My Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 random_functions.cpp File Reference

```
#include "random_functions.hpp"
```

Functions

- double random_number01 ()
- double random_number_01_GSL (gsl_rng ∗r)
- double rejection_sampl_algo (gsl_rng ∗r)
- double normal_cdf (double x)
- double normal_inverse_cdf (double x)
- std::vector< double > ∗ mueller_box_algo (double mu, double sigma)
- double sigma_naive (int N, std::vector< double > ∗sample)
- double sigma_algorithm (std::vector< double > ∗sample, int N)
- std::vector< double > ∗ wiener_process (gsl_rng ∗r, double T, double delta_t)
- std::vector< double > ∗ brownian_motion (gsl_rng ∗r, double T, double delta_t, std::vector< double > ∗w, double s0, double mu, double sigma)

2.1.1 Function Documentation

2.1.1.1 brownian_motion()

```
std::vector<double>∗ brownian_motion (
            gsl_rng ∗ r,
            double T,
            double delta_t,
            std::vector< double > ∗ w,
            double s0,
            double mu,
            double sigma )
```

Simulates brownian_motion path for the given values of wiener process

Parameters

r	Pointer to the gsl_rng object for generating standard normal distributed numbers
T	Time period of simulated process
delta↩ _t	Step of discretisation
w	Pointer to the vector with values of wiener process ar discretisation points
s0	Value of brownian_motion at time = 0
mu	Drift
sigma	Volatility

Returns

Pointer to the vector of values at discretisation points

2.1.1.2 mueller_box_algo()

```
std::vector<double>* mueller_box_algo (
             double mu,
             double sigma )
```

Box Mueller Algorithm

Parameters

mu	It is mean for the normal distribution
sigma	It is sigma for the normal distribution

Returns

It returns pointer to the vector with 2 normal distributed values

2.1.1.3 normal_cdf()

```
double normal_cdf (
             double x )
```

Moro's algorithm is an approximation to the c.d.f. of the standard normal distribution with an accurancy of 8 digits

Parameters

x	double value, point at which $p(x)$ will be calculated

Returns

If everything worked fine returns $p(x)$

2.1.1.4 normal_inverse_cdf()

```
double normal_inverse_cdf (
             double x )
```

Calculates the inverse CDF of the standard normal distribution for a parameter x.

Parameters

x	The parameter for the inverse CDF of the standard normal distribution.

Returns

The value of the inverse CDF at x.

2.1.1.5 random_number01()

```
double random_number01 ( )
```

Draws a random number bewtween $[0, 1]$ via rand.

Returns

The drawn random number.

2.1.1.6 random_number_01_GSL()

```
double random_number_01_GSL (
             gsl_rng * r )
```

Draws a random number in $[0, 1]$ via the gsl.

Parameters

r	A pointer to the random number generator which is used.

Returns

The drawn random number.

2.1.1.7 rejection_sampl_algo()

```
double rejection_sampl_algo (
            gsl_rng * r )
```

Main function for random number evaluation. A first random number x_1 is drawn via rand. A second random number x_2 is drawn via gsl_rng_uniform.

Furthermore, an array of 10 doubles is allocated. However, someone seems to have forgotten to free the allocated space again in the end...

Parameters

argc	The number of arguments provided.
argv	An array of arguments (argv[0] is the name of the executable).

Returns

If everything worked fine, 0 is returned. Rejection Sampling Algorithm

The algorithm produces standard normal distributed value

Parameters

r	a pointer to gsl_rng object

Returns

x returns double value, which is standard normal distributed

interval bounds $[a, b]$, s.t. $\int_a^b p\left(x\right) dx = 1$, p(x) density for a standard normal distribution

2.1.1.8 sigma_algorithm()

```
double sigma_algorithm (
            std::vector< double > * sample,
            int N )
```

Calculates the variance for N given values.

Parameters

sample	Samples to calculate the variance of.
N	Number of samples.

Returns

The calculated variance of the samples.

2.1.1.9 sigma_naive()

```
double sigma_naive (
            int N,
            std::vector< double > * sample )
```

naively computing mean mu and sigma of a collection of normal distributed samples

Parameters

N	Number of given samples
sample	Pointer to the vector of double valued normal distributed samples

Returns

It returns sigma

2.1.1.10 wiener_process()

```
std::vector<double>* wiener_process (
            gsl_rng * r,
            double T,
            double delta_t )
```

Simulates a wiener process

Parameters

r	Pointer to the gsl_rng object for generating standard normal distributed numbers
T	Time period of simulated process
delta_t	Step of discretisation

Returns

Pointer to the vector of values at discretisation points

Index