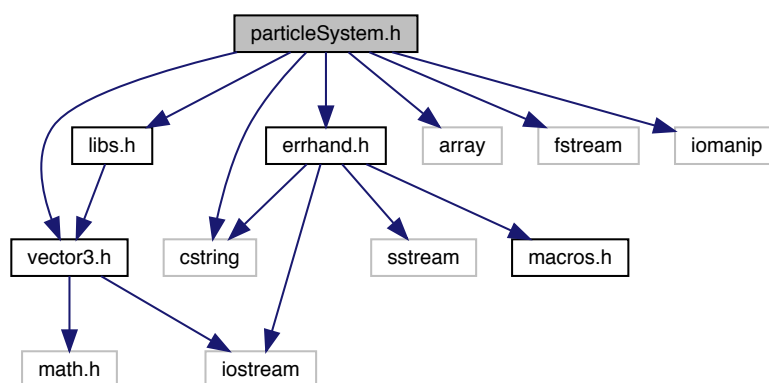
## Classes

- class `particles< Dtype, ArraySize >`  
Class of dynamical variable.

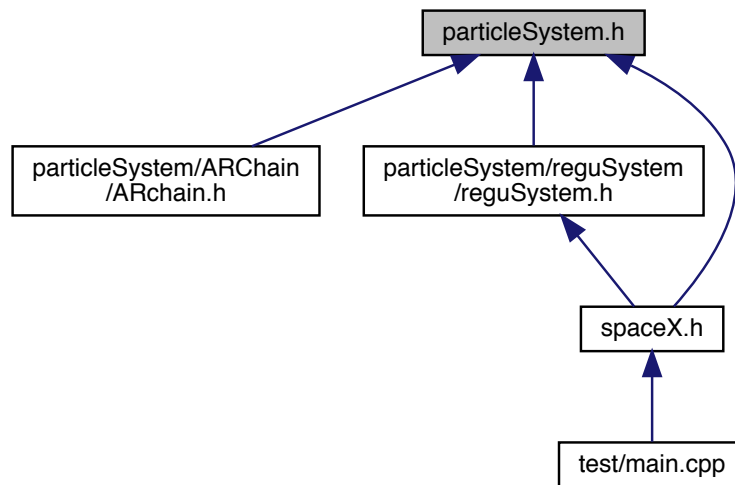
## 8.17 particleSystem.h File Reference

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

Include dependency graph for particleSystem.h:



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



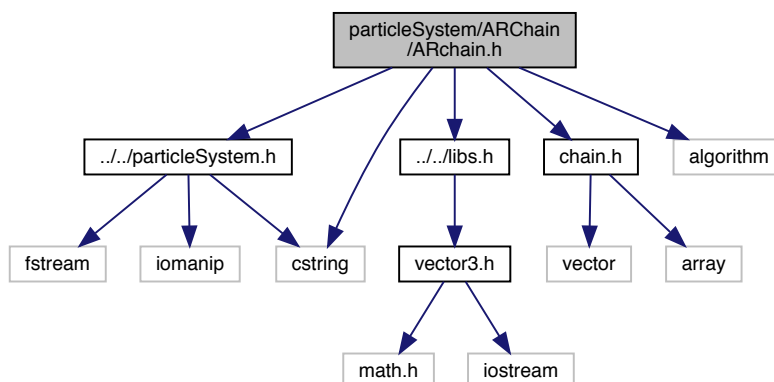
## Classes

- class [particleSystem](#) < [Particles](#), [Interaction](#) >  
Base class of particle System.

## 8.18 particleSystem/ARChain/ARchain.h File Reference

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

Include dependency graph for ARchain.h:



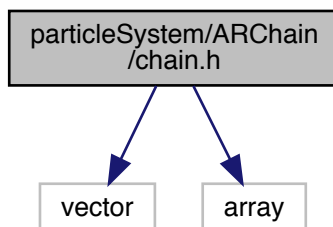


## Classes

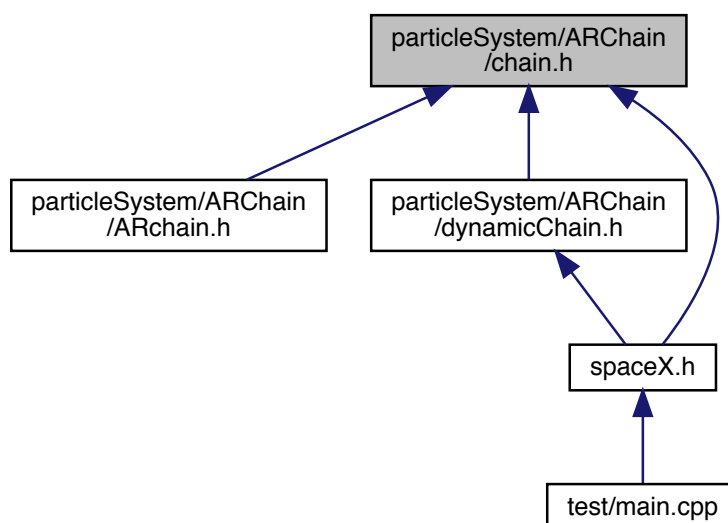
- class [ARchain](#)< [Particles](#), [Interaction](#), [Regularitor](#) >  
*Algorithmic Regularization chain System.*

## 8.19 particleSystem/ARChain/chain.h File Reference

```
#include <vector>
#include <array>
Include dependency graph for chain.h:
```



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



## Classes

- struct [SpaceH::chain::Node< Scalar >](#)

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

## Namespaces

- [SpaceH](#)
- [SpaceH::chain](#)

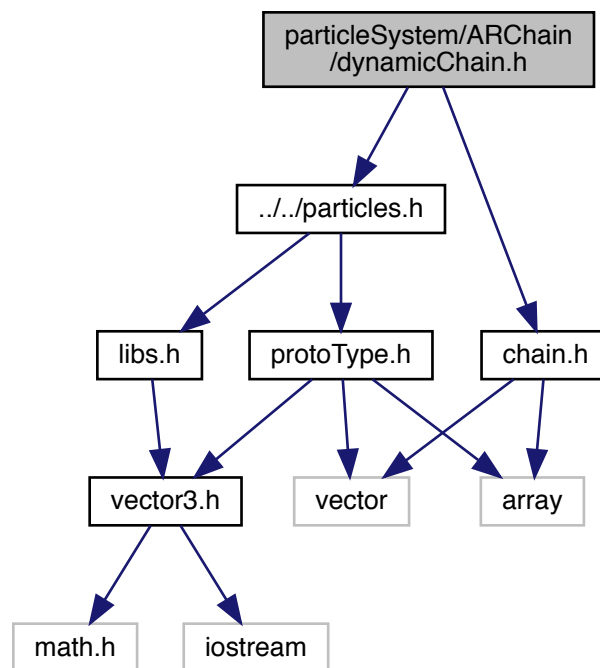
## Functions

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

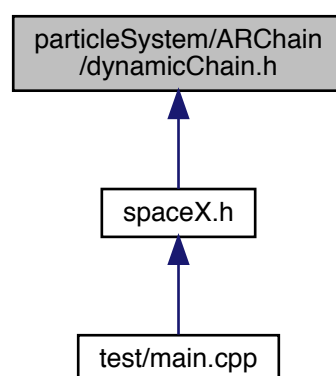
## 8.20 particleSystem/ARChain/dynamicChain.h File Reference

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

Include dependency graph for dynamicChain.h:



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



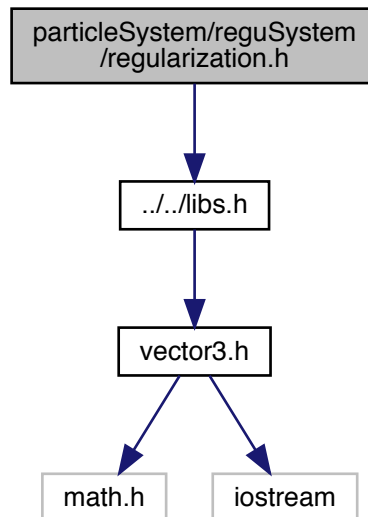
## Classes

- class [ChainParticles< Dtype, ArraySize >](#)  
*Class of dynamical variable.*

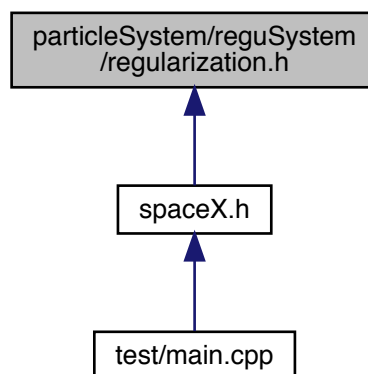
## 8.21 particleSystem/reguSystem/regularization.h File Reference

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

Include dependency graph for regularization.h:



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



### Classes

- class [logH< Particles >](#)

[logH](#) *extention algorithmatic regularization interface*

- class [TTL< Particles >](#)

*Time Transform Leapfrog algorithmatic regularization interface.*

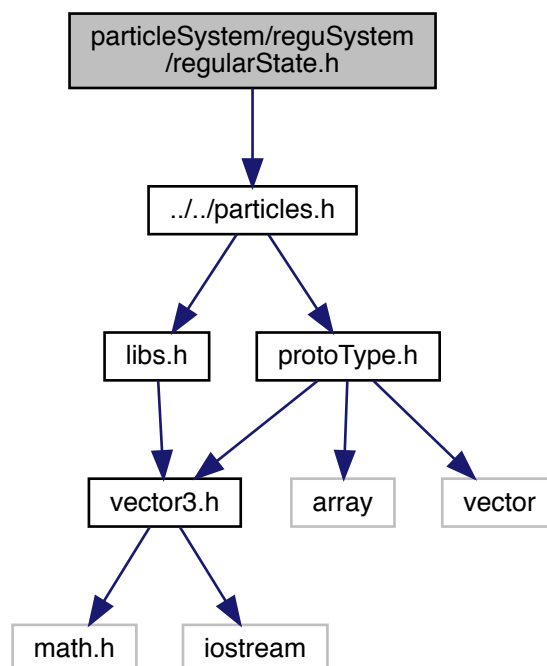
- class [NoRegu< Particles >](#)

*Ordinary algorithmatic regularization interface.*

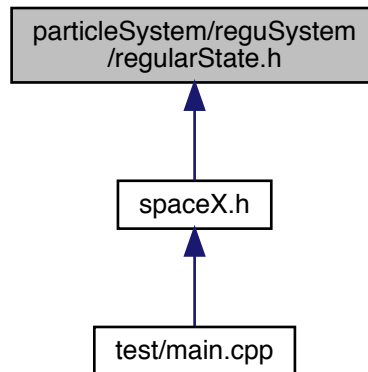
## 8.22 particleSystem/reguSystem/regularState.h File Reference

```
#include "../particles.h"
```

Include dependency graph for regularState.h:



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



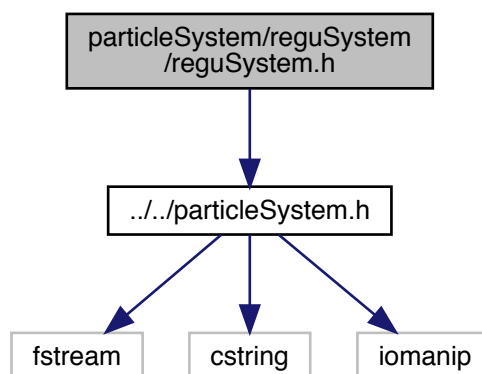
## Classes

- class [ReguParticles](#)< Dtype, ArraySize >  
*Class of dynamical system with regularization variables.*

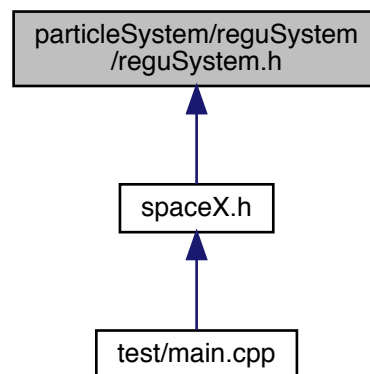
## 8.23 particleSystem/reguSystem/reguSystem.h File Reference

```
#include "../..//particleSystem.h"
```

Include dependency graph for reguSystem.h:



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



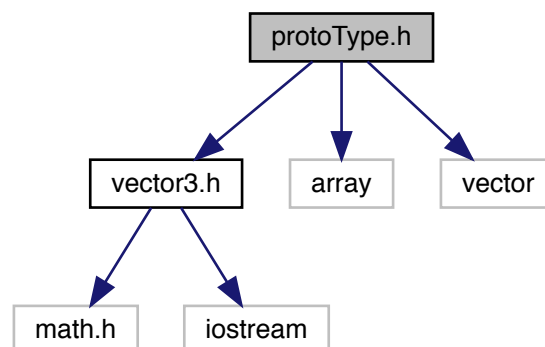
## Classes

- class [ReguSystem](#)< [Particles](#), [Interaction](#), [Regularitor](#) >  
*Regularized particle System.*

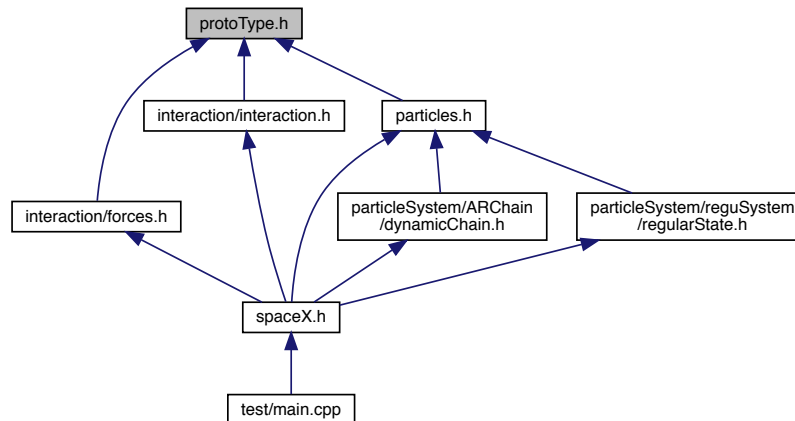
## 8.24 protoType.h File Reference

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

Include dependency graph for protoType.h:



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



## Classes

- struct [SpaceH::ArrayWrapper< T, S >](#)
- struct [SpaceH::ArrayWrapper< T, DYNAMICAL >](#)
- struct [SpaceH::get\\_value\\_type< T >](#)
- struct [SpaceH::ProtoType< Dtype, Size >](#)

## Namespaces

- [SpaceH](#)

## Enumerations

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

## Variables

- constexpr size\_t [SpaceH::DYNAMICAL](#) = 0

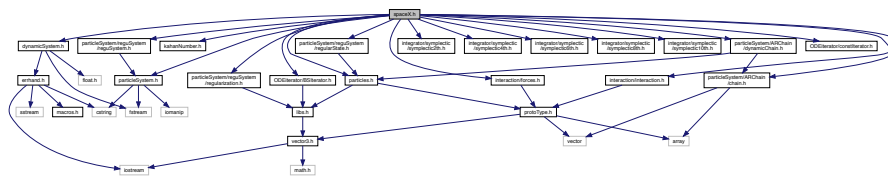


## 8.25 README.md File Reference

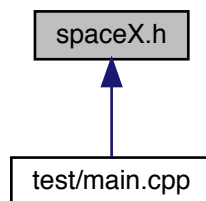
### 8.26 spaceX.h File Reference

```
#include "dynamicSystem.h"
#include "particles.h"
#include "kahanNumber.h"
#include "particleSystem.h"
#include "particleSystem/reguSystem/regularization.h"
#include "particleSystem/reguSystem/reguSystem.h"
#include "particleSystem/reguSystem/regularState.h"
#include "particleSystem/ARChain/chain.h"
#include "particleSystem/ARChain/dynamicChain.h"
#include "integrator/symplectic/symplectic2th.h"
#include "integrator/symplectic/symplectic4th.h"
#include "integrator/symplectic/symplectic6th.h"
#include "integrator/symplectic/symplectic8th.h"
#include "integrator/symplectic/symplectic10th.h"
#include "ODEiterator/BSIterator.h"
#include "ODEiterator/constIterator.h"
#include "interaction/interaction.h"
#include "interaction/forces.h"
```

Include dependency graph for spaceX.h:



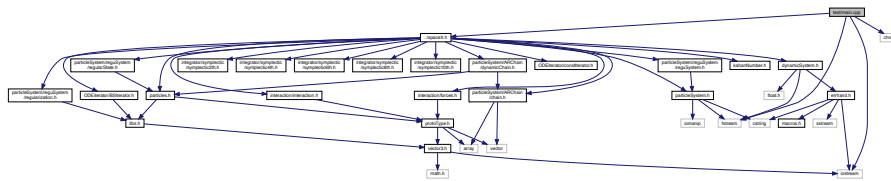
This graph shows which files directly or indirectly include this file:



### 8.27 test/main.cpp File Reference

```
#include "../spaceX.h"
#include <chrono>
```

```
#include <fstream>
#include <iostream>
Include dependency graph for main.cpp:
```



## Typedefs

- typedef std::chrono::time\_point< std::chrono::high\_resolution\_clock > [resolutionClock](#)
- using [scalar](#) = double

## Functions

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

## Variables

- const size\_t [N](#) = 3

## 8.27.1 Typedef Documentation

### 8.27.1.1 resolutionClock

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

### 8.27.1.2 scalar

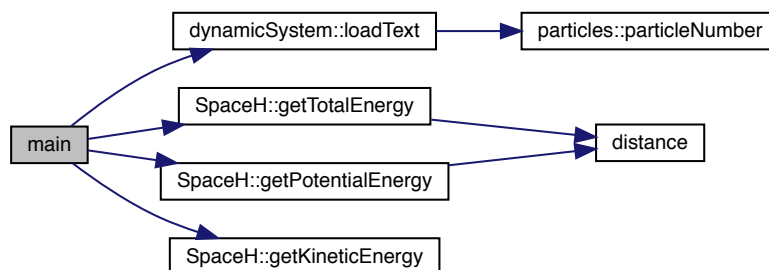
```
using scalar = double
```

## 8.27.2 Function Documentation

## 8.27.2.1 main()

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

Here is the call graph for this function:



## 8.27.3 Variable Documentation

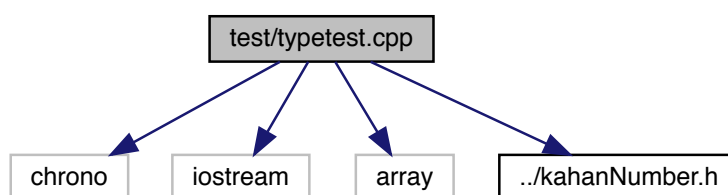
## 8.27.3.1 N

```
const size_t N = 3
```

## 8.28 test/typetest.cpp File Reference

```
#include <chrono>
#include <iostream>
#include <array>
#include "../kahanNumber.h"
```

Include dependency graph for `typetest.cpp`:



## Classes

- struct [dtype< T >](#)

## Typedefs

- typedef std::chrono::time\_point< std::chrono::high\_resolution\_clock > [resolutionClock](#)

## Functions

- int [main](#) ()

### 8.28.1 Typedef Documentation

#### 8.28.1.1 resolutionClock

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

### 8.28.2 Function Documentation

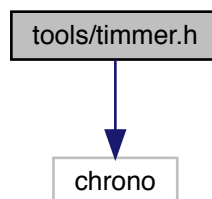
#### 8.28.2.1 main()

```
int main ( )
```

## 8.29 tools/timmer.h File Reference

```
#include <chrono>
```

Include dependency graph for timmer.h:



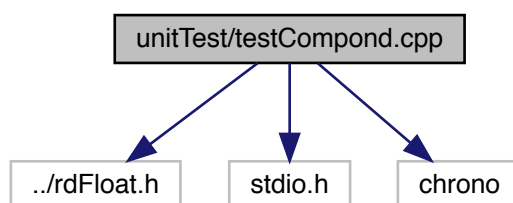
## Classes

- class [ProgressBar](#)

## 8.30 unitTest/testCompond.cpp File Reference

```
#include "../rdFloat.h"
#include <stdio.h>
#include <chrono>
```

Include dependency graph for testCompond.cpp:



## Typedefs

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

## Functions

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

### 8.30.1 Typedef Documentation

#### 8.30.1.1 resolutionClock

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

### 8.30.2 Function Documentation

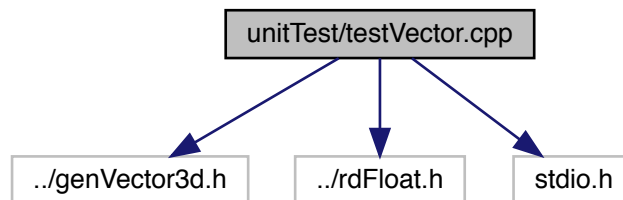
### 8.30.2.1 main()

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

## 8.31 unitTest/testVector.cpp File Reference

```
#include "../genVector3d.h"
#include "../rdFloat.h"
#include <stdio.h>
```

Include dependency graph for testVector.cpp:



### Functions

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

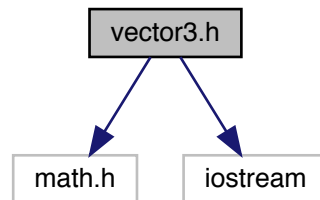
### 8.31.1 Function Documentation

#### 8.31.1.1 main()

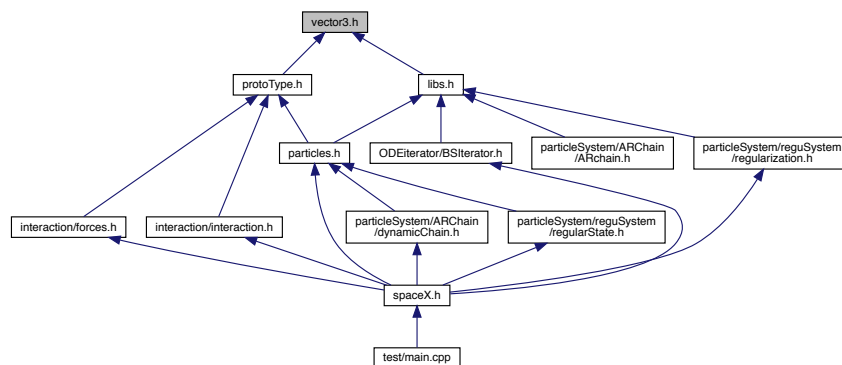
```
int main (
    int argc,
    char ** argv )
```

## 8.32 vector3.h File Reference

```
#include <math.h>
#include <iostream>
Include dependency graph for vector3.h:
```



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



### Classes

- struct [vec3< T >](#)  
Self 3D vector class.

### Typedefs

- typedef [vec3< double >](#) [vec3d](#)
- typedef [vec3< float >](#) [vec3f](#)
- typedef [vec3< int >](#) [vec3i](#)
- typedef [vec3< char >](#) [vec3c](#)

## Functions

- `template<typename T>`  
  `T distance (const vec3< T > &v1, const vec3< T > &v2)`  
    *Calculate the Euclid distance of two vectors.*

## 8.32.1 Typedef Documentation

### 8.32.1.1 vec3c

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

### 8.32.1.2 vec3d

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

### 8.32.1.3 vec3f

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

### 8.32.1.4 vec3i

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

## 8.32.2 Function Documentation

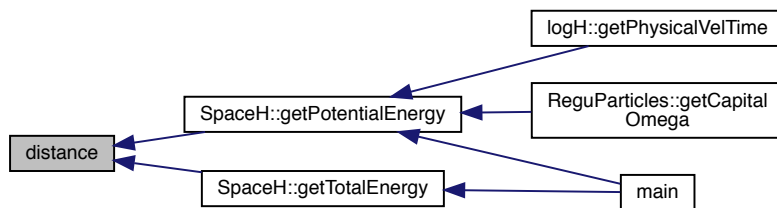


## 8.32.2.1 distance()

```
template<typename T >
T distance (
    const vec3< T > & v1,
    const vec3< T > & v2 ) [inline]
```

Calculate the Euclid distance of two vectors.

Here is the caller graph for this function:





# Index

- ~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 >, [97](#)

- 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
- INTEGRATOR\_TYPE
  - 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 EvolvedData::↵ 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 EvolvedData::↵ 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 EvolvedData::↵ 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 Evolved↔ 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 Evolved↔ 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](#)