

SMC Template Class 1.0

Adam M. Johansen

Generated by Doxygen 1.5.5

Mon Jul 14 16:34:16 2008

Contents

1	smctc – A Template Class Library for SMC Simulation	1
1.1	Summary	1
1.2	License	1
1.3	Using the Template Library	1
1.4	Examples	2
2	Namespace Index	3
2.1	Namespace List	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	smc Namespace Reference	9
5.2	std Namespace Reference	11
6	Class Documentation	13
6.1	smc::exception Class Reference	13
6.2	smc::gslrnginfo Class Reference	16
6.3	smc::history< Particle > Class Template Reference	19
6.4	smc::historyelement< Particle > Class Template Reference	25
6.5	smc::historyflags Class Reference	33
6.6	smc::moveset< Space > Class Template Reference	35
6.7	smc::particle< Space > Class Template Reference	42
6.8	smc::rng Class Reference	48
6.9	smc::sampler< Space > Class Template Reference	56

7	File Documentation	69
7.1	include/history.hh File Reference	69
7.2	include/moveset.hh File Reference	71
7.3	include/particle.hh File Reference	73
7.4	include/rng.hh File Reference	75
7.5	include/sampler.hh File Reference	77
7.6	include/smc-exception.hh File Reference	80
7.7	include/smctc.hh File Reference	83
7.8	src/history.cc File Reference	85
7.9	src/rng.cc File Reference	86
7.10	src/smc-exception.cc File Reference	87

Chapter 1

smctc – A Template Class Library for SMC Simulation

Version:

1.0

Author:

Adam M. Johansen

1.1 Summary

The SMC template class library (SMCTC) is intended to be used to implement SMC algorithms ranging from simple particle filter to complicated SMC algorithms from simple particle filters to the SMC samplers of Del Moral, Doucet and Jasra (2006) within a generic framework.

1.2 License

SMCTC is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SMCTC is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SMCTC. If not, see <http://www.gnu.org/licenses/>.

The software is still in development but is thought to be sufficiently fully-featured and stable for use.

1.3 Using the Template Library

In order to use the template library it is necessary to place the set of header files in a suitable include directory and then to instantiate the various classes with the appropriate type for your sampler.

For implementation details, please see the accompanying user guide.

1.4 Examples

1.4.1 Simple Particle Filter

This example provides a simple particle filter for the approximate filtering of nonlinear, nongaussian state space models.

1.4.2 Samplers for Rare Event Simulation

The main.cc file provides an example of a use of the template library to estimate Gaussian tail probabilities within an SMC samplers framework by making use of a sequence of intermediate distributions defined by introducing a potential function which varies from flat to being very close to an indicator function on the tail event.

This example is taken from from Johansen, Del Moral and Doucet (2006).

1.4.3 Acknowledgements

Thanks are due to Edmund Jackson and Mark Briers for testing SMCTC on a variety of platforms. The Visual C++ project and solution files are kindly provided by Mark Briers.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

smc (The Sequential Monte Carlo namespace)	9
std (The standard namespace)	11

Chapter 3

Class Index

3.1 Class List

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

smc::exception (SMC Exception class)	13
smc::gslrnginfo (A gsl-rng information handling class (not templated))	16
smc::history< Particle > (A template class for the history associated with a particle system evolving in SMC)	19
smc::historyelement< Particle > (A template class for the elements of a linked list to be used for the history of the sampler)	25
smc::historyflags (Holds a set of flags which describe various properties of the particle system at a given time)	33
smc::moveset< Space > (A template class for a set of moves for use in an SMC samplers framework)	35
smc::particle< Space > (A template class for the particles of an SMC algorithm)	42
smc::rng (A random number generator class)	48
smc::sampler< Space > (A template class for an interacting particle system suitable for SMC sampling)	56

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

include/ history.hh (Classes and function related to the history of the sampler)	69
include/ moveset.hh (Classes and functions which deal with collections of sampler proposal "moves")	71
include/ particle.hh (Class used to store and manipulate a single particle)	73
include/ rng.hh (Random number generation)	75
include/ sampler.hh (Defines the overall sampler object)	77
include/ smc-exception.hh (This file defines exception-handling facilities)	80
include/ smctc.hh (The main header file for SMCTC)	83
src/ history.cc (This file contains the untemplated functions used for storing the history of the system)	85
src/ rng.cc (This file contains the untemplated functions used for dealing with random number generation)	86
src/ smc-exception.cc (The untemplated smc::exception class is implemented here)	87

Chapter 5

Namespace Documentation

5.1 smc Namespace Reference

The Sequential Monte Carlo namespace.

Classes

- class [historyflags](#)
The [historyflags](#) class holds a set of flags which describe various properties of the [particle](#) system at a given time.
- class [historyelement](#)
A template class for the elements of a linked list to be used for the [history](#) of the [sampler](#).
- class [history](#)
A template class for the [history](#) associated with a [particle](#) system evolving in SMC.
- class [moveset](#)
A template class for a set of moves for use in an SMC samplers framework.
- class [particle](#)
A template class for the particles of an SMC algorithm.
- class [gslrnginfo](#)
A gsl-rng information handling class (not templated).
- class [rng](#)
A random number generator class.
- class [sampler](#)
A template class for an interacting [particle](#) system suitable for SMC sampling.
- class [exception](#)
SMC Exception class.

Variables

- [gslrnginfo rngset](#)

The global application instance of the [gslrnginfo](#) class:.

5.1.1 Detailed Description

The Sequential Monte Carlo namespace.

The classes and functions within this namespace are intended to be used for producing implemenetations of SMC samplers and related simulation techniques.

5.1.2 Variable Documentation

5.1.2.1 [gslrnginfo smc::rngset](#)

The global application instance of the [gslrnginfo](#) class:.

Definition at line 79 of file rng.cc.

5.2 std Namespace Reference

The standard namespace.

Functions

- `template<class Particle>`
`ostream & operator<< (ostream &os, smc::history< Particle > h)`
This function will ultimately allow the standard stream operators to be used to display a particle history in a human readable form.
- `template<class Space>`
`std::ostream & operator<< (std::ostream &os, smc::particle< Space > &p)`
Produce a human readable display of an `smc::particle` class using the standard stream operators.
- `template<class Space>`
`std::ostream & operator<< (std::ostream &os, smc::sampler< Space > &s)`
Produce a human-readable display of the state of an `smc::sampler` class using the stream operator.
- `std::ostream & operator<< (std::ostream &, smc::exception &)`
Produce a human-readable display of the state of an `smc::exception` class using the stream operator.

5.2.1 Detailed Description

The standard namespace.

The classes provided within the standard libraries reside within this namespace and the TDSMC class library adds a number of additional operator overloads to some of the standard classes to allow them to deal with our structures.

5.2.2 Function Documentation

5.2.2.1 `std::ostream & std::operator<< (std::ostream & os, smc::exception & e)`

Produce a human-readable display of the state of an `smc::exception` class using the stream operator.

Display a human-readable version of an SMC exception.

Produce a human-readable display of the state of an `smc::exception` class using the stream operator.

Parameters:

- os* The output stream to which the display should be made.
- e* The exception which is to be displayed.
- os* The stream to write to.
- e* The exception class to display.

Returns:

os

Definition at line 32 of file smc-exception.cc.

References `smc::exception::lCode`, `smc::exception::lLine`, `smc::exception::szFile`, and `smc::exception::szMessage`.

5.2.2.2 `template<class Space> std::ostream& std::operator<< (std::ostream & os, smc::sampler< Space > & s) [inline]`

Produce a human-readable display of the state of an `smc::sampler` class using the stream operator.

Parameters:

- os* The output stream to which the display should be made.
- s* The sampler which is to be displayed.

Definition at line 547 of file sampler.hh.

5.2.2.3 `template<class Space> std::ostream& std::operator<< (std::ostream & os, smc::particle< Space > & p) [inline]`

Produce a human readable display of an `smc::particle` class using the standard stream operators.

Parameters:

- os* The output stream to which the display should be made.
- p* The particle which is to be displayed.

Definition at line 141 of file particle.hh.

5.2.2.4 `template<class Particle> ostream& std::operator<< (ostream & os, smc::history< Particle > h) [inline]`

This function will ultimately allow the standard stream operators to be used to display a particle history in a human readable form.

It isn't yet fully implemented.

Definition at line 473 of file history.hh.

Chapter 6

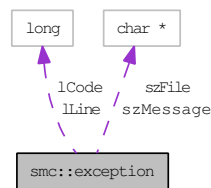
Class Documentation

6.1 smc::exception Class Reference

SMC Exception class.

```
#include <smc-exception.hh>
```

Collaboration diagram for smc::exception:



Public Member Functions

- [exception](#) (char *, long, long, char *)
Generate an SMCTC Exception class with the specified initialisation.

Public Attributes

- char * [szFile](#)
The source file from which the code generating the [exception](#) was generated.
- long [lLine](#)
The line of that source file which generates the [exception](#).
- long [lCode](#)
A numerical code indicating the nature of the [exception](#) generated.
- char * [szMessage](#)
A human-readable explanation of the cause of the [exception](#).

6.1.1 Detailed Description

SMC Exception class.

This class holds details of unrecoverable errors which occur within the SMCTC library. An instance of it is thrown whenever such an error occurs.

Definition at line 48 of file smc-exception.hh.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `smc::exception::exception (char * szN, long lL, long lC, char * szM)`

Generate an SMCTC Exception class with the specified initialisation.

This constructor fills the four elements of the class with their specified values. It is used to allow a single-line command to create and throw an [exception](#).

Parameters:

szN The name of the source file generating the [exception](#).

lL The line in that file responsible for the [exception](#).

lC The numerical code identifying the [exception](#).

szM An textual explanation of the problem.

Definition at line 17 of file smc-exception.cc.

References lCode, lLine, szFile, and szMessage.

6.1.3 Member Data Documentation

6.1.3.1 `char* smc::exception::szFile`

The source file from which the code generating the [exception](#) was generated.

Definition at line 50 of file smc-exception.hh.

Referenced by exception(), and std::operator<<().

6.1.3.2 `long smc::exception::lLine`

The line of that source file which generates the [exception](#).

Definition at line 51 of file smc-exception.hh.

Referenced by exception(), and std::operator<<().

6.1.3.3 `long smc::exception::lCode`

A numerical code indicating the nature of the [exception](#) generated.

Definition at line 52 of file smc-exception.hh.

Referenced by exception(), and std::operator<<().

6.1.3.4 char* smc::exception::szMessage

A human-readable explanation of the cause of the [exception](#).

Definition at line 53 of file smc-exception.hh.

Referenced by [exception\(\)](#), and [std::operator<<\(\)](#).

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

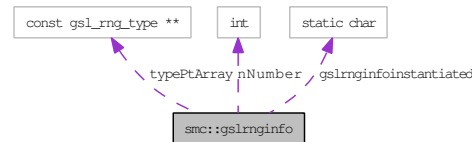
- [include/smc-exception.hh](#)
- [src/smc-exception.cc](#)

6.2 smc::gslrnginfo Class Reference

A gsl-rng information handling class (not templated).

```
#include <rng.hh>
```

Collaboration diagram for smc::gslrnginfo:



Public Member Functions

- [gslrnginfo \(\)](#)
- [int GetNumber \(\)](#)
Returns the number of available random number generators.
- [const char * GetNameByIndex \(int nIndex\)](#)
Returns the name of random number generator number nIndex.
- [const gsl_rng_type * GetPointerByIndex \(int nIndex\)](#)
Returns a pointer to random number generator nIndex.
- [const gsl_rng_type * GetPointerByName \(const char *szName\)](#)
Returns a pointer to the random number generator with name szName (or null if it doesn't exist).

Private Attributes

- [const gsl_rng_type ** typePtArray](#)
This is a null terminated array of the available random number generators.
- [int nNumber](#)
The number of available random number generators.

Static Private Attributes

- [static char gslrnginfoinstantiated = 0](#)
This flag indicates that a [gslrnginfo](#) class has been instantiated.

6.2.1 Detailed Description

A gsl-rng information handling class (not templated).

Definition at line 37 of file rng.hh.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 smc::gslrnginfo::gslrnginfo ()

The GSL provides a mechanism for obtaining a list of available random number generators.

This class provides a wrapper for this mechanism and makes it simple to implement software which allows the nature of the random number generator to be specified at runtime (and to allow it to be a user-specifiable parameter).

For example, [gslrnginfo::GetNumber](#) can be used to determine how many RNGs are available and [gslrnginfo::GetNameByIndex](#) can then be used to populate a list box with their names. Once the user has selected on the [gslrnginfo::GetPointerByName](#) function can be used to obtain a pointer to the appropriate type and this can be used to produce a random number generator of the desired type.

There should be exactly one instance of this class in any program, and that instance is created by the library. A global variable is used to detect multiple instantiations and to throw an [exception](#) when this occurs.

Definition at line 29 of file rng.cc.

References [gslrnginfoinstantiated](#), [nNumber](#), [SMC_EXCEPTION](#), [SMCX_MULTIPLE_INSTANTIATION](#), and [typePtArray](#).

6.2.3 Member Function Documentation

6.2.3.1 int smc::gslrnginfo::GetNumber (void)

Returns the number of available random number generators.

This function returns the number of available generators.

Definition at line 47 of file rng.cc.

References [nNumber](#).

6.2.3.2 const char * smc::gslrnginfo::GetNameByIndex (int nIndex)

Returns the name of random number generator number nIndex.

This function returns the name of the specified generator.

Definition at line 53 of file rng.cc.

References [nNumber](#), and [typePtArray](#).

6.2.3.3 const gsl_rng_type * smc::gslrnginfo::GetPointerByIndex (int nIndex)

Returns a pointer to random number generator nIndex.

This function returns a pointer to the specified generator type.

Definition at line 61 of file rng.cc.

References [nNumber](#), and [typePtArray](#).

6.2.3.4 const gsl_rng_type * smc::gslrnginfo::GetPointerByName (const char * szName)

Returns a pointer to the random number generator with name szName (or null if it doesn't exist).

This function returns a pointer to the specified generator type.

Definition at line 69 of file [rng.cc](#).

References [nNumber](#), and [typePtArray](#).

6.2.4 Member Data Documentation

6.2.4.1 `char smc::gslrnginfo::gslrnginfoinstantiated = 0` `[static, private]`

This flag indicates that a [gslrnginfo](#) class has been instantiated.

Initialises the counter to zero so that the first attempt at creating the relevant class is permitted.

Definition at line 40 of file [rng.hh](#).

Referenced by [gslrnginfo\(\)](#).

6.2.4.2 `const gsl_rng_type** smc::gslrnginfo::typePtArray` `[private]`

This is a null terminated array of the available random number generators.

Definition at line 42 of file [rng.hh](#).

Referenced by [GetNameByIndex\(\)](#), [GetPointerByIndex\(\)](#), [GetPointerByName\(\)](#), and [gslrnginfo\(\)](#).

6.2.4.3 `int smc::gslrnginfo::nNumber` `[private]`

The number of available random number generators.

Definition at line 44 of file [rng.hh](#).

Referenced by [GetNameByIndex\(\)](#), [GetNumber\(\)](#), [GetPointerByIndex\(\)](#), [GetPointerByName\(\)](#), and [gslrnginfo\(\)](#).

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

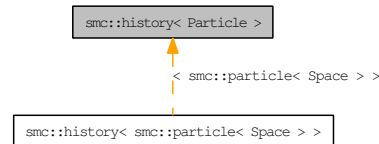
- [include/rng.hh](#)
- [src/rng.cc](#)

6.3 smc::history< Particle > Class Template Reference

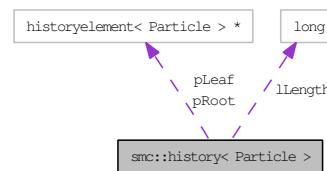
A template class for the [history](#) associated with a [particle](#) system evolving in SMC.

```
#include <history.hh>
```

Inheritance diagram for smc::history< Particle >:



Collaboration diagram for smc::history< Particle >:



Public Member Functions

- [history](#) ()
The argument free constructor creates an empty list.
- const [historyelement](#)< Particle > * [GetElement](#) (void) const
This function returns a pointer to the first element of the [history](#).
- double [GetESS](#) (long lGeneration) const
Returns the effective sample size of the specified [particle](#) generation.
- long [GetLength](#) (void) const
Returns the number of generations recorded within the [history](#).
- double [IntegratePathSampling](#) (double(*pIntegrand)(long, Particle, void *), double(*pWidth)(long, void *), void *pAuxiliary)
Integrate the supplied function over the path of the [particle](#) ensemble.
- double [IntegratePathSamplingFinalStep](#) (double(*pIntegrand)(long, Particle, void *), void *pAuxiliary) const
- void [OstreamMCMCRecordToStream](#) (std::ostream &os) const
Output a vector indicating the number of accepted MCMC moves at each time instance.
- void [OstreamResamplingRecordToStream](#) (std::ostream &os) const
Output a 0-1 value vector indicating the times at which resampling occurred to an output stream.
- Particle * [Pop](#) (void)

Remove the terminal [particle](#) generation from the list and return that [particle](#).

- void [Pop](#) (long *plNumber, Particle **ppNew, int *pnAccept, [historyflags](#) *phf)
Remove the terminal [particle](#) generation from the list and stick it in the supplied data structures.
- void [Push](#) (long lNumber, Particle *pNew, int nAccept, [historyflags](#) hf)
Append the supplied [particle](#) generation to the end of the list.

Private Attributes

- long [lLength](#)
The length of the [history](#) in time steps.
- [historyelement](#)< Particle > * [pRoot](#)
The first time step.
- [historyelement](#)< Particle > * [pLeaf](#)
The most recent time step.

6.3.1 Detailed Description

```
template<class Particle> class smc::history< Particle >
```

A template class for the [history](#) associated with a [particle](#) system evolving in SMC.

The [history](#) is a template class which should have an associated class type corresponding to a `_particle_` of the desired type, not the type itself.

Essentially, this is implemented as a doubly linked list.

Definition at line 233 of file `history.hh`.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `template<class Particle> smc::history< Particle >::history () [inline]`

The argument free constructor creates an empty list.

This constructor simply sets the root and leaf pointers to NULL and the list length to zero.

Definition at line 277 of file `history.hh`.

References `smc::history< Particle >::lLength`, `smc::history< Particle >::pLeaf`, and `smc::history< Particle >::pRoot`.

6.3.3 Member Function Documentation

6.3.3.1 `template<class Particle> const historyelement<Particle >* smc::history< Particle >::GetElement (void) const [inline]`

This function returns a pointer to the first element of the [history](#).

Definition at line 248 of file history.hh.

6.3.3.2 `template<class Particle> double smc::history< Particle >::GetESS (long lGeneration) const [inline]`

Returns the effective sample size of the specified [particle](#) generation.

Definition at line 285 of file history.hh.

References `smc::historyelement< Particle >::GetNext()`, and `smc::history< Particle >::pRoot`.

Here is the call graph for this function:



6.3.3.3 `template<class Particle> long smc::history< Particle >::GetLength (void) const [inline]`

Returns the number of generations recorded within the [history](#).

Definition at line 253 of file history.hh.

6.3.3.4 `template<class Particle> double smc::history< Particle >::IntegratePathSampling (double*)(long, Particle, void *) pIntegrand, double*)(long, void *) pWidth, void * pAuxiliary) [inline]`

Integrate the supplied function over the path of the [particle](#) ensemble.

This function performs a trapezoidal integration of the type which is useful when using path sampling to estimate the normalising constant of a potential function in those cases where a sequence of distributions is produced by deforming the initial distribution by a sequence of progressively more influential potential functions which are proportional to the density of some other distribution with respect to the starting distribution.

The integrand is integrated at every time point in the [particle](#) system [history](#). The results of this integration are taken to be point-evaluations of the path sampling integrand which are spaced on a grid of intervals given by the width function. The path sampling integral is then calculated by performing a suitable trapezoidal integration and the results of this integration is returned.

`pAuxiliary` is passed to both of the user specified functions to allow the user to pass additional data to either or both of these functions in a convenient manner. It is safe to use NULL if no such data is used by either function.

Parameters:

pIntegrand The function to integrated. The first argument is evolution time, the second a [particle](#) at which the function is to be evaluated and the final argument is always `pAuxiliary`.

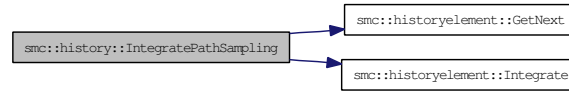
pWidth The function which returns the width of the path sampling grid at the specified evolution time. The final argument is always `pAuxiliary`

pAuxiliary A pointer to auxiliary data to pass to both of the above functions

Definition at line 348 of file history.hh.

References `smc::historyelement< Particle >::GetNext()`, `smc::historyelement< Particle >::Integrate()`, and `smc::history< Particle >::pRoot`.

Here is the call graph for this function:



6.3.3.5 `template<class Particle> double smc::history< Particle >::IntegratePathSamplingFinalStep (double*)(long, Particle, void *) pIntegrand, void * pAuxiliary) const [inline]`

Definition at line 367 of file `history.hh`.

References `smc::history< Particle >::lLength`, and `smc::history< Particle >::pLeaf`.

6.3.3.6 `template<class Particle> void smc::history< Particle >::OstreamMCMCRecordToStream (std::ostream & os) const [inline]`

Output a vector indicating the number of accepted MCMC moves at each time instance.

This function records the MCMC acceptance [history](#) to the specified output stream as a list of the number of moves accepted at each time instant.

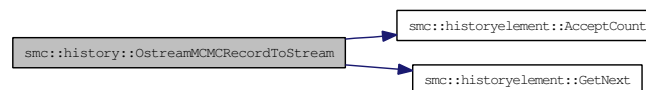
Parameters:

os The output stream to send the data to.

Definition at line 300 of file `history.hh`.

References `smc::historyelement< Particle >::AcceptCount()`, `smc::historyelement< Particle >::GetNext()`, and `smc::history< Particle >::pRoot`.

Here is the call graph for this function:



6.3.3.7 `template<class Particle> void smc::history< Particle >::OstreamResamplingRecordToStream (std::ostream & os) const [inline]`

Output a 0-1 value vector indicating the times at which resampling occurred to an output stream.

This function records the resampling [history](#) to the specified output stream as a 0-1 valued list which takes the value 1 for those time instances when resampling occurred and 0 otherwise.

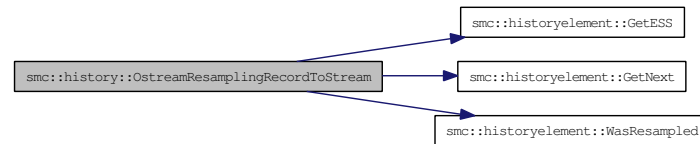
Parameters:

os The output stream to send the data to.

Definition at line 314 of file history.hh.

References smc::historyelement< Particle >::GetESS(), smc::historyelement< Particle >::GetNext(), smc::history< Particle >::pRoot, and smc::historyelement< Particle >::WasResampled().

Here is the call graph for this function:



6.3.3.8 template<class Particle> Particle * smc::history< Particle >::Pop (void) [inline]

Remove the terminal [particle](#) generation from the list and return that [particle](#).

[Pop\(\)](#) operates just as the standard stack operation does. It removes the [particle](#) generation currently occupying the terminal position in the chain, decrements the length counter and returns the [particle](#) set as an array.

Definition at line 376 of file history.hh.

References smc::history< Particle >::ILength, smc::history< Particle >::pLeaf, and smc::history< Particle >::pRoot.

6.3.3.9 template<class Particle> void smc::history< Particle >::Pop (long * *plNumber*, Particle ** *ppNew*, int * *pnAccept*, historyflags * *phf*) [inline]

Remove the terminal [particle](#) generation from the list and stick it in the supplied data structures.

Pop operates as the usual stack operation

If called with four pointers of this sort, it removes the last [particle](#) generation from the [history](#) stack and places them in the reference objects.

Definition at line 401 of file history.hh.

References smc::history< Particle >::ILength, smc::history< Particle >::pLeaf, and smc::history< Particle >::pRoot.

6.3.3.10 template<class Particle> void smc::history< Particle >::Push (long *lNumber*, Particle * *pNew*, int *nAccepts*, historyflags *hf*) [inline]

Append the supplied [particle](#) generation to the end of the list.

Push operates just like the standard stack operation: it adds the specified [particle](#) set generation to the [history](#) of the [particle](#) set and increments the stack counter.

Parameters:

lNumber The number of particles present in this generation of the system.

pNew An array containing the particles present in this generation of the system.

nAccepts The number of accepted MCMC moves during this iteration of the system

hf The [historyflags](#) associated with this generation of the system.

Definition at line 440 of file history.hh.

References `smc::historyelement< Particle >::GetLast()`, `smc::history< Particle >::lLength`, `smc::history< Particle >::pLeaf`, and `smc::history< Particle >::pRoot`.

Here is the call graph for this function:



6.3.4 Member Data Documentation

6.3.4.1 `template<class Particle> long smc::history< Particle >::lLength` [private]

The length of the [history](#) in time steps.

Definition at line 237 of file history.hh.

Referenced by `smc::history< Particle >::history()`, `smc::history< Particle >::IntegratePathSamplingFinalStep()`, `smc::history< Particle >::Pop()`, and `smc::history< Particle >::Push()`.

6.3.4.2 `template<class Particle> historyelement<Particle>* smc::history< Particle >::pRoot` [private]

The first time step.

Definition at line 239 of file history.hh.

Referenced by `smc::history< Particle >::GetESS()`, `smc::history< Particle >::history()`, `smc::history< Particle >::IntegratePathSampling()`, `smc::history< Particle >::OstreamMCMCRecordToStream()`, `smc::history< Particle >::OstreamResamplingRecordToStream()`, `smc::history< Particle >::Pop()`, and `smc::history< Particle >::Push()`.

6.3.4.3 `template<class Particle> historyelement<Particle>* smc::history< Particle >::pLeaf` [private]

The most recent time step.

Definition at line 241 of file history.hh.

Referenced by `smc::history< Particle >::history()`, `smc::history< Particle >::IntegratePathSamplingFinalStep()`, `smc::history< Particle >::Pop()`, and `smc::history< Particle >::Push()`.

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

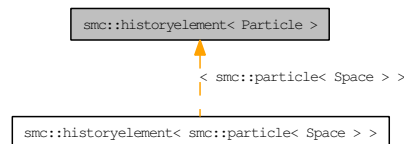
- [include/history.hh](#)

6.4 smc::historyelement< Particle > Class Template Reference

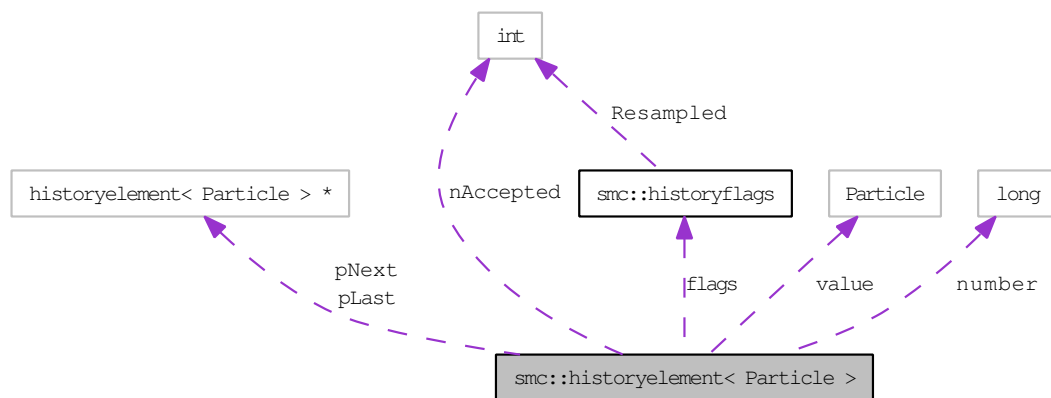
A template class for the elements of a linked list to be used for the [history](#) of the [sampler](#).

```
#include <history.hh>
```

Inheritance diagram for smc::historyelement< Particle >:



Collaboration diagram for smc::historyelement< Particle >:



Public Member Functions

- [historyelement](#) ()
The null constructor creates an empty [history](#) element.
- [historyelement](#) (long lNumber, Particle *pNew, int nAccepts, [historyflags](#) hf)
A constructor with four arguments initialises the [particle](#) set.
- [historyelement](#) (long lNumber, Particle *pNew, [historyelement](#)< Particle > *pParent, [historyelement](#)< Particle > *pChild, int nAccepts, [historyflags](#) hf)
A constructor with six arguments initialises the whole structure.
- [~historyelement](#) ()
The destructor tidies up.
- double [GetESS](#) (void) const
Returns the effective sample size of this [particle](#) generation.
- [historyflags](#) [GetFlags](#) (void) const
Returns the flags.

- `historyelement< Particle > * GetLast (void) const`
Returns the parent of this element.
- `historyelement< Particle > * GetNext (void) const`
Returns the child of this element.
- `long GetNumber (void) const`
Returns the number of particles present.
- `Particle * GetValues (void) const`
*Returns a pointer to the current *particle* set.*
- `void InsertAfter (long lNumber, Particle *pNew)`
*Add a new *history* element with the specified values after the current one and maintain the list structure.*
- `long double Integrate (long lTime, double(*pIntegrand)(long, Particle, void *), void *pAuxiliary)`
*Integrate the supplied function according to the empirical measure of the *particle* ensemble.*
- `void SetLast (historyelement< Particle > *pParent)`
Sets the elements parent.
- `void SetNext (historyelement< Particle > *pChild)`
Sets the elements child.
- `void SetValue (long lNumber, Particle *pNew)`
*Sets the *particle* set to the specified values.*
- `int AcceptCount (void)`
Returns the number of MCMC moves accepted during this iteration.
- `int WasResampled (void)`
*Returns true if the *particle* set.*
- `historyelement< Particle > operator<< (unsigned long ulCount)`
The left shift operator returns the element a number of positions prior to this one.
- `historyelement< Particle > operator>> (unsigned long ulCount)`
The right shift operator returns the element a number of positions after this one.

Private Attributes

- `long number`
The number of particles (presently redundant as this is not a function of iteration).
- `int nAccepted`
Number of MCMC moves accepted during this iteration.

- Particle * [value](#)
The particles themselves (values and weights).
- [historyflags flags](#)
Flags associated with this iteration.
- [historyelement< Particle > * pLast](#)
The parent of this node.
- [historyelement< Particle > * pNext](#)
The child of this node.

6.4.1 Detailed Description

template<class Particle> class smc::historyelement< Particle >

A template class for the elements of a linked list to be used for the [history](#) of the [sampler](#).

Definition at line 46 of file history.hh.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 template<class Particle> smc::historyelement< Particle >::historyelement () `[inline]`

The null constructor creates an empty [history](#) element.

Definition at line 122 of file history.hh.

References [smc::historyelement< Particle >::nAccepted](#), [smc::historyelement< Particle >::number](#), [smc::historyelement< Particle >::pLast](#), [smc::historyelement< Particle >::pNext](#), and [smc::historyelement< Particle >::value](#).

6.4.2.2 template<class Particle> smc::historyelement< Particle >::historyelement (long lNumber, Particle * pNew, int nAccepts, historyflags hf) `[inline]`

A constructor with four arguments initialises the [particle](#) set.

Parameters:

- lNumber** The number of particles present in the [particle](#) generation
- pNew** The array of particles which are present in the [particle](#) generation
- nAccepts** The number of MCMC moves that were accepted during this [particle](#) generation
- hf** The [historyflags](#) associated with the [particle](#) generation

Definition at line 138 of file history.hh.

References [smc::historyelement< Particle >::nAccepted](#), [smc::historyelement< Particle >::number](#), [smc::historyelement< Particle >::pLast](#), [smc::historyelement< Particle >::pNext](#), and [smc::historyelement< Particle >::value](#).

6.4.2.3 `template<class Particle> smc::historyelement< Particle >::historyelement (long lNumber, Particle *pNew, historyelement< Particle > *pParent, historyelement< Particle > *pChild, int nAccepts, historyflags hf)` `[inline]`

A constructor with six arguments initialises the whole structure.

Parameters:

lNumber The number of particles present in the [particle](#) generation
pNew The array of particles which are present in the [particle](#) generation
pParent A pointer to the previous [particle](#) generation
pChild A pointer to the next [particle](#) generation
nAccepts The number of MCMC moves that were accepted during this [particle](#) generation
hf The [historyflags](#) associated with the [particle](#) generation

Definition at line 158 of file history.hh.

References `smc::historyelement< Particle >::nAccepted`, `smc::historyelement< Particle >::number`, `smc::historyelement< Particle >::pLast`, `smc::historyelement< Particle >::pNext`, and `smc::historyelement< Particle >::value`.

6.4.2.4 `template<class Particle> smc::historyelement< Particle >::~~historyelement (void)` `[inline]`

The destructor tidies up.

Definition at line 174 of file history.hh.

References `smc::historyelement< Particle >::value`.

6.4.3 Member Function Documentation

6.4.3.1 `template<class Particle> double smc::historyelement< Particle >::GetESS (void) const` `[inline]`

Returns the effective sample size of this [particle](#) generation.

Definition at line 181 of file history.hh.

References `smc::historyelement< Particle >::number`, and `smc::historyelement< Particle >::value`.

Referenced by `smc::history< Particle >::OstreamResamplingRecordToStream()`.

6.4.3.2 `template<class Particle> historyflags smc::historyelement< Particle >::GetFlags (void) const` `[inline]`

Returns the flags.

Definition at line 70 of file history.hh.

6.4.3.3 `template<class Particle> historyelement<Particle>* smc::historyelement< Particle >::GetLast (void) const` `[inline]`

Returns the parent of this element.

Definition at line 72 of file history.hh.

Referenced by smc::history< Particle >::Push().

6.4.3.4 `template<class Particle> historyelement<Particle>* smc::historyelement< Particle >::GetNext (void) const [inline]`

Returns the child of this element.

Definition at line 74 of file history.hh.

Referenced by smc::history< Particle >::GetESS(), smc::history< Particle >::IntegratePathSampling(), smc::history< Particle >::OstreamMCMCRecordToStream(), and smc::history< Particle >::OstreamResamplingRecordToStream().

6.4.3.5 `template<class Particle> long smc::historyelement< Particle >::GetNumber (void) const [inline]`

Returns the number of particles present.

Definition at line 76 of file history.hh.

6.4.3.6 `template<class Particle> Particle* smc::historyelement< Particle >::GetValues (void) const [inline]`

Returns a pointer to the current [particle](#) set.

Definition at line 78 of file history.hh.

6.4.3.7 `template<class Particle> void smc::historyelement< Particle >::InsertAfter (long lNumber, Particle *pNew) [inline]`

Add a new [history](#) element with the specified values after the current one and maintain the list structure.

Parameters:

lNumber The number of particles in the generation to be inserted

pNew The value of the [particle](#) generation to be inserted

Definition at line 219 of file history.hh.

References smc::historyelement< Particle >::pNext.

6.4.3.8 `template<class Particle> long double smc::historyelement< Particle >::Integrate (long lTime, double(*) (long, Particle, void *) pIntegrand, void * pAuxiliary) [inline]`

Integrate the supplied function according to the empirical measure of the [particle](#) ensemble.

Parameters:

lTime The timestep at which the integration is to be carried out

pIntegrand The function which is to be integrated

pAuxiliary A pointer to additional information which is passed to the integrand function

Definition at line 198 of file history.hh.

References `smc::historyelement< Particle >::number`, and `smc::historyelement< Particle >::value`.

Referenced by `smc::history< Particle >::IntegratePathSampling()`.

6.4.3.9 `template<class Particle> void smc::historyelement< Particle >::SetLast (historyelement< Particle > * pParent)` [inline]

Sets the elements parent.

Definition at line 84 of file history.hh.

6.4.3.10 `template<class Particle> void smc::historyelement< Particle >::SetNext (historyelement< Particle > * pChild)` [inline]

Sets the elements child.

Definition at line 86 of file history.hh.

6.4.3.11 `template<class Particle> void smc::historyelement< Particle >::SetValue (long lNumber, Particle * pNew)`

Sets the [particle](#) set to the specified values.

6.4.3.12 `template<class Particle> int smc::historyelement< Particle >::AcceptCount (void)` [inline]

Returns the number of MCMC moves accepted during this iteration.

Definition at line 91 of file history.hh.

Referenced by `smc::history< Particle >::OstreamMCMCRecordToStream()`.

6.4.3.13 `template<class Particle> int smc::historyelement< Particle >::WasResampled (void)` [inline]

Returns true if the [particle](#) set.

Definition at line 93 of file history.hh.

Referenced by `smc::history< Particle >::OstreamResamplingRecordToStream()`.

6.4.3.14 `template<class Particle> historyelement<Particle> smc::historyelement< Particle >::operator<< (unsigned long ulCount)` [inline]

The left shift operator returns the element a number of positions prior to this one.

Parameters:

ulCount The number of positions by which to shift.

Returns:

The element a number of positions before this one.

Definition at line 99 of file history.hh.

6.4.3.15 `template<class Particle> historyelement<Particle> smc::historyelement< Particle >::operator>> (unsigned long ulCount) [inline]`

The right shift operator returns the element a number of positions after this one.

Parameters:

ulCount The number of positions by which to shift.

Returns:

The right shift operator returns the element a number of positions after this one.

Definition at line 111 of file history.hh.

6.4.4 Member Data Documentation

6.4.4.1 `template<class Particle> long smc::historyelement< Particle >::number [private]`

The number of particles (presently redundant as this is not a function of iteration).

Definition at line 49 of file history.hh.

Referenced by `smc::historyelement< Particle >::GetESS()`, `smc::historyelement< smc::particle< Space > >::GetNumber()`, `smc::historyelement< Particle >::historyelement()`, and `smc::historyelement< Particle >::Integrate()`.

6.4.4.2 `template<class Particle> int smc::historyelement< Particle >::nAccepted [private]`

Number of MCMC moves accepted during this iteration.

Definition at line 50 of file history.hh.

Referenced by `smc::historyelement< smc::particle< Space > >::AcceptCount()`, and `smc::historyelement< Particle >::historyelement()`.

6.4.4.3 `template<class Particle> Particle* smc::historyelement< Particle >::value [private]`

The particles themselves (values and weights).

Definition at line 51 of file history.hh.

Referenced by `smc::historyelement< Particle >::GetESS()`, `smc::historyelement< smc::particle< Space > >::GetValues()`, `smc::historyelement< Particle >::historyelement()`, `smc::historyelement< Particle >::Integrate()`, and `smc::historyelement< Particle >::~~historyelement()`.

6.4.4.4 `template<class Particle> historyflags smc::historyelement< Particle >::flags [private]`

Flags associated with this iteration.

Definition at line 52 of file history.hh.

Referenced by `smc::historyelement< smc::particle< Space > >::GetFlags()`, and `smc::historyelement< smc::particle< Space > >::WasResampled()`.

6.4.4.5 `template<class Particle> historyelement<Particle>*` `smc::historyelement< Particle >::pLast` [private]

The parent of this node.

Definition at line 53 of file `history.hh`.

Referenced by `smc::historyelement< smc::particle< Space > >::GetLast()`, `smc::historyelement< Particle >::historyelement()`, `smc::historyelement< smc::particle< Space > >::operator<<()`, and `smc::historyelement< smc::particle< Space > >::SetLast()`.

6.4.4.6 `template<class Particle> historyelement<Particle>*` `smc::historyelement< Particle >::pNext` [private]

The child of this node.

Definition at line 54 of file `history.hh`.

Referenced by `smc::historyelement< smc::particle< Space > >::GetNext()`, `smc::historyelement< Particle >::historyelement()`, `smc::historyelement< Particle >::InsertAfter()`, `smc::historyelement< smc::particle< Space > >::operator>>()`, and `smc::historyelement< smc::particle< Space > >::SetNext()`.

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

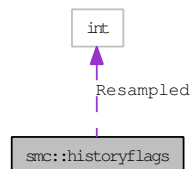
- `include/history.hh`

6.5 smc::historyflags Class Reference

The [historyflags](#) class holds a set of flags which describe various properties of the [particle](#) system at a given time.

```
#include <history.hh>
```

Collaboration diagram for smc::historyflags:



Public Member Functions

- [historyflags](#) (int wasResampled)
Create a new set of [history](#) flags corresponding to the specified properties.
- int [WasResampled](#) (void)
This function returns true if the flag set indicates that the ensemble was resampled during the described iteration.

Private Attributes

- int [Resampled](#): 1
true if the [particle](#) system was resampled during the described iteration.

6.5.1 Detailed Description

The [historyflags](#) class holds a set of flags which describe various properties of the [particle](#) system at a given time.

Definition at line 32 of file history.hh.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 smc::historyflags::historyflags (int wasResampled)

Create a new set of [history](#) flags corresponding to the specified properties.

This constructor produces an initialised [historyflags](#) instance.

Parameters:

wasResampled An indicator which should be nonzero if the [particle](#) system was resampled during the iteration being described

Definition at line 12 of file history.cc.

References Resampled.

6.5.3 Member Function Documentation

6.5.3.1 `int smc::historyflags::WasResampled (void)` `[inline]`

This function returns true if the flag set indicates that the ensemble was resampled during the described iteration.

Definition at line 42 of file history.hh.

References Resampled.

Referenced by `smc::historyelement< smc::particle< Space > >::WasResampled()`.

6.5.4 Member Data Documentation

6.5.4.1 `int smc::historyflags::Resampled` `[private]`

true if the [particle](#) system was resampled during the described iteration.

Definition at line 36 of file history.hh.

Referenced by `historyflags()`, and `WasResampled()`.

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

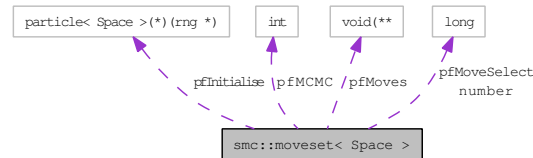
- [include/history.hh](#)
- [src/history.cc](#)

6.6 smc::moveset< Space > Class Template Reference

A template class for a set of moves for use in an SMC samplers framework.

```
#include <moveset.hh>
```

Collaboration diagram for smc::moveset< Space >:



Public Member Functions

- `moveset ()`
Create a completely unspecified *moveset*.
- `moveset (particle< Space >(*pfInit)(rng *), void(*pfNewMoves)(long, particle< Space > &, rng *), int(*pfNewMCMC)(long, particle< Space > &, rng *))`
Create a reduced *moveset* with a single move.
- `moveset (particle< Space >(*pfInit)(rng *), long(*pfMoveSelector)(long, const particle< Space > &, rng *), long nMoves, void(*pfNewMoves)(long, particle< Space > &, rng *), int(*pfNewMCMC)(long, particle< Space > &, rng *))`
Create a fully specified *moveset*.
- `particle< Space > DoInit (rng *pRng)`
Initialise a *particle*.
- `int DoMCMC (long lTime, particle< Space > &pFrom, rng *pRng)`
Perform an MCMC move on a *particle*.
- `void DoMove (long lTime, particle< Space > &pFrom, rng *pRng)`
Select an appropriate move at time *lTime* and apply it to *pFrom*.
- `~moveset ()`
Free the memory used for the array of move pointers when deleting.
- `void SetInitialisor (particle< Space >(*pfInit)(rng *))`
Set the initialisation function.
- `void SetMCMCFunction (int(*pfNewMCMC)(long, particle< Space > &, rng *))`
Set the MCMC function.
- `void SetMoveSelectionFunction (long(*pfMoveSelectNew)(long, const particle< Space > &, rng *))`
Set the move selection function.

- void [SetMoveFunctions](#) (long nMoves, void(**pfNewMoves)(long, [particle](#)< Space > &, [rng](#) *))
Set the individual move functions to the supplied array of such functions.
- [moveset](#)< Space > & [operator=](#) ([moveset](#)< Space > &pFrom)
Moveset assignment should allocate buffers and deep copy all members.

Private Attributes

- long [number](#)
The number of moves which are present in the set.
- [particle](#)< Space >(* [pfInitialise](#))([rng](#) *)
The function which initialises a [particle](#).
- long(* [pfMoveSelect](#))(long, const [particle](#)< Space > &, [rng](#) *)
The function which selects a move for a given [particle](#) at a given time.
- void(** [pfMoves](#))(long, [particle](#)< Space > &, [rng](#) *)
The functions which perform actual moves.
- int(* [pfMCMC](#))(long, [particle](#)< Space > &, [rng](#) *)
A Markov Chain Monte Carlo move.

6.6.1 Detailed Description

template<class Space> class smc::moveset< Space >

A template class for a set of moves for use in an SMC samplers framework.

Definition at line 36 of file moveset.hh.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 **template<class Space> smc::moveset< Space >::moveset ()** [inline]

Create a completely unspecified [moveset](#).

The argument free [smc::moveset](#) constructor simply sets the number of available moves to zero and sets all of the associated function pointers to NULL.

Definition at line 97 of file moveset.hh.

References [smc::moveset< Space >::number](#), [smc::moveset< Space >::pfInitialise](#), [smc::moveset< Space >::pfMCMC](#), [smc::moveset< Space >::pfMoves](#), and [smc::moveset< Space >::pfMoveSelect](#).

6.6.2.2 **template<class Space> smc::moveset< Space >::moveset ([particle](#)< Space >(*)([rng](#) *) [pfInit](#), void(*) (long, [particle](#)< Space > &, [rng](#) *) [pfNewMoves](#), int(*) (long, [particle](#)< Space > &, [rng](#) *) [pfNewMCMC](#))** [inline]

Create a reduced [moveset](#) with a single move.

The three argument [moveset](#) constructor creates a new set of moves and sets all of the relevant function pointers to the supplied values. Only a single move should exist if this constructor is used.

Parameters:

pfInit The function which should be used to initialise particles when the system is initialised

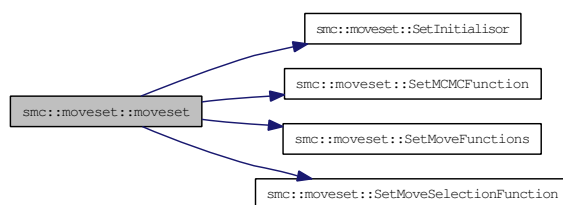
pfNewMoves An functions which moves a [particle](#) at a specified time to a new location

pfNewMCMC The function which should be called to apply an MCMC move (if any)

Definition at line 113 of file moveset.hh.

References smc::moveset< Space >::pfMoves, smc::moveset< Space >::SetInitialisor(), smc::moveset< Space >::SetMCMCFunction(), smc::moveset< Space >::SetMoveFunctions(), and smc::moveset< Space >::SetMoveSelectionFunction().

Here is the call graph for this function:



6.6.2.3 `template<class Space> smc::moveset< Space >::moveset (particle< Space >(*))(rng *)
pfInit, long*)(long, const particle< Space > &, rng *) pfMoveSelector, long nMoves,
 void(**)(long, particle< Space > &, rng *) pfNewMoves, int*)(long, particle< Space > &,
 rng *) pfNewMCMC) [inline]`

Create a fully specified [moveset](#).

The five argument [moveset](#) constructor creates a new set of moves and sets all of the relevant function pointers to the supplied values.

Parameters:

pfInit The function which should be used to initialise particles when the system is initialised

pfMoveSelector The function which selects a move to apply, at a specified time, to a specified [particle](#)

nMoves The number of moves which are defined in general

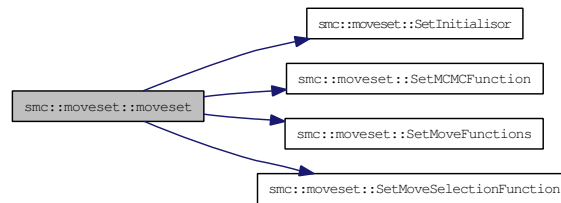
pfNewMoves An array of functions which move a [particle](#) at a specified time to a new location

pfNewMCMC The function which should be called to apply an MCMC move (if any)

Definition at line 132 of file moveset.hh.

References smc::moveset< Space >::pfMoves, smc::moveset< Space >::SetInitialisor(), smc::moveset< Space >::SetMCMCFunction(), smc::moveset< Space >::SetMoveFunctions(), and smc::moveset< Space >::SetMoveSelectionFunction().

Here is the call graph for this function:



6.6.2.4 `template<class Space> smc::moveset< Space >::~~moveset ()` [inline]

Free the memory used for the array of move pointers when deleting.

Definition at line 144 of file moveset.hh.

References `smc::moveset< Space >::pfMoves`.

6.6.3 Member Function Documentation

6.6.3.1 `template<class Space> particle<Space> smc::moveset< Space >::DoInit (rng * pRng)` [inline]

Initialise a [particle](#).

Definition at line 63 of file moveset.hh.

References `smc::moveset< Space >::pfInitialise`.

6.6.3.2 `template<class Space> int smc::moveset< Space >::DoMCMC (long lTime, particle< Space > & pFrom, rng * pRng)` [inline]

Perform an MCMC move on a [particle](#).

Definition at line 151 of file moveset.hh.

References `smc::moveset< Space >::pfMCMC`.

6.6.3.3 `template<class Space> void smc::moveset< Space >::DoMove (long lTime, particle< Space > & pFrom, rng * pRng)` [inline]

Select an appropriate move at time lTime and apply it to pFrom.

Definition at line 162 of file moveset.hh.

References `smc::moveset< Space >::number`, `smc::moveset< Space >::pfMoves`, and `smc::moveset< Space >::pfMoveSelect`.

6.6.3.4 `template<class Space> void smc::moveset< Space >::SetInitialisor (particle< Space >(*) (rng *) pFInit)` [inline]

Set the initialisation function.

Parameters:

pfInit is a function which returns a [particle](#) generated according to the initial distribution

Definition at line 74 of file moveset.hh.

References smc::moveset< Space >::pfInitialise.

Referenced by smc::moveset< Space >::moveset(), and smc::moveset< Space >::operator=().

6.6.3.5 `template<class Space> void smc::moveset< Space >::SetMCMCFunction (int(*) (long, particle< Space > &, rng *) pfNewMCMC) [inline]`

Set the MCMC function.

Parameters:

pfNewMCMC The function which performs an MCMC move

Definition at line 79 of file moveset.hh.

References smc::moveset< Space >::pfMCMC.

Referenced by smc::moveset< Space >::moveset(), and smc::moveset< Space >::operator=().

6.6.3.6 `template<class Space> void smc::moveset< Space >::SetMoveSelectionFunction (long(*) (long, const particle< Space > &, rng *) pfMoveSelectNew) [inline]`

Set the move selection function.

Parameters:

pfMoveSelectNew returns the index of move to perform at the specified time given a specified [particle](#)

Definition at line 83 of file moveset.hh.

References smc::moveset< Space >::pfMoveSelect.

Referenced by smc::moveset< Space >::moveset(), and smc::moveset< Space >::operator=().

6.6.3.7 `template<class Space> void smc::moveset< Space >::SetMoveFunctions (long nMoves, void(**) (long, particle< Space > &, rng *) pfNewMoves) [inline]`

Set the individual move functions to the supplied array of such functions.

Parameters:

nMoves The number of moves which are defined in general.

pfNewMoves An array of functions which move a [particle](#) at a specified time to a new location

The move functions accept two arguments, the first of which corresponds to the system evolution time and the second to an initial [particle](#) position and the second to a weighted starting position. It returns a new weighted position corresponding to the moved [particle](#).

Definition at line 177 of file moveset.hh.

References smc::moveset< Space >::number, and smc::moveset< Space >::pfMoves.

Referenced by smc::moveset< Space >::moveset(), and smc::moveset< Space >::operator=().

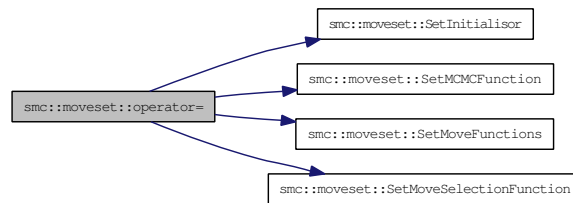
6.6.3.8 `template<class Space> moveset< Space > & smc::moveset< Space >::operator=(moveset< Space > & pFrom) [inline]`

Moveset assignment should allocate buffers and deep copy all members.

Definition at line 189 of file moveset.hh.

References `smc::moveset< Space >::number`, `smc::moveset< Space >::pfInitialise`, `smc::moveset< Space >::pfMCMC`, `smc::moveset< Space >::pfMoves`, `smc::moveset< Space >::pfMoveSelect`, `smc::moveset< Space >::SetInitialisor()`, `smc::moveset< Space >::SetMCMCFunction()`, `smc::moveset< Space >::SetMoveFunctions()`, and `smc::moveset< Space >::SetMoveSelectionFunction()`.

Here is the call graph for this function:



6.6.4 Member Data Documentation

6.6.4.1 `template<class Space> long smc::moveset< Space >::number [private]`

The number of moves which are present in the set.

Definition at line 40 of file moveset.hh.

Referenced by `smc::moveset< Space >::DoMove()`, `smc::moveset< Space >::moveset()`, `smc::moveset< Space >::operator=()`, and `smc::moveset< Space >::SetMoveFunctions()`.

6.6.4.2 `template<class Space> particle<Space>(* smc::moveset< Space >::pfInitialise)(rng *) [private]`

The function which initialises a [particle](#).

Referenced by `smc::moveset< Space >::DoInit()`, `smc::moveset< Space >::moveset()`, `smc::moveset< Space >::operator=()`, and `smc::moveset< Space >::SetInitialisor()`.

6.6.4.3 `template<class Space> long(* smc::moveset< Space >::pfMoveSelect)(long, const particle< Space > &, rng *) [private]`

The function which selects a move for a given [particle](#) at a given time.

Referenced by `smc::moveset< Space >::DoMove()`, `smc::moveset< Space >::moveset()`, `smc::moveset< Space >::operator=()`, and `smc::moveset< Space >::SetMoveSelectionFunction()`.

6.6.4.4 `template<class Space> void(** smc::moveset< Space >::pfMoves)(long, particle< Space > &, rng *) [private]`

The functions which perform actual moves.

Referenced by smc::moveset< Space >::DoMove(), smc::moveset< Space >::moveset(), smc::moveset< Space >::operator=(), smc::moveset< Space >::SetMoveFunctions(), and smc::moveset< Space >::~~moveset().

6.6.4.5 `template<class Space> int(* smc::moveset< Space >::pfMCMC)(long, particle< Space > &, rng *) [private]`

A Markov Chain Monte Carlo move.

Referenced by smc::moveset< Space >::DoMCMC(), smc::moveset< Space >::moveset(), smc::moveset< Space >::operator=(), and smc::moveset< Space >::SetMCMCFunction().

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

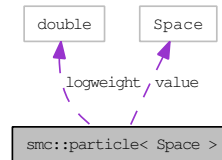
- include/[moveset.hh](#)

6.7 smc::particle< Space > Class Template Reference

A template class for the particles of an SMC algorithm.

```
#include <particle.hh>
```

Collaboration diagram for smc::particle< Space >:



Public Member Functions

- [particle](#) ()
Create a [particle](#) with undefined value and weight NAN.
- [particle](#) (Space sInit, double dLogWeight)
Constructor which initialises the particles value and weight.
- [particle](#) (const [particle](#)< Space > &pFrom)
The copy constructor performs a shallow copy.
- [particle](#)< Space > & [operator=](#) (const [particle](#)< Space > &pFrom)
The assignment operator performs a shallow copy.
- [~particle](#) ()
Dispose of a [particle](#) which is no longer required.
- Space [GetValue](#) (void) const
Returns the particle's value.
- Space * [GetValuePointer](#) (void)
Returns a pointer to the value to allow for more efficient changes.
- double [GetLogWeight](#) (void) const
Returns the particle's log weight.
- double [GetWeight](#) (void) const
Returns the particle's unnormalised weight.
- void [Set](#) (Space sValue, double dLogWeight)
Sets the particle's value and weight explicitly.
- void [SetValue](#) (const Space &sValue)
Sets the particle's value explicitly.
- void [SetLogWeight](#) (const double &dLogWeight)

Sets the particle's log weight explicitly.

- void [SetWeight](#) (const double &dWeight)
Sets the particles weight explicitly.
- void [AddToLogWeight](#) (double dIncrement)
Increase the log weight by a specified amount.
- void [MultiplyWeightBy](#) (double dMultiplier)
Multiply the weight by a specified factor.

Private Attributes

- Space [value](#)
Value of this [particle](#).
- double [logweight](#)
Natural logarithm of this particle's weight.

6.7.1 Detailed Description

template<class Space> class smc::particle< Space >

A template class for the particles of an SMC algorithm.

Definition at line 35 of file particle.hh.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 template<class Space> smc::particle< Space >::particle () [inline]

Create a [particle](#) with undefined value and weight NAN.

Definition at line 94 of file particle.hh.

References `smc::particle< Space >::logweight`.

6.7.2.2 template<class Space> smc::particle< Space >::particle (Space *sInit*, double *dLogWeight*) [inline]

Constructor which initialises the particles value and weight.

Create a [particle](#) with value `sInit` and log weight `dLogWeight`

Parameters:

sInit The initial value of the [particle](#)

dLogWeight The initial value of the natural logarithm of the [particle](#) weight

Definition at line 111 of file particle.hh.

References `smc::particle< Space >::logweight`, and `smc::particle< Space >::value`.

6.7.2.3 `template<class Space> smc::particle< Space >::particle (const particle< Space > & pFrom) [inline]`

The copy constructor performs a shallow copy.

Copy constructor.

Definition at line 101 of file particle.hh.

References `smc::particle< Space >::logweight`, and `smc::particle< Space >::value`.

6.7.2.4 `template<class Space> smc::particle< Space >::~~particle () [inline]`

Dispose of a [particle](#) which is no longer required.

Definition at line 119 of file particle.hh.

6.7.3 Member Function Documentation

6.7.3.1 `template<class Space> particle< Space > & smc::particle< Space >::operator= (const particle< Space > & pFrom) [inline]`

The assignment operator performs a shallow copy.

Copy the values of `pFrom` to the values of this to set this [particle](#) identical to `pFrom` in a deep copy sense.

Definition at line 126 of file particle.hh.

References `smc::particle< Space >::logweight`, and `smc::particle< Space >::value`.

6.7.3.2 `template<class Space> Space smc::particle< Space >::GetValue (void) const [inline]`

Returns the particle's value.

Definition at line 55 of file particle.hh.

References `smc::particle< Space >::value`.

6.7.3.3 `template<class Space> Space* smc::particle< Space >::GetValuePointer (void) [inline]`

Returns a pointer to the value to allow for more efficient changes.

Definition at line 57 of file particle.hh.

References `smc::particle< Space >::value`.

6.7.3.4 `template<class Space> double smc::particle< Space >::GetLogWeight (void) const [inline]`

Returns the particle's log weight.

Definition at line 59 of file particle.hh.

References `smc::particle< Space >::logweight`.

6.7.3.5 `template<class Space> double smc::particle< Space >::GetWeight (void) const` [inline]

Returns the particle's unnormalised weight.

Definition at line 61 of file particle.hh.

References smc::particle< Space >::logweight.

6.7.3.6 `template<class Space> void smc::particle< Space >::Set (Space sValue, double dLogWeight)` [inline]

Sets the particle's value and weight explicitly.

Parameters:

sValue The [particle](#) value to use

dLogWeight The natural logarithm of the new [particle](#) weight

Definition at line 67 of file particle.hh.

References smc::particle< Space >::logweight, and smc::particle< Space >::value.

6.7.3.7 `template<class Space> void smc::particle< Space >::SetValue (const Space & sValue)` [inline]

Sets the particle's value explicitly.

Parameters:

sValue The [particle](#) value to use

Definition at line 71 of file particle.hh.

References smc::particle< Space >::value.

6.7.3.8 `template<class Space> void smc::particle< Space >::SetLogWeight (const double & dLogWeight)` [inline]

Sets the particle's log weight explicitly.

Parameters:

dLogWeight The natural logarithm of the new [particle](#) weight

Definition at line 75 of file particle.hh.

References smc::particle< Space >::logweight.

6.7.3.9 `template<class Space> void smc::particle< Space >::SetWeight (const double & dWeight)` [inline]

Sets the particles weight explicitly.

Parameters:

dWeight The new [particle](#) weight

Definition at line 79 of file particle.hh.

References `smc::particle< Space >::logweight`.

6.7.3.10 `template<class Space> void smc::particle< Space >::AddToLogWeight (double dIncrement) [inline]`

Increase the log weight by a specified amount.

Parameters:

dIncrement The amount to add to the log weight.

Definition at line 84 of file particle.hh.

References `smc::particle< Space >::logweight`.

6.7.3.11 `template<class Space> void smc::particle< Space >::MultiplyWeightBy (double dMultiplier) [inline]`

Multiply the weight by a specified factor.

Parameters:

dMultiplier The factor to multiply the weight by.

Definition at line 88 of file particle.hh.

References `smc::particle< Space >::logweight`.

6.7.4 Member Data Documentation

6.7.4.1 `template<class Space> Space smc::particle< Space >::value [private]`

Value of this [particle](#).

Definition at line 39 of file particle.hh.

Referenced by `smc::particle< Space >::GetValue()`, `smc::particle< Space >::GetValuePointer()`, `smc::particle< Space >::operator=()`, `smc::particle< Space >::particle()`, `smc::particle< Space >::Set()`, and `smc::particle< Space >::SetValue()`.

6.7.4.2 `template<class Space> double smc::particle< Space >::logweight [private]`

Natural logarithm of this particle's weight.

Definition at line 41 of file particle.hh.

Referenced by `smc::particle< Space >::AddToLogWeight()`, `smc::particle< Space >::GetLogWeight()`, `smc::particle< Space >::GetWeight()`, `smc::particle< Space >::MultiplyWeightBy()`, `smc::particle<`

Space >::operator=(), smc::particle< Space >::particle(), smc::particle< Space >::Set(), smc::particle< Space >::SetLogWeight(), and smc::particle< Space >::SetWeight().

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

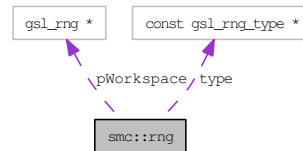
- [include/particle.hh](#)

6.8 smc::rng Class Reference

A random number generator class.

```
#include <rng.hh>
```

Collaboration diagram for smc::rng:



Public Member Functions

- [rng](#) ()
Initialise the random number generator using default settings.
- [rng](#) (const [gsl_rng_type](#) *Type)
Initialise the random number generator using the default seed for the type.
- [rng](#) (const [gsl_rng_type](#) *Type, unsigned long int lSeed)
Initialise the random number generator using specified type and seed.
- [~rng](#) ()
Free the workspace allocated for random number generation.
- [gsl_rng *](#) [GetRaw](#) (void)
Provide access to the raw random number generator.
- void [Multinomial](#) (unsigned n, unsigned k, const double *w, unsigned *X)
Generate a multinomial random vector with parameters (n,w[1:k]) and store it in X.
- long [UniformDiscrete](#) (long lMin, long lMax)
Returns a random integer generated uniformly between the minimum and maximum values specified.
- double [Beta](#) (double da, double db)
Returns a random number generated from a Beta distribution with the specified parameters.
- double [Cauchy](#) (double dScale)
Returns a random number generated from a Cauchy distribution with the specified scale parameter.
- double [Exponential](#) (double dMean)
Returns a random number generated from an exponential distribution with the specified mean.
- double [Gamma](#) (double dAlpha, double dBeta)
Return a random number generated from a gamma distribution with shape alpha and scale beta.
- double [Laplacian](#) (double dScale)

Returns a random number generated from a Laplacian distribution with the specified scale.

- double [Lognormal](#) (double dMu, double dSigma)

Returns a random number generated from a Lognormal distribution of location mu and scale sigma.

- double [Normal](#) (double dMean, double dStd)

Return a random number generated from a normal distribution with a specified mean and standard deviation.

- double [NormalS](#) (void)

Return a random number generated from a standard normal distribution.

- double [NormalTruncated](#) (double dMean, double dStd, double dThreshold)

Returns a random number from a normal distribution, conditional upon it exceeding the specified threshold.

- double [StudentT](#) (double dDF)

Return a student-t random number generated with a specified number of degrees of freedom.

- double [Uniform](#) (double dMin, double dMax)

Return a random number generated uniformly between dMin and dMax.

- double [UniformS](#) (void)

Returns a random number generated from the standard uniform[0,1) distribution.

Private Attributes

- const gsl_rng_type * [type](#)

This is the type of random number generator underlying the class.

- gsl_rng * [pWorkspace](#)

This is a pointer to the internal workspace of the [rng](#) including its current state.

6.8.1 Detailed Description

A random number generator class.

At present this serves as a wrapper for the gsl random number generation code.

Definition at line 67 of file rng.hh.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 smc::rng::rng (void)

Initialise the random number generator using default settings.

When called without any arguments, the constructor for the [smc::rng](#) class simply allocates a buffer for a random number generator of type `gsl_rng_default` (something which can be set at run-time via an environment variable) using its default seed (which again can be over-ridden using an environment variable).

Definition at line 83 of file rng.cc.

References pWorkspace, and type.

6.8.2.2 `smc::rng::rng (const gsl_rng_type * Type)`

Initialise the random number generator using the default seed for the type.

When called with a single argument, the constructor for the `smc::rng` class allocates a buffer for a random number generator of the specified type and initialises it with the default seed (which can be set using and environment variable if one wishes to vary it at run-time).

Parameters:

Type The type of a GSL random number generator

Definition at line 95 of file rng.cc.

References pWorkspace, and type.

6.8.2.3 `smc::rng::rng (const gsl_rng_type * Type, unsigned long int lSeed)`

Initialise the random number generator using specified type and seed.

When called with a pair of arguments, the constructor for the `smc::rng` class allocates a buffer for the specified random number generator type and initialises it with the specified seed (note that zero has special significance and is used to specify the seed with which the generator was originally used).

Parameters:

Type The type of a GSL random number generator

lSeed The value with which the generator is to be seeded

Definition at line 108 of file rng.cc.

References pWorkspace, and type.

6.8.2.4 `smc::rng::~~rng ()`

Free the workspace allocated for random number generation.

The destructor presently does no more than call the `gsl_rng_free` function to deallocate the memory which was previously allocate to the random number generator.

Definition at line 118 of file rng.cc.

References pWorkspace.

6.8.3 Member Function Documentation

6.8.3.1 `gsl_rng * smc::rng::GetRaw (void)`

Provide access to the raw random number generator.

This function returns a pointer to the underlying GSL random number generator which may be used to provide random number facilities which are not explicitly provided by the intermediate layer of `smc::rng`.

Definition at line 125 of file rng.cc.

References pWorkspace.

6.8.3.2 void smc::rng::Multinomial (unsigned n , unsigned k , const double * w , unsigned * X)

Generate a multinomial random vector with parameters ($n, w[1:k]$) and store it in X .

This function simply passes the relevant arguments on to gsl_ran_multinomial.

Parameters:

- n Number of entities to assign.
- k Number of categories.
- w Weights of category elements
- X Array in which to return the sample values.

Definition at line 136 of file rng.cc.

References pWorkspace.

Referenced by smc::sampler< Space >::Resample().

6.8.3.3 long smc::rng::UniformDiscrete (long $lMin$, long $lMax$)

Returns a random integer generated uniformly between the minimum and maximum values specified.

This function simply calls gsl_rng_uniform_int and shifts the result as appropriate such that the result is an integer generated uniformly from those between the two arguments (inclusive of those points).

Parameters:

- $lMin$ The smallest value which can be returned
- $lMax$ the largest value which can be returned

Definition at line 149 of file rng.cc.

References pWorkspace.

6.8.3.4 double smc::rng::Beta (double da , double db)

Returns a random number generated from a Beta distribution with the specified parameters.

This function simply calls gsl_ran_beta with the specified parameters.

Parameters:

- da The parameter associated with "x".
- db The parameter associated with "1-x".

Definition at line 159 of file rng.cc.

References pWorkspace.

6.8.3.5 **double smc::rng::Cauchy (double *dScale*)**

Returns a random number generated from a Cauchy distribution with the specified scale parameter.

This function simply calls `gsl_ran_cauchy` with the specified parameter.

Parameters:

dScale The scale parameter of the distribution.

Definition at line 168 of file `rng.cc`.

References `pWorkspace`.

6.8.3.6 **double smc::rng::Exponential (double *dMean*)**

Returns a random number generated from an exponential distribution with the specified mean.

This function simply calls `gsl_ran_exponential` with the specified parameters.

Parameters:

dMean The scale (not rate) (and mean) of the distribution.

Definition at line 177 of file `rng.cc`.

References `pWorkspace`.

6.8.3.7 **double smc::rng::Gamma (double *dAlpha*, double *dBeta*)**

Return a random number generated from a gamma distribution with shape `alpha` and scale `beta`.

This function simply calls `gsl_ran_gamma` with the specified parameters.

Parameters:

dAlpha The shape of the distribution (integers lead to Erlang distributions)

dBeta The scale (not rate) of the distribution.

Definition at line 187 of file `rng.cc`.

References `pWorkspace`.

6.8.3.8 **double smc::rng::Laplacian (double *dScale*)**

Returns a random number generated from a Laplacian distribution with the specified scale.

This function simply calls `gsl_ran_laplace` with the specified parameters.

Parameters:

dScale The scale (not rate) of the distribution.

Definition at line 196 of file `rng.cc`.

References `pWorkspace`.

6.8.3.9 double smc::rng::Lognormal (double *dMu*, double *dSigma*)

Returns a random number generated from a Lognormal distribution of location mu and scale sigma.

This function simply calls `gsl_ran_lognormal` with the specified parameters.

Parameters:

dMu The location parameter of the distribution.

dSigma The scale parameter of the distribution.

Definition at line 206 of file `rng.cc`.

References `pWorkspace`.

6.8.3.10 double smc::rng::Normal (double *dMean*, double *dStd*)

Return a random number generated from a normal distribution with a specified mean and standard deviation.

This function simply calls `gsl_ran_gaussian` with the specified standard deviation and shifts the result.

Parameters:

dMean The mean of the distribution.

dStd The standard deviation of the distribution

Definition at line 216 of file `rng.cc`.

References `pWorkspace`.

6.8.3.11 double smc::rng::NormalS (void)

Return a random number generated from a standard normal distribution.

This function simply calls `gsl_ran_ugaussian` returns the result.

Definition at line 222 of file `rng.cc`.

References `pWorkspace`.

6.8.3.12 double smc::rng::NormalTruncated (double *dMean*, double *dStd*, double *dThreshold*)

Returns a random number from a normal distribution, conditional upon it exceeding the specified threshold.

This function simply calls `gsl_ran_gaussian_tail` with the specified parameters and performs appropriate shifting.

Parameters:

dMean The mean of the distribution.

dStd The standard deviation of the distribution

dThreshold The lower truncation threshold.

Definition at line 233 of file `rng.cc`.

References `pWorkspace`.

6.8.3.13 double smc::rng::StudentT (double *dDF*)

Return a student-t random number generated with a specified number of degrees of freedom.

This function simply calls `gsl_rng_tdist` with the specified number of degrees of freedom.

Parameters:

dDF The number of degrees of freedom.

Definition at line 241 of file `rng.cc`.

References `pWorkspace`.

6.8.3.14 double smc::rng::Uniform (double *dMin*, double *dMax*)

Return a random number generated uniformly between *dMin* and *dMax*.

This function simply calls `gsl_rng_uniform` and scales and shifts appropriately.

Parameters:

dMin The lowest value with positive density.

dMax The largest value with positive density.

Definition at line 250 of file `rng.cc`.

References `pWorkspace`.

Referenced by `smc::sampler< Space >::Resample()`.

6.8.3.15 double smc::rng::UniformS (void)

Returns a random number generated from the standard uniform[0,1) distribution.

This function simply class `gsl_rng_uniform`.

Definition at line 262 of file `rng.cc`.

References `pWorkspace`.

6.8.4 Member Data Documentation

6.8.4.1 const gsl_rng_type* smc::rng::type [private]

This is the type of random number generator underlying the class.

Definition at line 70 of file `rng.hh`.

Referenced by `rng()`.

6.8.4.2 gsl_rng* smc::rng::pWorkspace [private]

This is a pointer to the internal workspace of the `rng` including its current state.

Definition at line 72 of file `rng.hh`.

Referenced by Beta(), Cauchy(), Exponential(), Gamma(), GetRaw(), Laplacian(), Lognormal(), Multinomial(), Normal(), NormalS(), NormalTruncated(), rng(), StudentT(), Uniform(), UniformDiscrete(), UniformS(), and \sim rng().

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

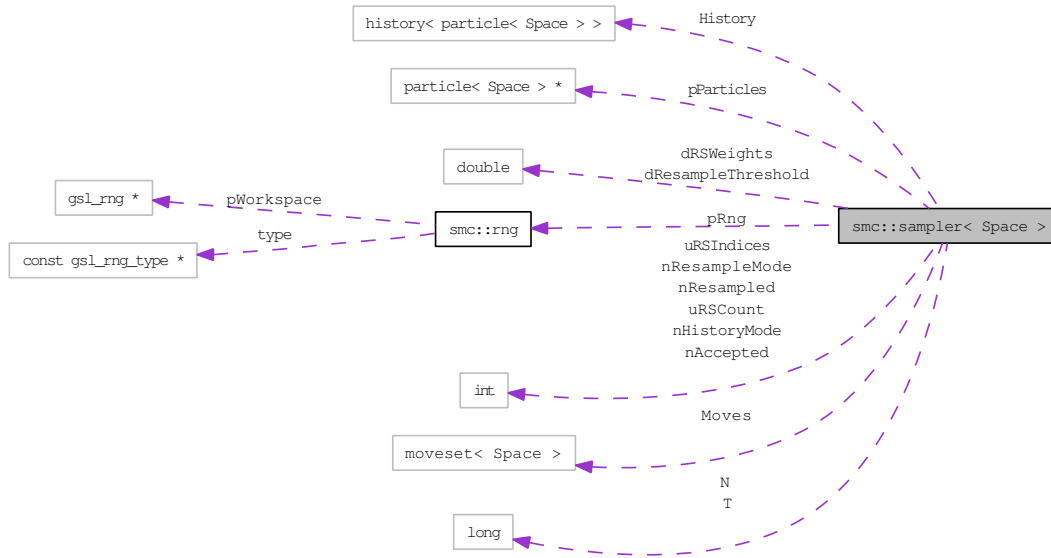
- [include/rng.hh](#)
- [src/rng.cc](#)

6.9 smc::sampler< Space > Class Template Reference

A template class for an interacting [particle](#) system suitable for SMC sampling.

```
#include <sampler.hh>
```

Collaboration diagram for smc::sampler< Space >:



Public Member Functions

- `sampler` (const `sampler< Space >` &sFrom)
Duplication of `smc::sampler` is not currently permitted.
- `sampler< Space > &operator=` (const `sampler< Space >` &sFrom)
Duplication of `smc::sampler` is not currently permitted.
- `sampler` (long lSize, int nHistoryMode)
Create an [particle](#) system containing lSize uninitialised particles with the specified mode.
- `sampler` (long lSize, int nHistoryMode, const `gsl_rng_type *` rngType, unsigned long nSeed)
Create an [particle](#) system constaining lSize uninitialised particles with the specified mode and random number generator.
- `~sampler` ()
Dispose of a [sampler](#).
- `double GetESS` (void) const
Calculates and Returns the Effective Sample Size.
- `const history< particle< Space > > * GetHistory` (void) const
Returns a pointer to the History of the [particle](#) system.

- long [GetNumber](#) (void) const
Returns the number of particles within the system.
- const Space & [GetParticleValue](#) (int n)
Return the value of [particle](#) n.
- double [GetParticleLogWeight](#) (int n)
Return the logarithmic unnormalized weight of [particle](#) n.
- double [GetParticleWeight](#) (int n)
Return the unnormalized weight of [particle](#) n.
- long [GetTime](#) (void) const
Returns the current evolution time of the system.
- void [Initialise](#) (void)
Initialise the [sampler](#) and its constituent particles.
- double [Integrate](#) (double(*pIntegrand)(Space, void *), void *pAuxiliary)
Integrate the supplied function with respect to the current [particle](#) set.
- double [IntegratePathSampling](#) (double(*pIntegrand)(long, [particle](#)< Space >, void *), double(*pWidth)(long, void *), void *pAuxiliary)
Integrate the supplied function over the path path using the supplied width function.
- void [Iterate](#) (void)
Perform one iteration of the simulation algorithm.
- void [IterateBack](#) (void)
Cancel one iteration of the simulation algorithm.
- double [IterateEss](#) (void)
Perform one iteration of the simulation algorithm and return the resulting ess.
- void [IterateUntil](#) (long lTerminate)
Perform iterations until the specified evolution time is reached.
- void [MoveParticles](#) (void)
Move the [particle](#) set by proposing an applying an appropriate move to each [particle](#).
- void [Resample](#) (long lMode)
Resample the [particle](#) set using the specified resampling scheme.
- void [SetMoveSet](#) (moveset< Space > &pNewMoveset)
Sets the entire [moveset](#) to the one which is supplied.
- void [SetResampleParams](#) (int nMode, double dThreshold)
Set Resampling Parameters.
- std::ostream & [StreamParticle](#) (std::ostream &os, long n)

Dump a specified [particle](#) to the specified output stream in a human readable form.

- `std::ostream & StreamParticles (std::ostream &os)`

Dump the entire [particle](#) set to the specified output stream in a human readable form.

Private Attributes

- `rng * pRng`

A random number generator.

- `long N`

Number of particles in the system.

- `long T`

The current evolution time of the system.

- `int nResampleMode`

The resampling mode which is to be employed.

- `double dResampleThreshold`

The effective sample size at which resampling should be used.

- `double * dRSWeights`

Structure used internally for resampling.

- `unsigned int * uRSCount`

Structure used internally for resampling.

- `unsigned int * uRSIndices`

Structure used internally for resampling.

- `particle< Space > * pParticles`

The particles within the system.

- `moveset< Space > Moves`

The set of moves available.

- `int nAccepted`

The number of MCMC moves which have been accepted during this iteration.

- `int nResampled`

A flag which tracks whether the ensemble was resampled during this iteration.

- `int nHistoryMode`

A mode flag which indicates whether historical information is stored.

- `history< particle< Space > > History`

The historical process associated with the [particle](#) system.

6.9.1 Detailed Description

template<class Space> class smc::sampler< Space >

A template class for an interacting [particle](#) system suitable for SMC sampling.

Definition at line 57 of file sampler.hh.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 template<class Space> smc::sampler< Space >::sampler (const sampler< Space > & sFrom) [inline]

Duplication of [smc::sampler](#) is not currently permitted.

Definition at line 96 of file sampler.hh.

References SMC_EXCEPTION, and SMCX_ILLEGAL_COPY.

6.9.2.2 template<class Space> smc::sampler< Space >::sampler (long lSize, int nHM) [inline]

Create an [particle](#) system containing lSize uninitialised particles with the specified mode.

The constructor prepares a [sampler](#) for use but does not assign any moves to the [moveset](#), initialise the particles or otherwise perform any sampling related tasks. Its main function is to allocate a region of memory in which to store the [particle](#) set and to initialise a random number generator.

Parameters:

lSize The number of particles present in the ensemble (at time 0 if this is a variable quantity)

nHM The [history](#) mode to use: set this to SMC_HISTORY_RAM to store the whole [history](#) of the system and SMC_HISTORY_NONE to avoid doing so.

Template Parameters:

Space The class used to represent a point in the sample space.

Structure used internally for resampling.

Structure used internally for resampling.

Definition at line 161 of file sampler.hh.

References smc::sampler< Space >::dResampleThreshold, smc::sampler< Space >::dRSWeights, smc::sampler< Space >::N, smc::sampler< Space >::nHistoryMode, smc::sampler< Space >::nResampleMode, smc::sampler< Space >::pParticles, smc::sampler< Space >::pRng, SMC_RESAMPLE_STRATIFIED, smc::sampler< Space >::uRSCount, and smc::sampler< Space >::uRSIndices.

6.9.2.3 template<class Space> smc::sampler< Space >::sampler (long lSize, int nHM, const gsl_rng_type * rngType, unsigned long rngSeed) [inline]

Create an [particle](#) system constaining lSize uninitialised particles with the specified mode and random number generator.

The constructor prepares a [sampler](#) for use but does not assign any moves to the [moveset](#), initialise the particles or otherwise perform any sampling related tasks. Its main function is to allocate a region of memory in which to store the [particle](#) set and to initialise a random number generator.

Parameters:

- ISize*** The number of particles present in the ensemble (at time 0 if this is a variable quantity)
- nHM*** The [history](#) mode to use: set this to SMC_HISTORY_RAM to store the whole [history](#) of the system and SMC_HISTORY_NONE to avoid doing so.
- rngType*** The type of random number generator to use
- rngSeed*** The seed to use for the random number generator

Template Parameters:

- Space*** The class used to represent a point in the sample space.

Structure used internally for resampling.

Structure used internally for resampling.

Definition at line 190 of file sampler.hh.

References `smc::sampler< Space >::dResampleThreshold`, `smc::sampler< Space >::dRSWeights`, `smc::sampler< Space >::N`, `smc::sampler< Space >::nHistoryMode`, `smc::sampler< Space >::nResampleMode`, `smc::sampler< Space >::pParticles`, `smc::sampler< Space >::pRng`, `SMC_HISTORY_RAM`, `SMC_RESAMPLE_STRATIFIED`, `smc::sampler< Space >::uRSCount`, and `smc::sampler< Space >::uRSIndices`.

6.9.2.4 `template<class Space> smc::sampler< Space >::~~sampler ()` [inline]

Dispose of a [sampler](#).

Definition at line 211 of file sampler.hh.

References `smc::sampler< Space >::dRSWeights`, `smc::sampler< Space >::pRng`, `smc::sampler< Space >::uRSCount`, and `smc::sampler< Space >::uRSIndices`.

6.9.3 Member Function Documentation

6.9.3.1 `template<class Space> sampler<Space>& smc::sampler< Space >::operator= (const sampler< Space > & sFrom)` [inline]

Duplication of [smc::sampler](#) is not currently permitted.

Definition at line 99 of file sampler.hh.

References `SMC_EXCEPTION`, and `SMCX_ILLEGAL_COPY`.

6.9.3.2 `template<class Space> double smc::sampler< Space >::GetESS (void) const` [inline]

Calculates and Returns the Effective Sample Size.

Definition at line 224 of file sampler.hh.

References `smc::sampler< Space >::N`, and `smc::sampler< Space >::pParticles`.

Referenced by `smc::sampler< Space >::IterateEss()`.

6.9.3.3 `template<class Space> const history<particle<Space> >* smc::sampler< Space >::GetHistory (void) const [inline]`

Returns a pointer to the History of the [particle](#) system.

Definition at line 111 of file `sampler.hh`.

References `smc::sampler< Space >::History`.

6.9.3.4 `template<class Space> long smc::sampler< Space >::GetNumber (void) const [inline]`

Returns the number of particles within the system.

Definition at line 113 of file `sampler.hh`.

References `smc::sampler< Space >::N`.

6.9.3.5 `template<class Space> const Space& smc::sampler< Space >::GetParticleValue (int n) [inline]`

Return the value of [particle](#) `n`.

Definition at line 115 of file `sampler.hh`.

References `smc::sampler< Space >::pParticles`.

6.9.3.6 `template<class Space> double smc::sampler< Space >::GetParticleLogWeight (int n) [inline]`

Return the logarithmic unnormalized weight of [particle](#) `n`.

Definition at line 117 of file `sampler.hh`.

References `smc::sampler< Space >::pParticles`.

6.9.3.7 `template<class Space> double smc::sampler< Space >::GetParticleWeight (int n) [inline]`

Return the unnormalized weight of [particle](#) `n`.

Definition at line 119 of file `sampler.hh`.

References `smc::sampler< Space >::pParticles`.

6.9.3.8 `template<class Space> long smc::sampler< Space >::GetTime (void) const [inline]`

Returns the current evolution time of the system.

Definition at line 121 of file `sampler.hh`.

References `smc::sampler< Space >::T`.

6.9.3.9 `template<class Space> void smc::sampler< Space >::Initialise (void) [inline]`

Initialise the [sampler](#) and its constituent particles.

At present this function resets the system evolution time to 0 and calls the [moveset](#) initialisor to assign each [particle](#) in the ensemble.

Note that the initialisation function must be specified before calling this function.

Definition at line 243 of file `sampler.hh`.

References `smc::sampler< Space >::History`, `smc::sampler< Space >::Moves`, `smc::sampler< Space >::N`, `smc::sampler< Space >::nHistoryMode`, `smc::sampler< Space >::nResampled`, `smc::sampler< Space >::pParticles`, `smc::sampler< Space >::pRng`, `SMC_HISTORY_NONE`, and `smc::sampler< Space >::T`.

6.9.3.10 `template<class Space> double smc::sampler< Space >::Integrate (double(*) (Space, void *) pIntegrand, void * pAuxiliary) [inline]`

Integrate the supplied function with respect to the current [particle](#) set.

This function returns the result of integrating the supplied function under the empirical measure associated with the [particle](#) set at the present time. The final argument of the integrand function is a pointer which will be supplied with *pAuxiliary* to allow for arbitrary additional information to be passed to the function being integrated.

Parameters:

pIntegrand The function to integrate with respect to the [particle](#) set

pAuxiliary A pointer to any auxiliary data which should be passed to the function

Definition at line 267 of file `sampler.hh`.

References `smc::sampler< Space >::N`, and `smc::sampler< Space >::pParticles`.

6.9.3.11 `template<class Space> double smc::sampler< Space >::IntegratePathSampling (double(*) (long, particle< Space >, void *) pIntegrand, double(*) (long, void *) pWidth, void * pAuxiliary) [inline]`

Integrate the supplied function over the path path using the supplied width function.

This function is intended to be used to estimate integrals of the sort which must be evaluated to determine the normalising constant of a distribution obtain using a sequence of potential functions proportional to densities with respect to the initial distribution to define a sequence of distributions leading up to the terminal, interesting distribution.

In this context, the [particle](#) set at each time is used to make an estimate of the path sampling integrand, and a trapezoidal integration is then performed to obtain an estimate of the path sampling integral which is the natural logarithm of the ratio of normalising densities.

Parameters:

pIntegrand The quantity which we wish to integrate at each time

pWidth A pointer to a function which specifies the width of each

Definition at line 293 of file `sampler.hh`.

References `smc::sampler< Space >::History`, `smc::sampler< Space >::N`, `smc::sampler< Space >::nAccepted`, `smc::sampler< Space >::nHistoryMode`, `smc::sampler< Space >::nResampled`, `smc::sampler< Space >::pParticles`, `SMC_EXCEPTION`, `SMC_HISTORY_NONE`, and `SMCX_MISSING_HISTORY`.

6.9.3.12 `template<class Space> void smc::sampler< Space >::Iterate (void)` [inline]

Perform one iteration of the simulation algorithm.

The iterate function:

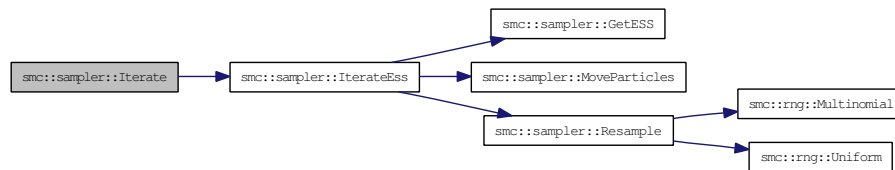
1. appends the current [particle](#) set to the [history](#) if desired
 - (a) moves the current [particle](#) set
2. checks the effective sample size and resamples if necessary
3. performs a mcmc step if required
4. increments the current evolution time

Definition at line 311 of file sampler.hh.

References `smc::sampler< Space >::IterateEss()`.

Referenced by `smc::sampler< Space >::IterateUntil()`.

Here is the call graph for this function:

**6.9.3.13** `template<class Space> void smc::sampler< Space >::IterateBack (void)` [inline]

Cancel one iteration of the simulation algorithm.

Definition at line 318 of file sampler.hh.

References `smc::sampler< Space >::History`, `smc::sampler< Space >::N`, `smc::sampler< Space >::nAccepted`, `smc::sampler< Space >::nHistoryMode`, `smc::sampler< Space >::pParticles`, `SMC_EXCEPTION`, `SMC_HISTORY_NONE`, `SMCX_MISSING_HISTORY`, and `smc::sampler< Space >::T`.

6.9.3.14 `template<class Space> double smc::sampler< Space >::IterateEss (void)` [inline]

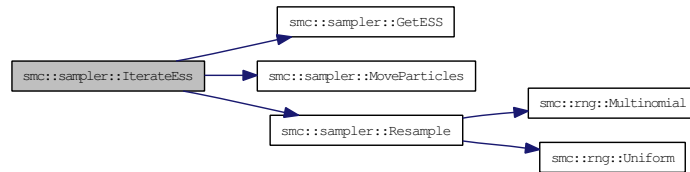
Perform one iteration of the simulation algorithm and return the resulting ess.

Definition at line 329 of file sampler.hh.

References `smc::sampler< Space >::dResampleThreshold`, `smc::sampler< Space >::GetESS()`, `smc::sampler< Space >::History`, `smc::sampler< Space >::MoveParticles()`, `smc::sampler< Space >::Moves`, `smc::sampler< Space >::N`, `smc::sampler< Space >::nAccepted`, `smc::sampler< Space >::nHistoryMode`, `smc::sampler< Space >::nResampled`, `smc::sampler< Space >::nResampleMode`, `smc::sampler< Space >::pParticles`, `smc::sampler< Space >::pRng`, `smc::sampler< Space >::Resample()`, `SMC_HISTORY_NONE`, and `smc::sampler< Space >::T`.

Referenced by `smc::sampler< Space >::Iterate()`.

Here is the call graph for this function:



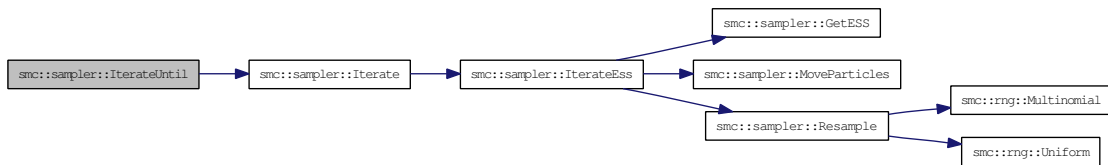
6.9.3.15 `template<class Space> void smc::sampler< Space >::IterateUntil (long lTerminate)` [inline]

Perform iterations until the specified evolution time is reached.

Definition at line 369 of file sampler.hh.

References `smc::sampler< Space >::Iterate()`, and `smc::sampler< Space >::T`.

Here is the call graph for this function:



6.9.3.16 `template<class Space> void smc::sampler< Space >::MoveParticles (void)` [inline]

Move the [particle](#) set by proposing an applying an appropriate move to each [particle](#).

Definition at line 376 of file sampler.hh.

References `smc::sampler< Space >::Moves`, `smc::sampler< Space >::N`, `smc::sampler< Space >::pParticles`, `smc::sampler< Space >::pRng`, and `smc::sampler< Space >::T`.

Referenced by `smc::sampler< Space >::IterateEss()`.

6.9.3.17 `template<class Space> void smc::sampler< Space >::Resample (long lMode)` [inline]

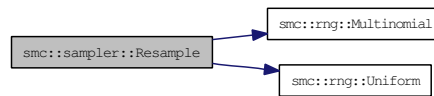
Resample the [particle](#) set using the specified resampling scheme.

Definition at line 385 of file sampler.hh.

References `smc::sampler< Space >::dRSWeights`, `smc::rng::Multinomial()`, `smc::sampler< Space >::N`, `smc::sampler< Space >::pParticles`, `smc::sampler< Space >::pRng`, `SMC_RESAMPLE_MULTINOMIAL`, `SMC_RESAMPLE_RESIDUAL`, `SMC_RESAMPLE_STRATIFIED`, `SMC_RESAMPLE_SYSTEMATIC`, `smc::rng::Uniform()`, `smc::sampler< Space >::uRSCount`, and `smc::sampler< Space >::uRSIndices`.

Referenced by `smc::sampler< Space >::IterateEss()`.

Here is the call graph for this function:



6.9.3.18 `template<class Space> void smc::sampler< Space >::SetMoveSet (moveset< Space > & pNewMoveset) [inline]`

Sets the entire [moveset](#) to the one which is supplied.

Definition at line 141 of file `sampler.hh`.

References `smc::sampler< Space >::Moves`.

6.9.3.19 `template<class Space> void smc::sampler< Space >::SetResampleParams (int nMode, double dThreshold) [inline]`

Set Resampling Parameters.

This function configures the resampling parameters, allowing the specification of both the resampling mode and the threshold at which resampling is used.

Parameters:

nMode The resampling mode to be used.

dThreshold The threshold at which resampling is deemed necessary.

The nMode parameter should be set to one of the following:

1. SMC_RESAMPLE_MULTINOMIAL to use multinomial resampling
2. SMC_RESAMPLE_RESIDUAL to use residual resampling
3. SMC_RESAMPLE_STRATIFIED to use stratified resampling
4. SMC_RESAMPLE_SYSTEMATIC to use systematic resampling

The dThreshold parameter can be set to a value in the range [0,1) corresponding to a fraction of the size of the [particle](#) set or it may be set to an integer corresponding to an actual effective sample size.

Definition at line 513 of file `sampler.hh`.

References `smc::sampler< Space >::dResampleThreshold`, `smc::sampler< Space >::N`, and `smc::sampler< Space >::nResampleMode`.

6.9.3.20 `template<class Space> std::ostream & smc::sampler< Space >::StreamParticle (std::ostream & os, long n) [inline]`

Dump a specified [particle](#) to the specified output stream in a human readable form.

Definition at line 523 of file `sampler.hh`.

References `smc::sampler< Space >::pParticles`.

6.9.3.21 `template<class Space> std::ostream & smc::sampler< Space >::StreamParticles` `(std::ostream & os) [inline]`

Dump the entire [particle](#) set to the specified output stream in a human readable form.

Definition at line 530 of file `sampler.hh`.

References `smc::sampler< Space >::N`, and `smc::sampler< Space >::pParticles`.

6.9.4 Member Data Documentation

6.9.4.1 `template<class Space> rng* smc::sampler< Space >::pRng [private]`

A random number generator.

Definition at line 61 of file `sampler.hh`.

Referenced by `smc::sampler< Space >::Initialise()`, `smc::sampler< Space >::IterateEss()`, `smc::sampler< Space >::MoveParticles()`, `smc::sampler< Space >::Resample()`, `smc::sampler< Space >::sampler()`, and `smc::sampler< Space >::~~sampler()`.

6.9.4.2 `template<class Space> long smc::sampler< Space >::N [private]`

Number of particles in the system.

Definition at line 64 of file `sampler.hh`.

Referenced by `smc::sampler< Space >::GetESS()`, `smc::sampler< Space >::GetNumber()`, `smc::sampler< Space >::Initialise()`, `smc::sampler< Space >::Integrate()`, `smc::sampler< Space >::IntegratePathSampling()`, `smc::sampler< Space >::IterateBack()`, `smc::sampler< Space >::IterateEss()`, `smc::sampler< Space >::MoveParticles()`, `smc::sampler< Space >::Resample()`, `smc::sampler< Space >::sampler()`, `smc::sampler< Space >::SetResampleParams()`, and `smc::sampler< Space >::StreamParticles()`.

6.9.4.3 `template<class Space> long smc::sampler< Space >::T [private]`

The current evolution time of the system.

Definition at line 66 of file `sampler.hh`.

Referenced by `smc::sampler< Space >::GetTime()`, `smc::sampler< Space >::Initialise()`, `smc::sampler< Space >::IterateBack()`, `smc::sampler< Space >::IterateEss()`, `smc::sampler< Space >::IterateUntil()`, and `smc::sampler< Space >::MoveParticles()`.

6.9.4.4 `template<class Space> int smc::sampler< Space >::nResampleMode [private]`

The resampling mode which is to be employed.

Definition at line 69 of file `sampler.hh`.

Referenced by `smc::sampler< Space >::IterateEss()`, `smc::sampler< Space >::sampler()`, and `smc::sampler< Space >::SetResampleParams()`.

6.9.4.5 `template<class Space> double smc::sampler< Space >::dResampleThreshold` [private]

The effective sample size at which resampling should be used.

Definition at line 71 of file sampler.hh.

Referenced by `smc::sampler< Space >::IterateEss()`, `smc::sampler< Space >::sampler()`, and `smc::sampler< Space >::SetResampleParams()`.

6.9.4.6 `template<class Space> double* smc::sampler< Space >::dRSWeights` [private]

Structure used internally for resampling.

Definition at line 73 of file sampler.hh.

Referenced by `smc::sampler< Space >::Resample()`, `smc::sampler< Space >::sampler()`, and `smc::sampler< Space >::~~sampler()`.

6.9.4.7 `template<class Space> unsigned int* smc::sampler< Space >::uRSCount` [private]

Structure used internally for resampling.

Definition at line 75 of file sampler.hh.

Referenced by `smc::sampler< Space >::Resample()`, `smc::sampler< Space >::sampler()`, and `smc::sampler< Space >::~~sampler()`.

6.9.4.8 `template<class Space> unsigned int* smc::sampler< Space >::uRSIndices` [private]

Structure used internally for resampling.

Definition at line 77 of file sampler.hh.

Referenced by `smc::sampler< Space >::Resample()`, `smc::sampler< Space >::sampler()`, and `smc::sampler< Space >::~~sampler()`.

6.9.4.9 `template<class Space> particle<Space>* smc::sampler< Space >::pParticles` [private]

The particles within the system.

Definition at line 80 of file sampler.hh.

Referenced by `smc::sampler< Space >::GetESS()`, `smc::sampler< Space >::GetParticleLogWeight()`, `smc::sampler< Space >::GetParticleValue()`, `smc::sampler< Space >::GetParticleWeight()`, `smc::sampler< Space >::Initialise()`, `smc::sampler< Space >::Integrate()`, `smc::sampler< Space >::IntegratePathSampling()`, `smc::sampler< Space >::IterateBack()`, `smc::sampler< Space >::IterateEss()`, `smc::sampler< Space >::MoveParticles()`, `smc::sampler< Space >::Resample()`, `smc::sampler< Space >::sampler()`, `smc::sampler< Space >::StreamParticle()`, and `smc::sampler< Space >::StreamParticles()`.

6.9.4.10 `template<class Space> moveset<Space> smc::sampler< Space >::Moves` [private]

The set of moves available.

Definition at line 82 of file `sampler.hh`.

Referenced by `smc::sampler< Space >::Initialise()`, `smc::sampler< Space >::IterateEss()`, `smc::sampler< Space >::MoveParticles()`, and `smc::sampler< Space >::SetMoveSet()`.

6.9.4.11 `template<class Space> int smc::sampler< Space >::nAccepted` [private]

The number of MCMC moves which have been accepted during this iteration.

Definition at line 85 of file `sampler.hh`.

Referenced by `smc::sampler< Space >::IntegratePathSampling()`, `smc::sampler< Space >::IterateBack()`, and `smc::sampler< Space >::IterateEss()`.

6.9.4.12 `template<class Space> int smc::sampler< Space >::nResampled` [private]

A flag which tracks whether the ensemble was resampled during this iteration.

Definition at line 87 of file `sampler.hh`.

Referenced by `smc::sampler< Space >::Initialise()`, `smc::sampler< Space >::IntegratePathSampling()`, and `smc::sampler< Space >::IterateEss()`.

6.9.4.13 `template<class Space> int smc::sampler< Space >::nHistoryMode` [private]

A mode flag which indicates whether historical information is stored.

Definition at line 90 of file `sampler.hh`.

Referenced by `smc::sampler< Space >::Initialise()`, `smc::sampler< Space >::IntegratePathSampling()`, `smc::sampler< Space >::IterateBack()`, `smc::sampler< Space >::IterateEss()`, and `smc::sampler< Space >::sampler()`.

6.9.4.14 `template<class Space> history<particle<Space> > smc::sampler< Space >::History` [private]

The historical process associated with the [particle](#) system.

Definition at line 92 of file `sampler.hh`.

Referenced by `smc::sampler< Space >::GetHistory()`, `smc::sampler< Space >::Initialise()`, `smc::sampler< Space >::IntegratePathSampling()`, `smc::sampler< Space >::IterateBack()`, and `smc::sampler< Space >::IterateEss()`.

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

- [include/sampler.hh](#)

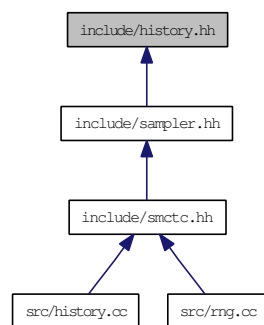
Chapter 7

File Documentation

7.1 include/history.hh File Reference

Classes and function related to the history of the sampler.

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



Namespaces

- namespace [smc](#)
- namespace [std](#)

Classes

- class [smc::historyflags](#)

The [historyflags](#) class holds a set of flags which describe various properties of the [particle](#) system at a given time.

- class [smc::historyelement< Particle >](#)

A template class for the elements of a linked list to be used for the [history](#) of the [sampler](#).

- class [smc::history< Particle >](#)

A template class for the [history](#) associated with a [particle](#) system evolving in SMC.

Defines

- `#define __SMC_HISTORY_HH 1.0`

Functions

- `template<class Particle>`
`ostream & std::operator<< (ostream &os, smc::history< Particle > h)`

This function will ultimately allow the standard stream operators to be used to display a particle history in a human readable form.

7.1.1 Detailed Description

Classes and function related to the history of the sampler.

This file contains template definitions for the classes used to store the history of an SMCTC sampler. It defines [smc::history](#), [smc::historyelement](#) and [smc::history](#).

Definition in file [history.hh](#).

7.1.2 Define Documentation

7.1.2.1 `#define __SMC_HISTORY_HH 1.0`

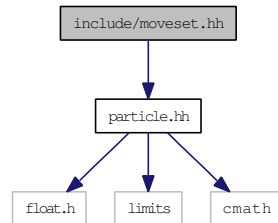
Definition at line 28 of file [history.hh](#).

7.2 include/moveset.hh File Reference

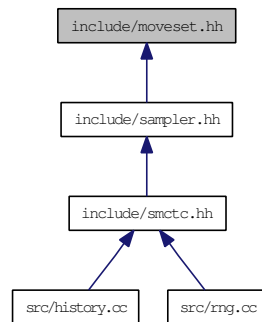
Classes and functions which deal with collections of sampler proposal "moves".

```
#include "particle.hh"
```

Include dependency graph for moveset.hh:



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



Namespaces

- namespace [smc](#)

Classes

- class [smc::moveset< Space >](#)

A template class for a set of moves for use in an SMC samplers framework.

Defines

- #define [__SMC_MOVESET_HH](#) 1.0

7.2.1 Detailed Description

Classes and functions which deal with collections of sampler proposal "moves".

This file contains definitions of [smc::moveset](#). It deals with the collections of proposal moves (including initialisation and MCMC moves) which must be dealt with by the sampler.

Definition in file [moveset.hh](#).

7.2.2 Define Documentation

7.2.2.1 `#define __SMC_MOVESET_HH 1.0`

Definition at line 28 of file moveset.hh.

7.3 include/particle.hh File Reference

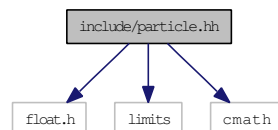
Class used to store and manipulate a single particle.

```
#include <float.h>
```

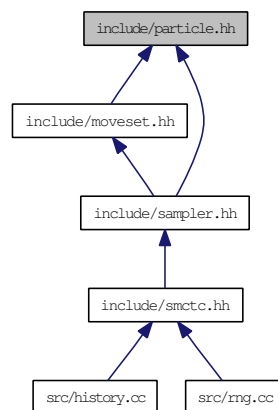
```
#include <limits>
```

```
#include <cmath>
```

Include dependency graph for particle.hh:



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



Namespaces

- namespace [smc](#)
- namespace [std](#)

Classes

- class [smc::particle< Space >](#)

A template class for the particles of an SMC algorithm.

Defines

- `#define __SMC_PARTICLE_HH 1.0`

Functions

- `template<class Space>`
`std::ostream & std::operator<< (std::ostream &os, smc::particle< Space > &p)`
Produce a human readable display of an [smc::particle](#) class using the standard stream operators.

7.3.1 Detailed Description

Class used to store and manipulate a single particle.

This file contains the [smc::particle](#) class which is used internally and passed to move functions.

Definition in file [particle.hh](#).

7.3.2 Define Documentation

7.3.2.1 `#define __SMC_PARTICLE_HH 1.0`

Definition at line 27 of file [particle.hh](#).

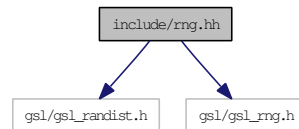
7.4 include/rng.hh File Reference

Random number generation.

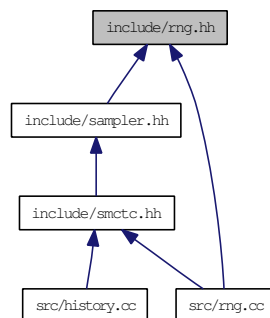
```
#include <gsl/gsl_randist.h>
```

```
#include <gsl/gsl_rng.h>
```

Include dependency graph for rng.hh:



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



Namespaces

- namespace `smc`

Classes

- class `smc::gslrnginfo`
A *gsl-rng* information handling class (not templated).
- class `smc::rng`
A random number generator class.

Defines

- `#define __SMC_RNG_HH` 1.0

Variables

- `gslrnginfo` `smc::rngset`

The global application instance of the [gslrnginfo](#) class.

7.4.1 Detailed Description

Random number generation.

This file contains the definitions for the [smc::rng](#) and [smc::rnginfo](#) class. It wraps the random number generation facilities provided by the GSL and provides a convenient interfaces to access several of its more commonly-used features.

Definition in file [rng.hh](#).

7.4.2 Define Documentation

7.4.2.1 `#define __SMC_RNG_HH 1.0`

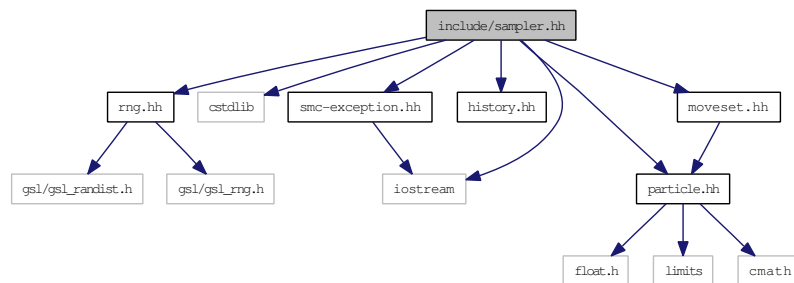
Definition at line 28 of file [rng.hh](#).

7.5 include/sampler.hh File Reference

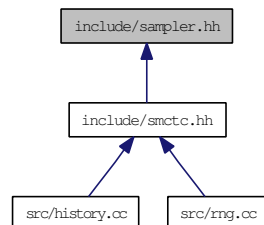
Defines the overall sampler object.

```
#include <iostream>
#include <cstdlib>
#include "rng.hh"
#include "history.hh"
#include "moveset.hh"
#include "particle.hh"
#include "smc-exception.hh"
```

Include dependency graph for sampler.hh:



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



Namespaces

- namespace `smc`
- namespace `std`

Classes

- class `smc::sampler< Space >`

A template class for an interacting `particle` system suitable for SMC sampling.

Defines

- `#define __SMC_SAMPLER_HH 1.0`
- `#define SMC_RESAMPLE_MULTINOMIAL 0`
Pass this to the resampling routine to use multinomial resampling.
- `#define SMC_RESAMPLE_RESIDUAL 1`
Pass this to the resampling routine to use residual resampling.
- `#define SMC_RESAMPLE_STRATIFIED 2`
Pass this to the resampling routine to use stratified resampling.
- `#define SMC_RESAMPLE_SYSTEMATIC 3`
Pass this to the resampling routine to use systematic resampling.
- `#define SMC_HISTORY_NONE 0`
Don't store the history of the particle system, only its final marginal.
- `#define SMC_HISTORY_RAM 1`
Store the entire history of the particle system in memory.

Functions

- `template<class Space>`
`std::ostream & std::operator<< (std::ostream &os, smc::sampler< Space > &s)`
Produce a human-readable display of the state of an [smc::sampler](#) class using the stream operator.

7.5.1 Detailed Description

Defines the overall sampler object.

This file defines the [smc::sampler](#) class which is used to implement entire particle systems.

Definition in file [sampler.hh](#).

7.5.2 Define Documentation

7.5.2.1 `#define __SMC_SAMPLER_HH 1.0`

Definition at line 28 of file [sampler.hh](#).

7.5.2.2 `#define SMC_HISTORY_NONE 0`

Don't store the history of the particle system, only its final marginal.

Definition at line 49 of file [sampler.hh](#).

Referenced by [smc::sampler](#)< Space >::Initialise(), [smc::sampler](#)< Space >::IntegratePathSampling(), [smc::sampler](#)< Space >::IterateBack(), and [smc::sampler](#)< Space >::IterateEss().

7.5.2.3 #define SMC_HISTORY_RAM 1

Store the entire history of the particle system in memory.

Definition at line 51 of file sampler.hh.

Referenced by `smc::sampler< Space >::sampler()`.

7.5.2.4 #define SMC_RESAMPLE_MULTINOMIAL 0

Pass this to the resampling routine to use multinomial resampling.

Definition at line 40 of file sampler.hh.

Referenced by `smc::sampler< Space >::Resample()`.

7.5.2.5 #define SMC_RESAMPLE_RESIDUAL 1

Pass this to the resampling routine to use residual resampling.

Definition at line 42 of file sampler.hh.

Referenced by `smc::sampler< Space >::Resample()`.

7.5.2.6 #define SMC_RESAMPLE_STRATIFIED 2

Pass this to the resampling routine to use stratified resampling.

Definition at line 44 of file sampler.hh.

Referenced by `smc::sampler< Space >::Resample()`, and `smc::sampler< Space >::sampler()`.

7.5.2.7 #define SMC_RESAMPLE_SYSTEMATIC 3

Pass this to the resampling routine to use systematic resampling.

Definition at line 46 of file sampler.hh.

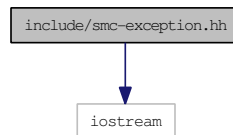
Referenced by `smc::sampler< Space >::Resample()`.

7.6 include/smc-exception.hh File Reference

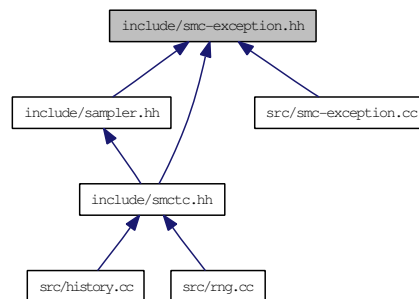
This file defines exception-handling facilities.

```
#include <iostream>
```

Include dependency graph for smc-exception.hh:



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



Namespaces

- namespace [smc](#)
- namespace [std](#)

Classes

- class [smc::exception](#)
SMC Exception class.

Defines

- #define [__SMC_EXCEPT_HH](#) 1.0
- #define [SMC_EXCEPTION](#)(code, error) [smc::exception](#)(__FILE__, __LINE__, code, error)
A macro which autocompletes the housekeeping components of an [smc::exception](#).
- #define [SMCX_FILE_NOT_FOUND](#) 0x0020
Exception thrown if a file cannot be accessed.
- #define [SMCX_ILLEGAL_COPY](#) 0x0001
Exception thrown when something which shouldn't be is duplicated.

- `#define SMCX_MISSING_HISTORY 0x0010`
Exception thrown if the sampler attempts to access history data which wasn't stored.
- `#define SMCX_MULTIPLE_INSTANTIATION 0x1000`
Exception thrown if an attempt is made to instantiate a class of which a single instance is permitted more than once.

Functions

- `std::ostream & std::operator<< (std::ostream &, smc::exception &)`
Produce a human-readable display of the state of an [smc::exception](#) class using the stream operator.

7.6.1 Detailed Description

This file defines exception-handling facilities.

The [smc::exception](#) class, which is used for exception handling by SMCTC, is defined.

Definition in file [smc-exception.hh](#).

7.6.2 Define Documentation

7.6.2.1 `#define __SMC_EXCEPT_HH 1.0`

Definition at line 27 of file [smc-exception.hh](#).

7.6.2.2 `#define SMC_EXCEPTION(code, error) smc::exception(__FILE__, __LINE__, code, error)`

A macro which autocompletes the housekeeping components of an [smc::exception](#).

Definition at line 32 of file [smc-exception.hh](#).

Referenced by [smc::gslrnginfo::gslrnginfo\(\)](#), [smc::sampler< Space >::IntegratePathSampling\(\)](#), [smc::sampler< Space >::IterateBack\(\)](#), [smc::sampler< Space >::operator=\(\)](#), and [smc::sampler< Space >::sampler\(\)](#).

7.6.2.3 `#define SMCX_FILE_NOT_FOUND 0x0020`

Exception thrown if a file cannot be accessed.

Definition at line 35 of file [smc-exception.hh](#).

7.6.2.4 `#define SMCX_ILLEGAL_COPY 0x0001`

Exception thrown when something which shouldn't be is duplicated.

Definition at line 37 of file [smc-exception.hh](#).

Referenced by [smc::sampler< Space >::operator=\(\)](#), and [smc::sampler< Space >::sampler\(\)](#).

7.6.2.5 #define SMCX_MISSING_HISTORY 0x0010

Exception thrown if the sampler attempts to access history data which wasn't stored.

Definition at line 39 of file smc-exception.hh.

Referenced by `smc::sampler< Space >::IntegratePathSampling()`, and `smc::sampler< Space >::IterateBack()`.

7.6.2.6 #define SMCX_MULTIPLE_INSTANTIATION 0x1000

Exception thrown if an attempt is made to instantiate a class of which a single instance is permitted more than once.

Definition at line 41 of file smc-exception.hh.

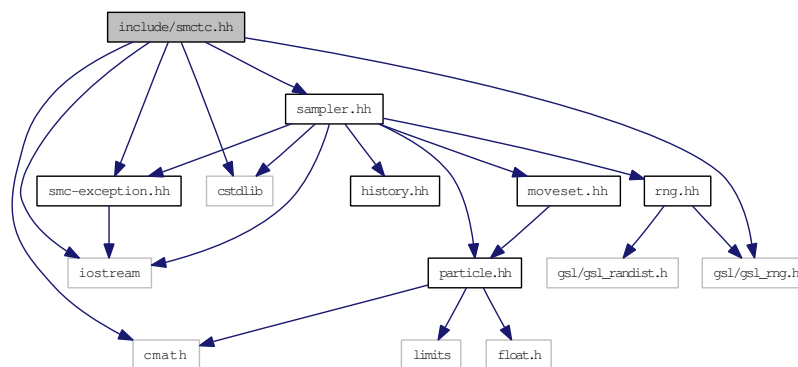
Referenced by `smc::gslrnginfo::gslrnginfo()`.

7.7 include/smctc.hh File Reference

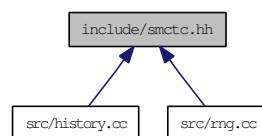
The main header file for SMCTC.

```
#include <cmath>
#include <cstdlib>
#include <iostream>
#include <gsl/gsl_rng.h>
#include "smc-exception.hh"
#include "sampler.hh"
```

Include dependency graph for smctc.hh:



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



Namespaces

- namespace [smc](#)
- namespace [std](#)

Defines

- `#define` [__SMC_TDSMC_HH](#) 1.0

7.7.1 Detailed Description

The main header file for SMCTC.

This file serves as an interface between user-space programs and the SMCTC library. This is the only header file which applications need to include to make use of the library (it includes such additional files as are necessary).

Definition in file [smctc.hh](#).

7.7.2 Define Documentation

7.7.2.1 #define __SMC_TDSMC_HH 1.0

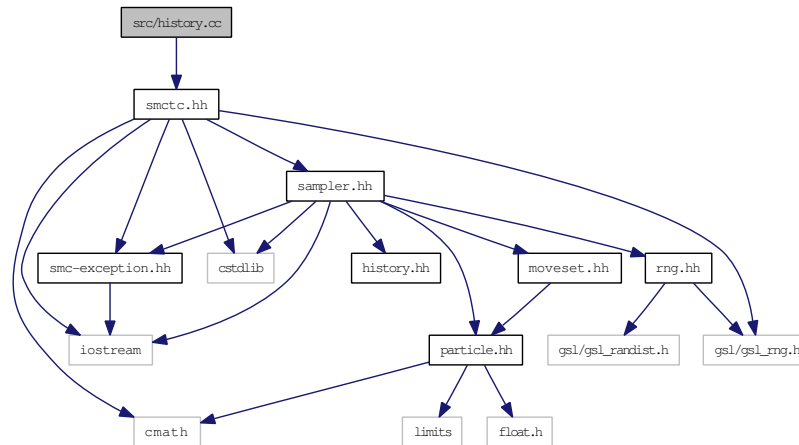
Definition at line 95 of file smctc.hh.

7.8 src/history.cc File Reference

This file contains the untemplated functions used for storing the history of the system.

```
#include "smctc.hh"
```

Include dependency graph for history.cc:



Namespaces

- namespace `smc`

7.8.1 Detailed Description

This file contains the untemplated functions used for storing the history of the system.

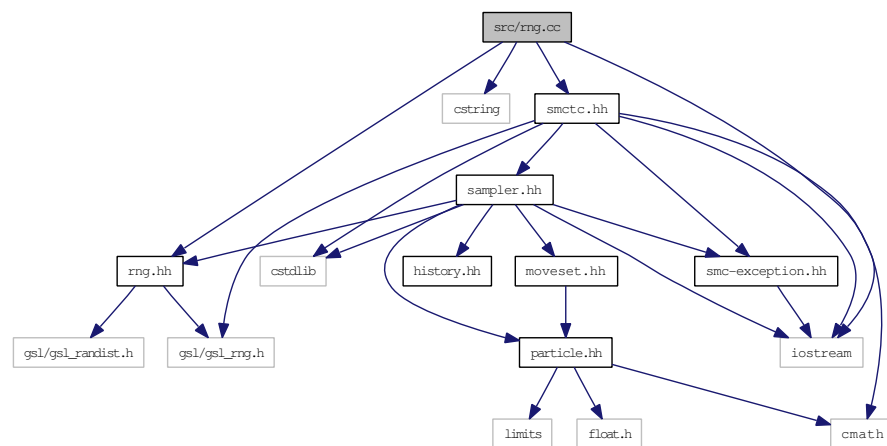
Definition in file `history.cc`.

7.9 src/rng.cc File Reference

This file contains the untemplated functions used for dealing with random number generation.

```
#include <iostream>
#include <cstring>
#include "rng.hh"
#include "smctc.hh"
```

Include dependency graph for rng.cc:



Namespaces

- namespace [smc](#)

7.9.1 Detailed Description

This file contains the untemplated functions used for dealing with random number generation.

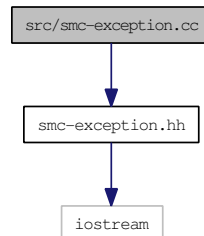
Definition in file [rng.cc](#).

7.10 src/smc-exception.cc File Reference

The untemplated [smc::exception](#) class is implemented here.

```
#include "smc-exception.hh"
```

Include dependency graph for smc-exception.cc:



Namespaces

- namespace [smc](#)
- namespace [std](#)

Functions

- `std::ostream & std::operator<< (std::ostream &, smc::exception &)`

Produce a human-readable display of the state of an [smc::exception](#) class using the stream operator.

7.10.1 Detailed Description

The untemplated [smc::exception](#) class is implemented here.

Definition in file [smc-exception.cc](#).

Index

- ~historyelement
 - smc::historyelement, [28](#)
- ~moveset
 - smc::moveset, [38](#)
- ~particle
 - smc::particle, [44](#)
- ~rng
 - smc::rng, [50](#)
- ~sampler
 - smc::sampler, [60](#)
- __SMC_EXCEPT_HH
 - smc-exception.hh, [81](#)
- __SMC_HISTORY_HH
 - history.hh, [70](#)
- __SMC_MOVESET_HH
 - moveset.hh, [72](#)
- __SMC_PARTICLE_HH
 - particle.hh, [74](#)
- __SMC_RNG_HH
 - rng.hh, [76](#)
- __SMC_SAMPLER_HH
 - sampler.hh, [78](#)
- __SMC_TDSMC_HH
 - smctc.hh, [84](#)
- AcceptCount
 - smc::historyelement, [30](#)
- AddToLogWeight
 - smc::particle, [46](#)
- Beta
 - smc::rng, [51](#)
- Cauchy
 - smc::rng, [51](#)
- DoInit
 - smc::moveset, [38](#)
- DoMCMC
 - smc::moveset, [38](#)
- DoMove
 - smc::moveset, [38](#)
- dResampleThreshold
 - smc::sampler, [66](#)
- dRSWeights
 - smc::sampler, [67](#)
- exception
 - smc::exception, [14](#)
- Exponential
 - smc::rng, [52](#)
- flags
 - smc::historyelement, [31](#)
- Gamma
 - smc::rng, [52](#)
- GetElement
 - smc::history, [20](#)
- GetESS
 - smc::history, [21](#)
 - smc::historyelement, [28](#)
 - smc::sampler, [60](#)
- GetFlags
 - smc::historyelement, [28](#)
- GetHistory
 - smc::sampler, [60](#)
- GetLast
 - smc::historyelement, [28](#)
- GetLength
 - smc::history, [21](#)
- GetLogWeight
 - smc::particle, [44](#)
- GetNameByIndex
 - smc::gslrnginfo, [17](#)
- GetNext
 - smc::historyelement, [29](#)
- GetNumber
 - smc::gslrnginfo, [17](#)
 - smc::historyelement, [29](#)
 - smc::sampler, [61](#)
- GetParticleLogWeight
 - smc::sampler, [61](#)
- GetParticleValue
 - smc::sampler, [61](#)
- GetParticleWeight
 - smc::sampler, [61](#)
- GetPointerByIndex
 - smc::gslrnginfo, [17](#)
- GetPointerByName
 - smc::gslrnginfo, [17](#)
- GetRaw

- smc::rng, 50
- GetTime
 - smc::sampler, 61
- GetValue
 - smc::particle, 44
- GetValuePointer
 - smc::particle, 44
- GetValues
 - smc::historyelement, 29
- GetWeight
 - smc::particle, 44
- gslrnginfo
 - smc::gslrnginfo, 17
- gslrnginfoinstantiated
 - smc::gslrnginfo, 18
- History
 - smc::sampler, 68
- history
 - smc::history, 20
- history.hh
 - __SMC_HISTORY_HH, 70
- historyelement
 - smc::historyelement, 27
- historyflags
 - smc::historyflags, 33
- include/history.hh, 69
- include/moveset.hh, 71
- include/particle.hh, 73
- include/rng.hh, 75
- include/sampler.hh, 77
- include/smc-exception.hh, 80
- include/smctc.hh, 83
- Initialise
 - smc::sampler, 61
- InsertAfter
 - smc::historyelement, 29
- Integrate
 - smc::historyelement, 29
 - smc::sampler, 62
- IntegratePathSampling
 - smc::history, 21
 - smc::sampler, 62
- IntegratePathSamplingFinalStep
 - smc::history, 22
- Iterate
 - smc::sampler, 62
- IterateBack
 - smc::sampler, 63
- IterateEss
 - smc::sampler, 63
- IterateUntil
 - smc::sampler, 64
- Laplacian
 - smc::rng, 52
- lCode
 - smc::exception, 14
- lLength
 - smc::history, 24
- lLine
 - smc::exception, 14
- Lognormal
 - smc::rng, 52
- logweight
 - smc::particle, 46
- MoveParticles
 - smc::sampler, 64
- Moves
 - smc::sampler, 67
- moveset
 - smc::moveset, 36, 37
- moveset.hh
 - __SMC_MOVESET_HH, 72
- Multinomial
 - smc::rng, 51
- MultiplyWeightBy
 - smc::particle, 46
- N
 - smc::sampler, 66
- nAccepted
 - smc::historyelement, 31
 - smc::sampler, 68
- nHistoryMode
 - smc::sampler, 68
- nNumber
 - smc::gslrnginfo, 18
- Normal
 - smc::rng, 53
- NormalS
 - smc::rng, 53
- NormalTruncated
 - smc::rng, 53
- nResampled
 - smc::sampler, 68
- nResampleMode
 - smc::sampler, 66
- number
 - smc::historyelement, 31
 - smc::moveset, 40
- operator<<
 - smc::historyelement, 30
 - std, 11, 12
- operator>>
 - smc::historyelement, 31

- operator=
 - smc::moveset, 39
 - smc::particle, 44
 - smc::sampler, 60
- OstreamMCMCRecordToStream
 - smc::history, 22
- OstreamResamplingRecordToStream
 - smc::history, 22
- particle
 - smc::particle, 43
- particle.hh
 - __SMC_PARTICLE_HH, 74
- pfInitialise
 - smc::moveset, 40
- pfMCMC
 - smc::moveset, 41
- pfMoves
 - smc::moveset, 40
- pfMoveSelect
 - smc::moveset, 40
- pLast
 - smc::historyelement, 32
- pLeaf
 - smc::history, 24
- pNext
 - smc::historyelement, 32
- Pop
 - smc::history, 23
- pParticles
 - smc::sampler, 67
- pRng
 - smc::sampler, 66
- pRoot
 - smc::history, 24
- Push
 - smc::history, 23
- pWorkspace
 - smc::rng, 54
- Resample
 - smc::sampler, 64
- Resampled
 - smc::historyflags, 34
- rng
 - smc::rng, 49, 50
- rng.hh
 - __SMC_RNG_HH, 76
- rngset
 - smc, 10
- sampler
 - smc::sampler, 59
- sampler.hh
 - __SMC_SAMPLER_HH, 78
 - SMC_HISTORY_NONE, 78
 - SMC_HISTORY_RAM, 78
 - SMC_RESAMPLE_MULTINOMIAL, 79
 - SMC_RESAMPLE_RESIDUAL, 79
 - SMC_RESAMPLE_STRATIFIED, 79
 - SMC_RESAMPLE_SYSTEMATIC, 79
- Set
 - smc::particle, 45
- SetInitialisor
 - smc::moveset, 38
- SetLast
 - smc::historyelement, 30
- SetLogWeight
 - smc::particle, 45
- SetMCMCFunction
 - smc::moveset, 39
- SetMoveFunctions
 - smc::moveset, 39
- SetMoveSelectionFunction
 - smc::moveset, 39
- SetMoveSet
 - smc::sampler, 65
- SetNext
 - smc::historyelement, 30
- SetResampleParams
 - smc::sampler, 65
- SetValue
 - smc::historyelement, 30
 - smc::particle, 45
- SetWeight
 - smc::particle, 45
- smc, 9
 - rngset, 10
- smc-exception.hh
 - __SMC_EXCEPT_HH, 81
 - SMC_EXCEPTION, 81
 - SMCX_FILE_NOT_FOUND, 81
 - SMCX_ILLEGAL_COPY, 81
 - SMCX_MISSING_HISTORY, 81
 - SMCX_MULTIPLE_INSTANTIATION, 82
- smc::exception, 13
 - exception, 14
 - lCode, 14
 - lLine, 14
 - szFile, 14
 - szMessage, 14
- smc::gslrnginfo, 16
 - GetNameByIndex, 17
 - GetNumber, 17
 - GetPointerByIndex, 17
 - GetPointerByName, 17
 - gslrnginfo, 17
 - gslrnginfoinstantiated, 18

- nNumber, 18
- typePtArray, 18
- smc::history, 19
 - GetElement, 20
 - GetESS, 21
 - GetLength, 21
 - history, 20
 - IntegratePathSampling, 21
 - IntegratePathSamplingFinalStep, 22
 - lLength, 24
 - OstreamMCMCRecordToStream, 22
 - OstreamResamplingRecordToStream, 22
 - pLeaf, 24
 - Pop, 23
 - pRoot, 24
 - Push, 23
- smc::historyelement, 25
 - ~historyelement, 28
 - AcceptCount, 30
 - flags, 31
 - GetESS, 28
 - GetFlags, 28
 - GetLast, 28
 - GetNext, 29
 - GetNumber, 29
 - GetValues, 29
 - historyelement, 27
 - InsertAfter, 29
 - Integrate, 29
 - nAccepted, 31
 - number, 31
 - operator<<, 30
 - operator>>, 31
 - pLast, 32
 - pNext, 32
 - SetLast, 30
 - SetNext, 30
 - SetValue, 30
 - value, 31
 - WasResampled, 30
- smc::historyflags, 33
 - historyflags, 33
 - Resampled, 34
 - WasResampled, 34
- smc::moveset, 35
 - ~moveset, 38
 - DoInit, 38
 - DoMCMC, 38
 - DoMove, 38
 - moveset, 36, 37
 - number, 40
 - operator=, 39
 - pfInitialise, 40
 - pfMCMC, 41
 - pfMoves, 40
 - pfMoveSelect, 40
 - SetInitialisor, 38
 - SetMCMCFunction, 39
 - SetMoveFunctions, 39
 - SetMoveSelectionFunction, 39
- smc::particle, 42
 - ~particle, 44
 - AddToLogWeight, 46
 - GetLogWeight, 44
 - GetValue, 44
 - GetValuePointer, 44
 - GetWeight, 44
 - logweight, 46
 - MultiplyWeightBy, 46
 - operator=, 44
 - particle, 43
 - Set, 45
 - SetLogWeight, 45
 - SetValue, 45
 - SetWeight, 45
 - value, 46
- smc::rng, 48
 - ~rng, 50
 - Beta, 51
 - Cauchy, 51
 - Exponential, 52
 - Gamma, 52
 - GetRaw, 50
 - Laplacian, 52
 - Lognormal, 52
 - Multinomial, 51
 - Normal, 53
 - NormalS, 53
 - NormalTruncated, 53
 - pWorkspace, 54
 - rng, 49, 50
 - StudentT, 53
 - type, 54
 - Uniform, 54
 - UniformDiscrete, 51
 - UniformS, 54
- smc::sampler, 56
 - ~sampler, 60
 - dResampleThreshold, 66
 - dRSWeights, 67
 - GetESS, 60
 - GetHistory, 60
 - GetNumber, 61
 - GetParticleLogWeight, 61
 - GetParticleValue, 61
 - GetParticleWeight, 61
 - GetTime, 61
 - History, 68

- Initialise, [61](#)
- Integrate, [62](#)
- IntegratePathSampling, [62](#)
- Iterate, [62](#)
- IterateBack, [63](#)
- IterateEss, [63](#)
- IterateUntil, [64](#)
- MoveParticles, [64](#)
- Moves, [67](#)
- N, [66](#)
- nAccepted, [68](#)
- nHistoryMode, [68](#)
- nResampled, [68](#)
- nResampleMode, [66](#)
- operator=, [60](#)
- pParticles, [67](#)
- pRng, [66](#)
- Resample, [64](#)
- sampler, [59](#)
- SetMoveSet, [65](#)
- SetResampleParams, [65](#)
- StreamParticle, [65](#)
- StreamParticles, [65](#)
- T, [66](#)
- uRSCount, [67](#)
- uRSIndices, [67](#)
- SMC_EXCEPTION
 - smc-exception.hh, [81](#)
- SMC_HISTORY_NONE
 - sampler.hh, [78](#)
- SMC_HISTORY_RAM
 - sampler.hh, [78](#)
- SMC_RESAMPLE_MULTINOMIAL
 - sampler.hh, [79](#)
- SMC_RESAMPLE_RESIDUAL
 - sampler.hh, [79](#)
- SMC_RESAMPLE_STRATIFIED
 - sampler.hh, [79](#)
- SMC_RESAMPLE_SYSTEMATIC
 - sampler.hh, [79](#)
- smctc.hh
 - __SMC_TDSMC_HH, [84](#)
- SMCX_FILE_NOT_FOUND
 - smc-exception.hh, [81](#)
- SMCX_ILLEGAL_COPY
 - smc-exception.hh, [81](#)
- SMCX_MISSING_HISTORY
 - smc-exception.hh, [81](#)
- SMCX_MULTIPLE_INSTANTIATION
 - smc-exception.hh, [82](#)
- src/history.cc, [85](#)
- src/rng.cc, [86](#)
- src/smc-exception.cc, [87](#)
- std, [11](#)
- operator<<, [11](#), [12](#)
- StreamParticle
 - smc::sampler, [65](#)
- StreamParticles
 - smc::sampler, [65](#)
- StudentT
 - smc::rng, [53](#)
- szFile
 - smc::exception, [14](#)
- szMessage
 - smc::exception, [14](#)
- T
 - smc::sampler, [66](#)
- type
 - smc::rng, [54](#)
- typePtArray
 - smc::gslrnginfo, [18](#)
- Uniform
 - smc::rng, [54](#)
- UniformDiscrete
 - smc::rng, [51](#)
- UniformS
 - smc::rng, [54](#)
- uRSCount
 - smc::sampler, [67](#)
- uRSIndices
 - smc::sampler, [67](#)
- value
 - smc::historyelement, [31](#)
 - smc::particle, [46](#)
- WasResampled
 - smc::historyelement, [30](#)
 - smc::historyflags, [34](#)