RngStreams

Multiple independent streams of pseudo-random numbers
Version: 1.0
Date: 24 December 2005

Pierre L'Ecuyer
Richard Simard
E. Jack Chen
W. David Kelton

This manual is for RngStreams, a package for generating multiple independent streams of pseudo-random numbers.

Copyright © 2003 Pierre L'Ecuyer, DIRO, University of Montreal.

> Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

RngStreams – Multiple independent streams of pseudo-random numbers

....

1 Installing RngStreams

To install the RngStreams package type

```
./configure --prefix=<prefix_path>
make
```

This should compile the library ('librngstreams.a') and an example program.

To install the library type:

```
make install
```

which installs

'<prefix_path>/lib/librngstreams.a',

'<prefix_path>/lib/librngstreams.so',

'<prefix_path>/include/Rngstream.h', and

'<prefix_path>/info/rngstreams.info'.

If --prefix is omitted, then /usr/local is used as default.

It is possible to remove these files by

```
make uninstall
```

Documentation

A manual can be found in directory 'doc' in various formats, including PS, PDF, HTML, Info and plain text.

Profiling and Verification

To compile and run two test programs type

```
make check
```

2 Interface to the package RngStreams

RngStream Data type

Contains the state of a stream from the present module. It is defined as

```
typedef struct RngStream_InfoState * RngStream;

struct RngStream_InfoState {
   double Cg[6], Bg[6], Ig[6];
   int Anti;
   int IncPrec;
   char *name;
};
```

The arrays `Ig`, `Bg`, and `Cg` contain the initial state, the starting point of the current substream, and the current state, respectively. This stream generates antithetic variates if `Anti` $\neq 0$. The precision of the output numbers is increased if `IncPrec` $\neq 0$.

`void` **RngStream_SetPackageSeed** (`unsigned long` *seed*[6]) Library Function

Sets the initial seed of the package `RngStreams` to the six integers in the vector *seed*. This will be the seed (initial state) of the first stream. If this procedure is not called, the default initial seed is `{12345, 12345, 12345, 12345, 12345, 12345}`. If it is called, the first 3 values of the seed must all be less than `m_1 = 4294967087`, and not all `0`; and the last 3 values must all be less than `m_2 = 4294944443`, and not all `0`.

`RngStream` **RngStream_CreateStream** (`const char` *name*[]) Library Function

Creates and returns a new stream with identifier *name*, whose state variable is of type `RngStream_InfoState`. This procedure reserves space to keep the information relative to the `RngStream`, initializes its seed `Ig`, sets `Bg` and `Cg` equal to `Ig`, sets its antithetic and precision switches to `0`. The seed `Ig` is equal to the initial seed of the package given by `RngStream_SetPackageSeed` if this is the first stream created, otherwise it is `Z` steps ahead of that of the most recently created stream.

`void` **RngStream_DeleteStream** (`RngStream` **pg*) Library Function

Deletes the stream **pg* created previously by `RngStream_CreateStream`, and recovers its memory. Otherwise, does nothing.

`void` **RngStream_ResetStartStream** (`RngStream` *g*) Library Function

Reinitializes the stream *g* to its initial state: `Cg` and `Bg` are set to `Ig`.

`void` **RngStream_ResetStartSubstream** (`RngStream` *g*) Library Function

Reinitializes the stream *g* to the beginning of its current substream: `Cg` is set to *Bg*.

`void` **RngStream_ResetNextSubstream** (`RngStream` *g*) Library Function

Reinitializes the stream *g* to the beginning of its next substream: `Ng` is computed, and `Cg` and `Bg` are set to `Ng`.

`void` **RngStream_SetAntithetic** (`RngStream` *g*, `int` *a*) Library Function

If $a \neq 0$, the stream *g* will start generating antithetic variates, i.e., *1-U* instead of *U*, until this method is called again with $a = 0$. By default, the streams are created with $a = 0$.

`void` **RngStream_IncreasedPrecis** (`RngStream` *g*, `int` *incp*) Library Function

After calling this procedure with $incp \neq 0$, each call (direct or indirect) to `RngStream_RandU01` for stream *g* will advance the state of the stream by 2 steps instead of 1, and will return a number with (roughly) 53 bits of precision instead of 32 bits. More specifically, in the non-antithetic case, when the precision is increased, the instruction `x = RngStream_RandU01(g)` is equivalent to `x = (RngStream_RandU01(g) + RngStream_RandU01(g) * fact) % 1.0` where the constant `fact` is equal to 2^{-24}. This also applies when calling `RngStream_RandU01` indirectly (e.g., by calling `RngStream_RandInt`, etc.). By default, or if this procedure is called again with $incp = 0$, each call to `RngStream_RandU01` for stream *g* advances the state by 1 step and returns a number with 32 bits of precision.

`void` **RngStream_SetSeed** (`RngStream` *g*, `unsigned long` *seed*[6]) Library Function

Sets the initial seed `Ig` of stream *g* to the vector *seed*. This vector must satisfy the same conditions as in `RngStream_SetPackageSeed`. The stream is then reset to this initial seed. The states and seeds of the other streams are not modified. As a result, after calling this procedure, the initial seeds of the streams are no longer spaced `Z` values apart. We discourage the use of this procedure.

`void` **RngStream_AdvanceState** (`RngStream` *g*, `long` *e*, `long` *c*) Library Function

Advances the state of stream *g* by *k* values, without modifying the states of other streams (as in `RngStream_SetSeed`), nor the values of `Bg` and `Ig` associated with this stream. If $e > 0$, then $k = 2^e + c$; if $e < 0$, then $k = -2^{-e} + c$; and if $e = 0$, then $k = c$. Note: *c* is allowed to take negative values. We discourage the use of this procedure.

`void` **RngStream_GetState** (`RngStream` *g*, `unsigned long` *seed*[6]) Library Function

Returns in *seed*[] the current state `Cg` of stream *g*. This is convenient if we want to save the state for subsequent use.

`void` **RngStream_WriteState** (`RngStream` *g*) Library Function

Prints (to standard output) the current state of stream *g*.

`void` **RngStream_WriteStateFull** (`RngStream` *g*) Library Function

Prints (to standard output) the name of stream *g* and the values of all its internal variables.

`double` **RngStream_RandU01** (`RngStream` *g*) Library Function

Returns a (pseudo)random number from the uniform distribution over the interval (0,1), using stream *g*, after advancing the state by one step. The returned number has 32 bits of precision in the sense that it is always a multiple of $1/(2^{32} - 208)$, unless `RngStream_IncreasedPrecis` has been called for this stream.

`long` **RngStream_RandInt** (`RngStream` *g*, `long` *i*, `long` *j*) Library Function

Returns a (pseudo)random number from the discrete uniform distribution over the integers `{` *i*, *i*+1, . . . , *j* `}`, using stream *g*. Makes one call to `RngStream_RandU01`.

3 Example

```
#include <stdio.h>
#include "RngStream.h"

int main (void)
{
   double x;
   int  i;
   RngStream gen;

   /* get a stream */
   gen = RngStream_CreateStream ("generator_1");

   /* sample from generator */
   for (i=0; i<10; i++) {
      x = RngStream_RandU01 (gen);
      printf ("%f\n", x );
   }

   return 0;
}
```