Returns

Log-normally distributed float with mean `mean` and standard deviation `stddev`

Description

Return a single log-normally distributed float derived from a normal distribution with mean `mean` and standard deviation `stddev` from the Philox4_32_10 generator in `state`, increment position of generator by one.

The implementation uses a Box-Muller transform to generate two normally distributed results, transforms them to log-normal distribution, then returns them one at a time. See curand_log_normal2() for a more efficient version that returns both results at once.

__device__ float curand_log_normal (curandStateXORWOW_t *state, float mean, float stddev)

Return a log-normally distributed float from an XORWOW generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed float with mean `mean` and standard deviation `stddev`

Description

Return a single log-normally distributed float derived from a normal distribution with mean `mean` and standard deviation `stddev` from the XORWOW generator in `state`, increment position of generator by one.

The implementation uses a Box-Muller transform to generate two normally distributed results, transforms them to log-normal distribution, then returns them one at a time. See curand_log_normal2() for a more efficient version that returns both results at once.

__device__ float2 curand_log_normal2 (curandStateMRG32k3a_t *state, float mean, float stddev)

Return two normally distributed floats from an MRG32k3a generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed float2 where each element is from a distribution with mean `mean` and standard deviation `stddev`

Description

Return two log-normally distributed floats derived from a normal distribution with mean `mean` and standard deviation `stddev` from the MRG32k3a generator in `state`, increment position of generator by two.

The implementation uses a Box-Muller transform to generate two normally distributed results, then transforms them to log-normal.

__device__ float2 curand_log_normal2 (curandStatePhilox4_32_10_t *state, float mean, float stddev)

Return two normally distributed floats from an Philox4_32_10 generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed float2 where each element is from a distribution with mean `mean` and standard deviation `stddev`

Description

Return two log-normally distributed floats derived from a normal distribution with mean `mean` and standard deviation `stddev` from the Philox4_32_10 generator in `state`, increment position of generator by two.

The implementation uses a Box-Muller transform to generate two normally distributed results, then transforms them to log-normal.

__device__ float2 curand_log_normal2 (curandStateXORWOW_t *state, float mean, float stddev)

Return two normally distributed floats from an XORWOW generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed float2 where each element is from a distribution with mean `mean` and standard deviation `stddev`

Description

Return two log-normally distributed floats derived from a normal distribution with mean `mean` and standard deviation `stddev` from the XORWOW generator in `state`, increment position of generator by two.

The implementation uses a Box-Muller transform to generate two normally distributed results, then transforms them to log-normal.

__device__ double2 curand_log_normal2_double (curandStateMRG32k3a_t *state, double mean, double stddev)

Return two log-normally distributed doubles from an MRG32k3a generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed double2 where each element is from a distribution with mean `mean` and standard deviation `stddev`

Description

Return two log-normally distributed doubles derived from a normal distribution with mean `mean` and standard deviation `stddev` from the MRG32k3a generator in `state`, increment position of generator by two.

The implementation uses a Box-Muller transform to generate two normally distributed results, and transforms them to log-normal distribution,.

__device__ double2 curand_log_normal2_double (curandStatePhilox4_32_10_t *state, double mean, double stddev)

Return two log-normally distributed doubles from an Philox4_32_10 generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed double4 where each element is from a distribution with mean `mean` and standard deviation `stddev`

Description

Return two log-normally distributed doubles derived from a normal distribution with mean `mean` and standard deviation `stddev` from the Philox4_32_10 generator in `state`, increment position of generator by four.

The implementation uses a Box-Muller transform to generate two normally distributed results, and transforms them to log-normal distribution,.

__device__ double2 curand_log_normal2_double (curandStateXORWOW_t *state, double mean, double stddev)

Return two log-normally distributed doubles from an XORWOW generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed double2 where each element is from a distribution with mean `mean` and standard deviation `stddev`

Description

Return two log-normally distributed doubles derived from a normal distribution with mean `mean` and standard deviation `stddev` from the XORWOW generator in `state`, increment position of generator by two.

The implementation uses a Box-Muller transform to generate two normally distributed results, and transforms them to log-normal distribution,.

__device__ float4 curand_log_normal4 (curandStatePhilox4_32_10_t *state, float mean, float stddev)

Return four normally distributed floats from an Philox4_32_10 generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed float4 where each element is from a distribution with mean `mean` and standard deviation `stddev`

Description

Return four log-normally distributed floats derived from a normal distribution with mean `mean` and standard deviation `stddev` from the Philox4_32_10 generator in `state`, increment position of generator by four.

The implementation uses a Box-Muller transform to generate two normally distributed results, then transforms them to log-normal.

__device__ double curand_log_normal_double (curandStateScrambledSobol64_t *state, double mean, double stddev)

Return a log-normally distributed double from a scrambled Sobol64 generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed double with mean `mean` and standard deviation `stddev`

Description

Return a single normally distributed double derived from a normal distribution with mean `mean` and standard deviation `stddev` from the scrambled Sobol64 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ double curand_log_normal_double (curandStateSobol64_t *state, double mean, double stddev)

Return a log-normally distributed double from a Sobol64 generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed double with mean `mean` and standard deviation `stddev`

Description

Return a single normally distributed double derived from a normal distribution with mean `mean` and standard deviation `stddev` from the Sobol64 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ double curand_log_normal_double (curandStateScrambledSobol32_t *state, double mean, double stddev)

Return a log-normally distributed double from a scrambled Sobol32 generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed double with mean `mean` and standard deviation `stddev`

Description

Return a single log-normally distributed double derived from a normal distribution with mean `mean` and standard deviation `stddev` from the scrambled Sobol32 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results, and transforms them into log-normal distribution.

__device__ double curand_log_normal_double (curandStateSobol32_t *state, double mean, double stddev)

Return a log-normally distributed double from a Sobol32 generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed double with mean `mean` and standard deviation `stddev`

Description

Return a single log-normally distributed double derived from a normal distribution with mean `mean` and standard deviation `stddev` from the Sobol32 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results, and transforms them into log-normal distribution.

__device__ double curand_log_normal_double (curandStateMtgp32_t *state, double mean, double stddev)

Return a log-normally distributed double from an MTGP32 generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed double with mean `mean` and standard deviation `stddev`

Description

Return a single log-normally distributed double derived from a normal distribution with mean `mean` and standard deviation `stddev` from the MTGP32 generator in `state`, increment position of generator.

The implementation uses the inverse cumulative distribution function to generate normally distributed results, and transforms them into log-normal distribution.

__device__ double curand_log_normal_double (curandStateMRG32k3a_t *state, double mean, double stddev)

Return a log-normally distributed double from an MRG32k3a generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed double with mean `mean` and standard deviation `stddev`

Description

Return a single normally distributed double derived from a normal distribution with mean `mean` and standard deviation `stddev` from the MRG32k3a generator in `state`, increment position of generator.

The implementation uses a Box-Muller transform to generate two normally distributed results, transforms them to log-normal distribution, then returns them one at a time. See curand_log_normal2_double() for a more efficient version that returns both results at once.

__device__ double curand_log_normal_double (curandStatePhilox4_32_10_t *state, double mean, double stddev)

Return a log-normally distributed double from an Philox4_32_10 generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed double with mean `mean` and standard deviation `stddev`

Description

Return a single normally distributed double derived from a normal distribution with mean `mean` and standard deviation `stddev` from the Philox4_32_10 generator in `state`, increment position of generator.

The implementation uses a Box-Muller transform to generate two normally distributed results, transforms them to log-normal distribution, then returns them one at a time. See curand_log_normal2_double() for a more efficient version that returns both results at once.

__device__ double curand_log_normal_double (curandStateXORWOW_t *state, double mean, double stddev)

Return a log-normally distributed double from an XORWOW generator.

Parameters

state
- Pointer to state to update

mean
- Mean of the related normal distribution

stddev
- Standard deviation of the related normal distribution

Returns

Log-normally distributed double with mean `mean` and standard deviation `stddev`

Description

Return a single normally distributed double derived from a normal distribution with mean `mean` and standard deviation `stddev` from the XORWOW generator in `state`, increment position of generator.

The implementation uses a Box-Muller transform to generate two normally distributed results, transforms them to log-normal distribution, then returns them one at a time. See curand_log_normal2_double() for a more efficient version that returns both results at once.

__device__ float curand_mtgp32_single (curandStateMtgp32_t *state)

Return a uniformly distributed float from a mtgp32 generator.

Parameters

state
- Pointer to state to update

Returns

uniformly distributed float between `0.0f` and `1.0f`

Description

Return a uniformly distributed float between `0.0f` and `1.0f` from the mtgp32 generator in `state`, increment position of generator. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

Note: This alternate derivation of a uniform float is provided for completeness with the original source

__device__ float curand_mtgp32_single_specific (curandStateMtgp32_t *state, unsigned char index, unsigned char n)

Return a uniformly distributed float from a specific position in a mtgp32 generator.

Parameters

state
- Pointer to state to update

index
- Index (0..255) of the position within the state to draw from and update

n
- The total number of postions in this state that are being updated by this invocation

Returns

uniformly distributed float between `0.0f` and `1.0f`

Description

Return a uniformly distributed float between `0.0f` and `1.0f` from position `index` of the mtgp32 generator in `state`, and increment position of generator by `n` positions, which must

be the total number of positions upddated in the state by the thread block, for this invocation. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

Note 1: Thread indices must range from 0...`n` - 1. The number of positions updated may not exceed 256. A thread block may update more than one state, but a given state may not be updated by more than one thread block.

Note 2: This alternate derivation of a uniform float is provided for completeness with the original source

__device__ unsigned int curand_mtgp32_specific (curandStateMtgp32_t *state, unsigned char index, unsigned char n)

Return 32-bits of pseudorandomness from a specific position in a mtgp32 generator.

Parameters

state

- Pointer to state to update

index

- Index (0..255) of the position within the state to draw from and update

n

- The total number of postions in this state that are being updated by this invocation

Returns

32-bits of pseudorandomness as an unsigned int, all bits valid to use.

Description

Return 32-bits of pseudorandomness from position `index` of the mtgp32 generator in `state`, increment position of generator by `n` positions, which must be the total number of positions upddated in the state by the thread block, for this invocation.

Note : Thread indices must range from 0... n - 1. The number of positions updated may not exceed 256. A thread block may update more than one state, but a given state may not be updated by more than one thread block.

__device__ float curand_normal (curandStateScrambledSobol64_t *state)

Return a normally distributed float from a scrambled Sobol64 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float with mean `0.0f` and standard deviation `1.0f`

Description

Return a single normally distributed float with mean `0.0f` and standard deviation `1.0f` from the scrambled Sobol64 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ float curand_normal (curandStateSobol64_t *state)

Return a normally distributed float from a Sobol64 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float with mean `0.0f` and standard deviation `1.0f`

Description

Return a single normally distributed float with mean `0.0f` and standard deviation `1.0f` from the Sobol64 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ float curand_normal (curandStateScrambledSobol32_t *state)

Return a normally distributed float from a scrambled Sobol32 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float with mean `0.0f` and standard deviation `1.0f`

Description

Return a single normally distributed float with mean `0.0f` and standard deviation `1.0f` from the scrambled Sobol32 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ float curand_normal (curandStateSobol32_t *state)

Return a normally distributed float from a Sobol32 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float with mean `0.0f` and standard deviation `1.0f`

Description

Return a single normally distributed float with mean `0.0f` and standard deviation `1.0f` from the Sobol32 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ float curand_normal (curandStateMtgp32_t *state)

Return a normally distributed float from a MTGP32 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float with mean `0.0f` and standard deviation `1.0f`

Description

Return a single normally distributed float with mean `0.0f` and standard deviation `1.0f` from the MTGP32 generator in `state`, increment position of generator.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ float curand_normal (curandStateMRG32k3a_t *state)

Return a normally distributed float from an MRG32k3a generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float with mean `0.0f` and standard deviation `1.0f`

Description

Return a single normally distributed float with mean `0.0f` and standard deviation `1.0f` from the MRG32k3a generator in `state`, increment position of generator by one.

The implementation uses a Box-Muller transform to generate two normally distributed results, then returns them one at a time. See curand_normal2() for a more efficient version that returns both results at once.

__device__ float curand_normal (curandStatePhilox4_32_10_t *state)

Return a normally distributed float from an Philox4_32_10 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float with mean `0.0f` and standard deviation `1.0f`

Description

Return a single normally distributed float with mean `0.0f` and standard deviation `1.0f` from the Philox4_32_10 generator in `state`, increment position of generator by one.

The implementation uses a Box-Muller transform to generate two normally distributed results, then returns them one at a time. See curand_normal2() for a more efficient version that returns both results at once.

__device__ float curand_normal (curandStateXORWOW_t *state)

Return a normally distributed float from an XORWOW generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float with mean `0.0f` and standard deviation `1.0f`

Description

Return a single normally distributed float with mean `0.0f` and standard deviation `1.0f` from the XORWOW generator in `state`, increment position of generator by one.

The implementation uses a Box-Muller transform to generate two normally distributed results, then returns them one at a time. See curand_normal2() for a more efficient version that returns both results at once.

__device__ float2 curand_normal2 (curandStateMRG32k3a_t *state)

Return two normally distributed floats from an MRG32k3a generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float2 where each element is from a distribution with mean `0.0f` and standard deviation `1.0f`

Description

Return two normally distributed floats with mean `0.0f` and standard deviation `1.0f` from the MRG32k3a generator in `state`, increment position of generator by two.

The implementation uses a Box-Muller transform to generate two normally distributed results.

__device__ float2 curand_normal2 (curandStatePhilox4_32_10_t *state)

Return two normally distributed floats from an Philox4_32_10 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float2 where each element is from a distribution with mean `0.0f` and standard deviation `1.0f`

Description

Return two normally distributed floats with mean `0.0f` and standard deviation `1.0f` from the Philox4_32_10 generator in `state`, increment position of generator by two.

The implementation uses a Box-Muller transform to generate two normally distributed results.

__device__ float2 curand_normal2 (curandStateXORWOW_t *state)

Return two normally distributed floats from an XORWOW generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float2 where each element is from a distribution with mean `0.0f` and standard deviation `1.0f`

Description

Return two normally distributed floats with mean `0.0f` and standard deviation `1.0f` from the XORWOW generator in `state`, increment position of generator by two.

The implementation uses a Box-Muller transform to generate two normally distributed results.

__device__ double2 curand_normal2_double (curandStateMRG32k3a_t *state)

Return two normally distributed doubles from an MRG32k3a generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed double2 where each element is from a distribution with mean `0.0` and standard deviation `1.0`

Description

Return two normally distributed doubles with mean `0.0` and standard deviation `1.0` from the MRG32k3a generator in `state`, increment position of generator.

The implementation uses a Box-Muller transform to generate two normally distributed results.

__device__ double2 curand_normal2_double (curandStatePhilox4_32_10_t *state)

Return two normally distributed doubles from an Philox4_32_10 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed double2 where each element is from a distribution with mean `0.0` and standard deviation `1.0`

Description

Return two normally distributed doubles with mean `0.0` and standard deviation `1.0` from the Philox4_32_10 generator in `state`, increment position of generator by 2.

The implementation uses a Box-Muller transform to generate two normally distributed results.

__device__ double2 curand_normal2_double (curandStateXORWOW_t *state)

Return two normally distributed doubles from an XORWOW generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed double2 where each element is from a distribution with mean `0.0` and standard deviation `1.0`

Description

Return two normally distributed doubles with mean `0.0` and standard deviation `1.0` from the XORWOW generator in `state`, increment position of generator by 2.

The implementation uses a Box-Muller transform to generate two normally distributed results.

__device__ float4 curand_normal4 (curandStatePhilox4_32_10_t *state)

Return four normally distributed floats from an Philox4_32_10 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed float2 where each element is from a distribution with mean `0.0f` and standard deviation `1.0f`

Description

Return four normally distributed floats with mean `0.0f` and standard deviation `1.0f` from the Philox4_32_10 generator in `state`, increment position of generator by four.

The implementation uses a Box-Muller transform to generate two normally distributed results.

__device__ double curand_normal_double (curandStateScrambledSobol64_t *state)

Return a normally distributed double from a scrambled Sobol64 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed double with mean `0.0` and standard deviation `1.0`

Description

Return a single normally distributed double with mean `0.0` and standard deviation `1.0` from the scrambled Sobol64 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ double curand_normal_double (curandStateSobol64_t *state)

Return a normally distributed double from a Sobol64 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed double with mean `0.0` and standard deviation `1.0`

Description

Return a single normally distributed double with mean `0.0` and standard deviation `1.0` from the Sobol64 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ double curand_normal_double (curandStateScrambledSobol32_t *state)

Return a normally distributed double from a scrambled Sobol32 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed double with mean `0.0` and standard deviation `1.0`

Description

Return a single normally distributed double with mean `0.0` and standard deviation `1.0` from the scrambled Sobol32 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ double curand_normal_double (curandStateSobol32_t *state)

Return a normally distributed double from an Sobol32 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed double with mean `0.0` and standard deviation `1.0`

Description

Return a single normally distributed double with mean `0.0` and standard deviation `1.0` from the Sobol32 generator in `state`, increment position of generator by one.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ double curand_normal_double (curandStateMtgp32_t *state)

Return a normally distributed double from an MTGP32 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed double with mean `0.0` and standard deviation `1.0`

Description

Return a single normally distributed double with mean `0.0` and standard deviation `1.0` from the MTGP32 generator in `state`, increment position of generator.

The implementation uses the inverse cumulative distribution function to generate normally distributed results.

__device__ double curand_normal_double (curandStateMRG32k3a_t *state)

Return a normally distributed double from an MRG32k3a generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed double with mean `0.0` and standard deviation `1.0`

Description

Return a single normally distributed double with mean `0.0` and standard deviation `1.0` from the XORWOW generator in `state`, increment position of generator.

The implementation uses a Box-Muller transform to generate two normally distributed results, then returns them one at a time. See curand_normal2_double() for a more efficient version that returns both results at once.

__device__ double curand_normal_double (curandStatePhilox4_32_10_t *state)

Return a normally distributed double from an Philox4_32_10 generator.

Parameters

state

- Pointer to state to update

Returns

Normally distributed double with mean `0.0` and standard deviation `1.0`

Description

Return a single normally distributed double with mean `0.0` and standard deviation `1.0` from the Philox4_32_10 generator in `state`, increment position of generator.

The implementation uses a Box-Muller transform to generate two normally distributed results, then returns them one at a time. See curand_normal2_double() for a more efficient version that returns both results at once.

__device__ double curand_normal_double (curandStateXORWOW_t *state)

Return a normally distributed double from an XORWOW generator.

Parameters

state
- Pointer to state to update

Returns

Normally distributed double with mean `0.0` and standard deviation `1.0`

Description

Return a single normally distributed double with mean `0.0` and standard deviation `1.0` from the XORWOW generator in `state`, increment position of generator.

The implementation uses a Box-Muller transform to generate two normally distributed results, then returns them one at a time. See curand_normal2_double() for a more efficient version that returns both results at once.

__device__ unsigned int curand_poisson (curandStateScrambledSobol64_t *state, double lambda)

Return a Poisson-distributed unsigned int from a scrambled Sobol64 generator.

Parameters

state
- Pointer to state to update

lambda
- Lambda of the Poisson distribution

Returns

Poisson-distributed unsigned int with lambda `lambda`

Description

Return a single unsigned int from a Poisson distribution with lambda `lambda` from the scrambled Sobol64 generator in `state`, increment position of generator by one.

__device__ unsigned int curand_poisson (curandStateSobol64_t *state, double lambda)

Return a Poisson-distributed unsigned int from a Sobol64 generator.

Parameters

state

- Pointer to state to update

lambda

- Lambda of the Poisson distribution

Returns

Poisson-distributed unsigned int with lambda `lambda`

Description

Return a single unsigned int from a Poisson distribution with lambda `lambda` from the Sobol64 generator in `state`, increment position of generator by one.

__device__ unsigned int curand_poisson (curandStateScrambledSobol32_t *state, double lambda)

Return a Poisson-distributed unsigned int from a scrambled Sobol32 generator.

Parameters

state

- Pointer to state to update

lambda

- Lambda of the Poisson distribution

Returns

Poisson-distributed unsigned int with lambda `lambda`

Description

Return a single unsigned int from a Poisson distribution with lambda `lambda` from the scrambled Sobol32 generator in `state`, increment the position of the generator by one.

__device__ unsigned int curand_poisson (curandStateSobol32_t *state, double lambda)

Return a Poisson-distributed unsigned int from a Sobol32 generator.

Parameters

state

- Pointer to state to update

lambda

- Lambda of the Poisson distribution

Returns

Poisson-distributed unsigned int with lambda `lambda`

Description

Return a single unsigned int from a Poisson distribution with lambda `lambda` from the Sobol32 generator in `state`, increment the position of the generator by one.

__device__ unsigned int curand_poisson (curandStateMtgp32_t *state, double lambda)

Return a Poisson-distributed unsigned int from a MTGP32 generator.

Parameters

state

- Pointer to state to update

lambda

- Lambda of the Poisson distribution

Returns

Poisson-distributed unsigned int with lambda `lambda`

Description

Return a single int from a Poisson distribution with lambda `lambda` from the MTGP32 generator in `state`, increment the position of the generator by one.

__device__ unsigned int curand_poisson (curandStateMRG32k3a_t *state, double lambda)

Return a Poisson-distributed unsigned int from a MRG32k3A generator.

Parameters

state

- Pointer to state to update

lambda

- Lambda of the Poisson distribution

Returns

Poisson-distributed unsigned int with lambda `lambda`

Description

Return a single unsigned int from a Poisson distribution with lambda `lambda` from the MRG32k3a generator in `state`, increment the position of the generator by a variable amount, depending on the algorithm used.

__device__ unsigned int curand_poisson (curandStatePhilox4_32_10_t *state, double lambda)

Return a Poisson-distributed unsigned int from a Philox4_32_10 generator.

Parameters

state

- Pointer to state to update

lambda

- Lambda of the Poisson distribution

Returns

Poisson-distributed unsigned int with lambda `lambda`

Description

Return a single unsigned int from a Poisson distribution with lambda `lambda` from the Philox4_32_10 generator in `state`, increment the position of the generator by a variable amount, depending on the algorithm used.

__device__ unsigned int curand_poisson (curandStateXORWOW_t *state, double lambda)

Return a Poisson-distributed unsigned int from a XORWOW generator.

Parameters

state

- Pointer to state to update

lambda

- Lambda of the Poisson distribution

Returns

Poisson-distributed unsigned int with lambda `lambda`

Description

Return a single unsigned int from a Poisson distribution with lambda `lambda` from the XORWOW generator in `state`, increment the position of the generator by a variable amount, depending on the algorithm used.

__device__ uint4 curand_poisson4 (curandStatePhilox4_32_10_t *state, double lambda)

Return four Poisson-distributed unsigned ints from a Philox4_32_10 generator.

Parameters

state

- Pointer to state to update

lambda

- Lambda of the Poisson distribution

Returns

Poisson-distributed unsigned int with lambda `lambda`

Description

Return a four unsigned ints from a Poisson distribution with lambda `lambda` from the Philox4_32_10 generator in `state`, increment the position of the generator by a variable amount, depending on the algorithm used.

__device__ float curand_uniform (curandStateScrambledSobol64_t *state)

Return a uniformly distributed float from a scrambled Sobol64 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed float between `0.0f` and `1.0f`

Description

Return a uniformly distributed float between `0.0f` and `1.0f` from the scrambled Sobol64 generator in `state`, increment position of generator. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

The implementation is guaranteed to use a single call to `curand()`.

__device__ float curand_uniform (curandStateSobol64_t *state)

Return a uniformly distributed float from a Sobol64 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed float between `0.0f` and `1.0f`

Description

Return a uniformly distributed float between `0.0f` and `1.0f` from the Sobol64 generator in `state`, increment position of generator. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

The implementation is guaranteed to use a single call to `curand()`.

__device__ float curand_uniform (curandStateScrambledSobol32_t *state)

Return a uniformly distributed float from a scrambled Sobol32 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed float between `0.0f` and `1.0f`

Description

Return a uniformly distributed float between `0.0f` and `1.0f` from the scrambled Sobol32 generator in `state`, increment position of generator. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

The implementation is guaranteed to use a single call to `curand()`.

__device__ float curand_uniform (curandStateSobol32_t *state)

Return a uniformly distributed float from a Sobol32 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed float between `0.0f` and `1.0f`

Description

Return a uniformly distributed float between `0.0f` and `1.0f` from the Sobol32 generator in `state`, increment position of generator. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

The implementation is guaranteed to use a single call to `curand()`.

__device__ float curand_uniform (curandStateMtgp32_t *state)

Return a uniformly distributed float from a MTGP32 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed float between `0.0f` and `1.0f`

Description

Return a uniformly distributed float between `0.0f` and `1.0f` from the MTGP32 generator in `state`, increment position of generator. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

__device__ float curand_uniform (curandStatePhilox4_32_10_t *state)

Return a uniformly distributed float from a Philox4_32_10 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed float between `0.0` and `1.0`

Description

Return a uniformly distributed float between `0.0f` and `1.0f` from the Philox4_32_10 generator in `state`, increment position of generator. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

__device__ float curand_uniform (curandStateMRG32k3a_t *state)

Return a uniformly distributed float from an MRG32k3a generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed float between `0.0f` and `1.0f`

Description

Return a uniformly distributed float between `0.0f` and `1.0f` from the MRG32k3a generator in `state`, increment position of generator. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

The implementation returns up to 23 bits of mantissa, with the minimum return value latexInlineFormula: =$ 2^{-32} $

__device__ float curand_uniform (curandStateXORWOW_t *state)

Return a uniformly distributed float from an XORWOW generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed float between `0.0f` and `1.0f`

Description

Return a uniformly distributed float between `0.0f` and `1.0f` from the XORWOW generator in `state`, increment position of generator. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

The implementation may use any number of calls to `curand()` to get enough random bits to create the return value. The current implementation uses one call.

__device__ double2 curand_uniform2_double (curandStatePhilox4_32_10_t *state)

Return a uniformly distributed tuple of 2 doubles from an Philox4_32_10 generator.

Parameters

state

- Pointer to state to update

Returns

2 uniformly distributed doubles between `0.0` and `1.0`

Description

Return a uniformly distributed 2 doubles (double4) between `0.0` and `1.0` from the Philox4_32_10 generator in `state`, increment position of generator by 4. Output range excludes `0.0` but includes `1.0`. Denormalized floating point outputs are never returned.

__device__ float4 curand_uniform4 (curandStatePhilox4_32_10_t *state)

Return a uniformly distributed tuple of 4 floats from a Philox4_32_10 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed float between `0.0` and `1.0`

Description

Return a uniformly distributed 4 floats between `0.0f` and `1.0f` from the Philox4_32_10 generator in `state`, increment position of generator by 4. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

__device__ double curand_uniform_double (curandStateScrambledSobol64_t *state)

Return a uniformly distributed double from a scrambled Sobol64 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed double between `0.0` and `1.0`

Description

Return a uniformly distributed double between `0.0` and `1.0` from the scrambled Sobol64 generator in `state`, increment position of generator. Output range excludes `0.0` but includes `1.0`. Denormalized floating point outputs are never returned.

The implementation is guaranteed to use a single call to `curand()` to preserve the quasirandom properties of the sequence.

__device__ double curand_uniform_double (curandStateSobol64_t *state)

Return a uniformly distributed double from a Sobol64 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed double between `0.0` and `1.0`

Description

Return a uniformly distributed double between `0.0` and `1.0` from the Sobol64 generator in `state`, increment position of generator. Output range excludes `0.0` but includes `1.0`. Denormalized floating point outputs are never returned.

The implementation is guaranteed to use a single call to `curand()` to preserve the quasirandom properties of the sequence.

__device__ double curand_uniform_double (curandStateScrambledSobol32_t *state)

Return a uniformly distributed double from a scrambled Sobol32 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed double between `0.0` and `1.0`

Description

Return a uniformly distributed double between `0.0` and `1.0` from the scrambled Sobol32 generator in `state`, increment position of generator. Output range excludes `0.0` but includes `1.0`. Denormalized floating point outputs are never returned.

The implementation is guaranteed to use a single call to `curand()` to preserve the quasirandom properties of the sequence.

Note that the implementation uses only 32 random bits to generate a single double precision value.

__device__ double curand_uniform_double (curandStateSobol32_t *state)

Return a uniformly distributed double from a Sobol32 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed double between `0.0` and `1.0`

Description

Return a uniformly distributed double between `0.0` and `1.0` from the Sobol32 generator in `state`, increment position of generator. Output range excludes `0.0` but includes `1.0`. Denormalized floating point outputs are never returned.

The implementation is guaranteed to use a single call to `curand()` to preserve the quasirandom properties of the sequence.

Note that the implementation uses only 32 random bits to generate a single double precision value.

__device__ double curand_uniform_double (curandStatePhilox4_32_10_t *state)

Return a uniformly distributed double from a Philox4_32_10 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed double between `0.0f` and `1.0f`

Description

Return a uniformly distributed double between `0.0f` and `1.0f` from the Philox4_32_10 generator in `state`, increment position of generator. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

Note that the implementation uses only 32 random bits to generate a single double precision value.

`curand_uniform2_double()` is recommended for higher quality uniformly distributed double precision values.

__device__ double curand_uniform_double (curandStateMtgp32_t *state)

Return a uniformly distributed double from a MTGP32 generator.

Parameters

state

- Pointer to state to update

Returns

uniformly distributed double between `0.0f` and `1.0f`

Description

Return a uniformly distributed double between `0.0f` and `1.0f` from the MTGP32 generator in `state`, increment position of generator. Output range excludes `0.0f` but includes `1.0f`. Denormalized floating point outputs are never returned.

Note that the implementation uses only 32 random bits to generate a single double precision value.

__device__ double curand_uniform_double (curandStateMRG32k3a_t *state)

Return a uniformly distributed double from an MRG32k3a generator.

Parameters

state
- Pointer to state to update

Returns

uniformly distributed double between `0.0` and `1.0`

Description

Return a uniformly distributed double between `0.0` and `1.0` from the MRG32k3a generator in `state`, increment position of generator. Output range excludes `0.0` but includes `1.0`. Denormalized floating point outputs are never returned.

Note the implementation returns at most 32 random bits of mantissa as outlined in the seminal paper by L'Ecuyer.

__device__ double curand_uniform_double (curandStateXORWOW_t *state)

Return a uniformly distributed double from an XORWOW generator.

Parameters

state
- Pointer to state to update

Returns

uniformly distributed double between `0.0` and `1.0`

Description

Return a uniformly distributed double between `0.0` and `1.0` from the XORWOW generator in `state`, increment position of generator. Output range excludes `0.0` but includes `1.0`. Denormalized floating point outputs are never returned.

The implementation may use any number of calls to `curand()` to get enough random bits to create the return value. The current implementation uses exactly two calls.

__host____forceinline__ curandStatus_t curandMakeMTGP32Constants (const mtgp32_params_fast_t params[], mtgp32_kernel_params_t *p)

Set up constant parameters for the mtgp32 generator.

Parameters

params

- Pointer to an array of type mtgp32_params_fast_t in host memory

p

- pointer to a structure of type mtgp32_kernel_params_t in device memory.

Returns

- CURAND_STATUS_ALLOCATION_FAILED if host memory could not be allocated
- CURAND_STATUS_INITIALIZATION_FAILED if the copy to device memory failed
- CURAND_STATUS_SUCCESS otherwise

Description

This host-side helper function re-organizes CURAND_NUM_MTGP32_PARAMS sets of generator parameters for use by kernel functions and copies the result to the specified location in device memory.

__host____forceinline__ curandStatus_t CURANDAPI curandMakeMTGP32KernelState (curandStateMtgp32_t *s, mtgp32_params_fast_t params[], mtgp32_kernel_params_t *k, int n, unsigned long long seed)

Set up initial states for the mtgp32 generator.

Parameters

s

- pointer to an array of states in device memory

params

- Pointer to an array of type mtgp32_params_fast_t in host memory

k
- pointer to a structure of type mtgp32_kernel_params_t in device memory

n
- number of parameter sets/states to initialize

seed
- seed value

Returns

- CURAND_STATUS_ALLOCATION_FAILED if host memory state could not be allocated
- CURAND_STATUS_INITIALIZATION_FAILED if the copy to device memory failed
- CURAND_STATUS_SUCCESS otherwise

Description

This host-side helper function initializes a number of states (one parameter set per state) for an mtgp32 generator. To accomplish this it allocates a state array in host memory, initializes that array, and copies the result to device memory.

template < typename T > __device__skipahead (unsigned long long n, T state)

Update Sobol64 state to skip `n` elements.

Parameters

n
- Number of elements to skip

state
- Pointer to state to update

Description

Update the Sobol64 state in `state` to skip ahead `n` elements.

All values of `n` are valid.

template < typename T > __device__skipahead (unsigned int n, T state)

Update Sobol32 state to skip `n` elements.

Parameters

n
- Number of elements to skip

state
- Pointer to state to update

Description

Update the Sobol32 state in `state` to skip ahead `n` elements.

All values of `n` are valid.

__device__ void skipahead (unsigned long long n, curandStateMRG32k3a_t *state)

Update MRG32k3a state to skip `n` elements.

Parameters

n
- Number of elements to skip

state
- Pointer to state to update

Description

Update the MRG32k3a state in `state` to skip ahead `n` elements.

All values of `n` are valid. Large values require more computation and so will take more time to complete.

__device__ void skipahead (unsigned long long n, curandStatePhilox4_32_10_t *state)

Update Philox4_32_10 state to skip `n` elements.

Parameters

n
- Number of elements to skip

state
- Pointer to state to update

Description

Update the Philox4_32_10 state in `state` to skip ahead `n` elements.

All values of `n` are valid.

__device__ void skipahead (unsigned long long n, curandStateXORWOW_t *state)

Update XORWOW state to skip `n` elements.

Parameters

n
- Number of elements to skip

state
- Pointer to state to update

Description

Update the XORWOW state in `state` to skip ahead `n` elements.

All values of `n` are valid. Large values require more computation and so will take more time to complete.

__device__ void skipahead_sequence (unsigned long long n, curandStateMRG32k3a_t *state)

Update MRG32k3a state to skip ahead `n` sequences.

Parameters

n
- Number of sequences to skip

state
- Pointer to state to update

Description

Update the MRG32k3a state in `state` to skip ahead `n` sequences. Each sequence is 2^{127} elements long, so this means the function will skip ahead 2^{127} * n elements.

All values of `n` are valid. Large values require more computation and so will take more time to complete.

__device__ void skipahead_sequence (unsigned long long n, curandStatePhilox4_32_10_t *state)

Update Philox4_32_10 state to skip ahead `n` subsequences.

Parameters

n

- Number of subsequences to skip

state

- Pointer to state to update

Description

Update the Philox4_32_10 state in `state` to skip ahead `n` subsequences. Each subsequence is 2^{66} elements long, so this means the function will skip ahead 2^{66} * n elements.

All values of `n` are valid.

__device__ void skipahead_sequence (unsigned long long n, curandStateXORWOW_t *state)

Update XORWOW state to skip ahead `n` subsequences.

Parameters

n

- Number of subsequences to skip

state

- Pointer to state to update

Description

Update the XORWOW state in `state` to skip ahead `n` subsequences. Each subsequence is 2^{67} elements long, so this means the function will skip ahead 2^{67} * n elements.

All values of `n` are valid. Large values require more computation and so will take more time to complete.

__device__ void skipahead_subsequence (unsigned long long n, curandStateMRG32k3a_t *state)

Update MRG32k3a state to skip ahead `n` subsequences.

Parameters

n

- Number of subsequences to skip

state

- Pointer to state to update

Description

Update the MRG32k3a state in `state` to skip ahead `n` subsequences. Each subsequence is 2^{127}

2^{76} elements long, so this means the function will skip ahead 2^{67} * n elements.

Valid values of `n` are 0 to 2^{51} . Note `n` will be masked to 51 bits

Appendix A. Bibliography

[1] Mutsuo Saito. A Variant of Mersenne Twister Suitable for Graphic Processors. *arXiv:1005.4973v2 [cs.MS]*, Jun 2010.

[2] S. Joe and F. Y. Kuo. Remark on Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software*, 29:49-57, March 2003.

[3] Jiri Matousek. Journal of Complexity. *ACM Transactions on Mathematical Software*, 14(4):527-556, December 1998.

[4] Art B. Owen. Local Antithetic Sampling with Scrambled Nets. *The Annals of Statistics*, 36(5):2319-2343, 2008.

[5] George Marsaglia. Xorshift RNGs. *Journal of Statistical Software*, 8(14), 2003. Available at http://www.jstatsoft.org/v08/i14/paper.

[6] Pierre L'Ecuyer and Richard Simard. TestU01: A C library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, 33(4), August 2007. Available at http://www.iro.umontreal.ca/~lecuyer/myftp/papers/testu01.pdf.

[7] Andrew Rukhin and Juan Soto and James Nechvatal and Miles Smid and Elaine Barker and Stefan Leigh and Mark Levenson and Mark Vangel and David Banks and Alan Heckert and James Dray and San Vo. "A Statistical Test Suite for the Validation of Random Number Generators and Pseudorandom Number Generators for Cryptographic Applications. Special Publication 800-22 Revision 1a, National Institute of Standards and Technology, April 2010. http://csrc.nist.gov/groups/ST/toolkit/rng/index.html.

[8] M. Matsumoto and T. Nishimura. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3-30, January 1988.

[9] Pierre L'Ecuyer. Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators. *Operations Research*, 47(1), Jan-Feb 1999.

[10] Pierre L'Ecuyer and Richard Simard and E. Jack Chen and W. David Kelton. An Object-Oriented Random-Number Package with Many Long Streams and Substreams. *Operations Research*, 50(6), Nov-Dec 2002.

[11] Karl Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine*, 50(302):157-175, July 1900.

[12] R. L. Placket. Karl Pearson and the chi-squared test. *International Statistics Review*, 51:59-72, 1983.

[13] Carlos M. Jarque and Anil K. Bera. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, 6(3):255-259, 1980.

[14] A. Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *G. Inst. Ital. Attuari*, 4(83), 1933.

[15] Frank J. Massey. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68-78, 1951.

[16] T. W. Anderson and D. A. Darling. Asymptotic theory of certain "goodness-of-fit" criteria based on stochastic processes. *Annals of Mathematical Statistics*, 23(2):193-212, 1952.

[17] John. K. Salmon, Mark A. Moraes, Ron O. Dror, and David E. Shaw. Parallel Random Numbers: As Easy as 1, 2, 3 *D.E Shaw Research, New York, NY 10036, USA*, 2011.

[18] P. Trędak, C. Woolley. Efficient implementation of Mersenne Twister MT19937 Random Number Generator on the GPU *GPU Technology Conference*, 2013.

Appendix B. Acknowledgements

NVIDIA would like to thank the following individuals and institutions for their contributions:

- Portions of the MTGP32 (Mersenne Twister for GPU) library routines were written by Mutsuo Saito and Makoto Matsumoto.
- Portions of the PHILOX4x32 library routines were developed by D. E. Shaw Research.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Trademarks

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
https://www.nvidia.com