RngStreams: An object-oriented random-number package in C with many long streams and substreams

This file describes a C interface to the `RngStreams` package. The backbone generator is the combined multiple recursive generator (CMRG) `Mrg32k3a` proposed in [1], implemented in 64-bit floating-point arithmetic. This backbone generator has period length $\rho \approx 2^{191}$. The values of V, W, and Z are 2^{51}, 2^{76}, and 2^{127}, respectively. The seed of the RNG, and the state of a stream at any given step, are 6-dimensional vectors of 32-bit integers. The default initial seed of the package is $(12345, 12345, 12345, 12345, 12345, 12345)$.

```
typedef struct RngStream_InfoState * RngStream;

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

The state of a stream from the present module. 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` $\neq 0$.

```
int RngStream_SetPackageSeed (unsigned long seed[6]);
```

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. Returns -1 for invalid seeds, and 0 otherwise.

```
RngStream RngStream_CreateStream (const char name[]);
```

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 I_g, sets B_g and C_g equal to I_g, sets its antithetic and precision switches to 0. The seed I_g 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 g);
```

Deletes the stream `g` created previously by `RngStream_CreateStream`, and recovers its memory.

```
void RngStream_ResetStartStream (RngStream g);
```

Reinitializes the stream `g` to its initial state: C_g and B_g are set to I_g.

```
void RngStream_ResetStartSubstream (RngStream g);
```

Reinitializes the stream `g` to the beginning of its current substream: C_g is set to B_g.

`void RngStream_ResetNextSubstream (RngStream g);`

Reinitializes the stream `g` to the beginning of its next substream: N_g is computed, and C_g and B_g are set to N_g.

`void RngStream_SetAntithetic (RngStream g, int a);`

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`.

`void RngStream_IncreasedPrecis (RngStream g, int incp);`

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, the instruction "`x = RngStream_RandU01(g)`" when the precision is increased 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.

`int RngStream_SetSeed (RngStream g, unsigned long seed[6]);`

Sets the initial seed I_g 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. Returns -1 for invalid seeds, and 0 otherwise.

`void RngStream_AdvanceState (RngStream g, long e, long c);`

Advances the state of stream `g` by k values, without modifying the states of other streams (as in `RngStream_SetSeed`), nor the values of B_g and I_g 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]);`

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

`void RngStream_WriteState (RngStream g);`

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

`void RngStream_WriteStateFull (RngStream g);`

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

`double RngStream_RandU01 (RngStream g);`

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.

`int RngStream_RandInt (RngStream g, int i, int j);`

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

References

[1] P. L'Ecuyer. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, 47(1):159–164, 1999.