

# Parameter estimation of a generalized Logisitc SDE with the ME method

## 1.0

Generated by Doxygen 1.9.1



<b>1 Todo List</b>	<b>1</b>
<b>2 Modules Index</b>	<b>3</b>
2.1 Modules List . . . . .	3
<b>3 File Index</b>	<b>5</b>
3.1 File List . . . . .	5
<b>4 Module Documentation</b>	<b>7</b>
4.1 mod_milstein_solver Module Reference . . . . .	7
4.1.1 Detailed Description . . . . .	7
4.1.2 Function/Subroutine Documentation . . . . .	8
4.1.2.1 get_brownian_increment() . . . . .	8
4.1.2.2 get_milstein_iteration() . . . . .	9
4.2 mod_random_number_generator Module Reference . . . . .	9
4.2.1 Detailed Description . . . . .	10
4.2.2 Function/Subroutine Documentation . . . . .	10
4.2.2.1 boxmuller() . . . . .	10
4.2.2.2 normalvar() . . . . .	11
4.2.2.3 unif() . . . . .	11
4.3 mod_stochastic_logistic_model Module Reference . . . . .	11
4.3.1 Detailed Description . . . . .	12
4.3.2 Function/Subroutine Documentation . . . . .	12
4.3.2.1 compute_diffusion() . . . . .	12
4.3.2.2 compute_drift() . . . . .	13
<b>5 File Documentation</b>	<b>15</b>
5.1 src/mod_milstein_solver.f95 File Reference . . . . .	15
5.2 src/mod_milstein_solver.f95.d File Reference . . . . .	15
5.3 src/mod_random_number_generator.f95 File Reference . . . . .	15
5.4 src/mod_random_number_generator.f95.d File Reference . . . . .	16
5.5 src/mod_stochastic_logistic_model.f95 File Reference . . . . .	16
5.6 src/mod_stochastic_logistic_model.f95.d File Reference . . . . .	16
<b>Index</b>	<b>17</b>



# Chapter 1

## Todo List

Subprogram `mod_random_number_generator::boxmuller()`

: Try other random-number generators, such like mersene an others

Subprogram `mod_random_number_generator::normalvar()`

: Implement this furnction such that returns a realization path of the standard Brownian motion



## Chapter 2

# Modules Index

### 2.1 Modules List

Here is a list of all modules with brief descriptions:

<a href="#">mod_milstein_solver</a>	This module compute the coefficients of the underlying generalized logistic SDE . . . . .	7
<a href="#">mod_random_number_generator</a>	This module implements the Box-Muller algorithm to generate random variables with standard Gaussian distribution from a uniform distributed random variable. This module enclose tree functions . . .	9
<a href="#">mod_stochastic_logistic_model</a>	This module compute the coefficients of the underlying generalized logistic SDE . . . . .	11





## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

src/ <a href="#">mod_milstein_solver.f95</a> . . . . .	15
src/ <a href="#">mod_milstein_solver.f95.d</a> . . . . .	15
src/ <a href="#">mod_random_number_generator.f95</a> . . . . .	15
src/ <a href="#">mod_random_number_generator.f95.d</a> . . . . .	16
src/ <a href="#">mod_stochastic_logistic_model.f95</a> . . . . .	16
src/ <a href="#">mod_stochastic_logistic_model.f95.d</a> . . . . .	16



## Chapter 4

# Module Documentation

### 4.1 mod\_milstein\_solver Module Reference

This module compute the coefficients of the underlying generalized logistic SDE.

#### Functions/Subroutines

- subroutine [get\\_brownian\\_increment](#) (delta, brownian\_start, brownian\_end)  
*Simulate an increment of the related Brownian path.*
- subroutine [get\\_milstein\\_iteration](#) (delta, current\_state, current\_drift, current\_diffusion, diffusion\_derivative, sigma, x\_next)  
*Computes a iteration of the Milstein method.*

#### 4.1.1 Detailed Description

This module compute the coefficients of the underlying generalized logistic SDE.

##### Author

E. Lince-Gomez, F. Baltazar-Larios, S. Diaz-Infante

$$\begin{aligned}dX(t) &= f(X(t))dt + g(X(t))dB(t) \\ f(x) &:= \alpha x \left[ 1 - \left( \frac{x}{K} \right)^m \right] \\ g(x) &:= \sigma x, \quad t > t_0 \\ x_0 &= X(0), \quad x_0 \in [0, 1].\end{aligned}$$

##### See also

the manuscript (...) for more details.

(details)

## 4.1.2 Function/Subroutine Documentation

### 4.1.2.1 `get_brownian_increment()`

```
subroutine mod_milstein_solver::get_brownian_increment (
    real(real32), intent(in) delta,
    real(real32), intent(in) brownian_start,
    real(real32), intent(out) brownian_end )
```

Simulate an increment of the related Brownian path.

This can be done by simulating the whole path

#### Parameters

in	<i>delta</i>	deterministic time step size
in	<i>brownian_start</i>	current observation of the Brownian process
in	<i>brownian_end</i>	observation at current time plus step-size

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.1.2.2 get\_milstein\_iteration()

```

subroutine mod_milstein_solver::get_milstein_iteration (
    real(real32), intent(in) delta,
    real(real32), intent(in) current_state,
    real(real32), intent(in) current_drift,
    real(real32), intent(in) current_diffusion,
    real(real32), intent(in) diffusion_derivative,
    real(real32), intent(in) sigma,
    real(real32), intent(out) x_next )

```

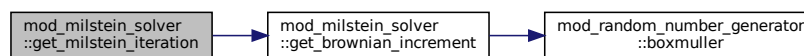
Computes a iteration of the Milstein method.

Defintion taken from the Kloedens book

## Parameters

in	<i>delta</i>	determininstic time step-size
in	<i>brownian_start</i>	current observation of the Brownian process the Brownian increment.

Here is the call graph for this function:



## 4.2 mod\_random\_number\_generator Module Reference

This module implements the Box-Muller algorithm to generate random variables with standard Gaussian distribution from a uniform ditributed random variable. This module enclose tree functions.

## Functions/Subroutines

- real function [unif](#) (ix)  
*Returns a random variables with uniform distribution using the standar gfortran random number generator, the returned value is a real 32.*
- real(real32) function [boxmuller](#) ()  
*Implementation of the Box-muller algorithm to genrate Gaussian random variables.*
- real(real32) function [normalvar](#) ()  
*Returns a number with Gaussian distribution using the Box-Muller algortihm.*

### 4.2.1 Detailed Description

This module implements the Box-Muller algorithm to generate random variables with standard Gaussian distribution from a uniform distributed random variable. This module enclose three functions.

#### Author

E. Lince-Gomez, F. Baltazar-Larios, S. Diaz-Infante

#### See also

Kloeden & Platen 1992

### 4.2.2 Function/Subroutine Documentation

#### 4.2.2.1 boxmuller()

```
real(real32) function mod_random_number_generator::boxmuller
```

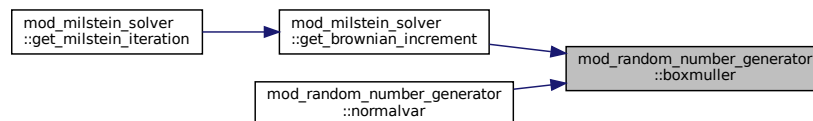
Implementation of the Box-muller algorithm to generate Gaussian random variables.

#### Parameters

in	seed	initial value for the uniform random generator
----	------	--

**Todo** : Try other random-number generators, such like mersenne and others

Here is the caller graph for this function:



#### 4.2.2.2 normalvar()

```
real(real32) function mod_random_number_generator::normalvar
```

Returns a number with Gaussian distribution using the Box-Muller algorithm.

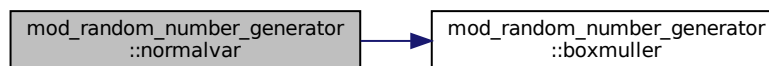
This function can be omitted in the case of a step forward implementation

##### Parameters

in	seed	a int32 with the initial value for the random uniform generator
----	------	---

**Todo** : Implement this function such that returns a realization path of the standard Brownian motion

Here is the call graph for this function:



#### 4.2.2.3 unif()

```
real function mod_random_number_generator::unif (
    integer(int32), intent(in) ix )
```

Returns a random variable with uniform distribution using the standard gfortran random number generator, the returned value is a real 32.

##### Parameters

in	ix	Dummy parameter for the seed initialization for the random generator
----	----	--

## 4.3 mod\_stochastic\_logistic\_model Module Reference

This module computes the coefficients of the underlying generalized logistic SDE.

## Functions/Subroutines

- subroutine `compute_drift` (alpha, m, x, y)  
     <BRIEF\_DESCRIPTION>
- subroutine `compute_diffusion` (sigma, x, y)  
     <BRIEF\_DESCRIPTION>

### 4.3.1 Detailed Description

This module compute the coefficients of the underlying generalized logistic SDE.

#### Author

E. Lince-Gomez, F. Baltazar-Larios, S. Diaz-Infante

$$\begin{aligned}
 dX(t) &= f(X(t))dt + g(X(t))dB(t) \\
 f(x) &:= \alpha x \left[ 1 - \left( \frac{x}{K} \right)^m \right] \\
 g(x) &:= \sigma x, \quad t > t_0 \\
 x_0 &= X(0), \quad x_0 \in [0, 1].
 \end{aligned}$$

#### See also

the manuscript (...) for more details.

#### Version

1.0

### 4.3.2 Function/Subroutine Documentation

#### 4.3.2.1 `compute_diffusion()`

```

subroutine mod_stochastic_logistic_model::compute_diffusion (
    real(real32), intent(in)  sigma,
    real(real32), intent(in)  x,
    real(real32), intent(out) y )

```

<BRIEF\_DESCRIPTION>

<DETAILED\_DESCRIPTION>



## Parameters

[<in	or out or inout>] <PARAM1>
[<in	or out or inout>] <PARAM2>

## 4.3.2.2 compute\_drift()

```
subroutine mod_stochastic_logistic_model::compute_drift (  
    real(real32), intent(in) alpha,  
    real(real32), intent(in) m,  
    real(real32), intent(in) x,  
    real(real32), intent(out) y )
```

&lt;BRIEF\_DESCRIPTION&gt;

&lt;DETAILED\_DESCRIPTION&gt;

## Parameters

[<in	or out or inout>] <PARAM1>
[<in	or out or inout>] <PARAM2>



## Chapter 5

# File Documentation

### 5.1 src/mod\_milstein\_solver.f95 File Reference

#### Modules

- module [mod\\_milstein\\_solver](#)

*This module compute the coefficients of the underlying generalized logistic SDE.*

#### Functions/Subroutines

- subroutine [mod\\_milstein\\_solver::get\\_brownian\\_increment](#) (delta, brownian\_start, brownian\_end)  
*Simulate an increment of the related Brownian path.*
- subroutine [mod\\_milstein\\_solver::get\\_milstein\\_iteration](#) (delta, current\_state, current\_drift, current\_diffusion, diffusion\_derivative, sigma, x\_next)  
*Computes a iteration of the Milstein method.*

### 5.2 src/mod\_milstein\_solver.f95.d File Reference

### 5.3 src/mod\_random\_number\_generator.f95 File Reference

#### Modules

- module [mod\\_random\\_number\\_generator](#)

*This module implements the Box-Muller algorithm to generate random variables with standard Gaussian distribution from a uniform distributed random variable. This module enclose three functions.*

## Functions/Subroutines

- real function `mod_random_number_generator::unif` (ix)  
*Returns a random variables with uniform distribution using the standar gfortran random number generator, the returned value is a real 32.*
- real(real32) function `mod_random_number_generator::boxmuller` ()  
*Implementation of the Box-muller algorithm to genrate Gaussian random variables.*
- real(real32) function `mod_random_number_generator::normalvar` ()  
*Returns a number with Gaussian distribution using the Box-Muller algortihm.*

## 5.4 src/mod\_random\_number\_generator.f95.d File Reference

## 5.5 src/mod\_stochastic\_logistic\_model.f95 File Reference

## Modules

- module `mod_stochastic_logistic_model`  
*This module compute the coefficients of the underlying generalized logistic SDE.*

## Functions/Subroutines

- subroutine `mod_stochastic_logistic_model::compute_drift` (alpha, m, x, y)  
<BRIEF\_DESCRIPTION>
- subroutine `mod_stochastic_logistic_model::compute_diffusion` (sigma, x, y)  
<BRIEF\_DESCRIPTION>

## 5.6 src/mod\_stochastic\_logistic\_model.f95.d File Reference

# Index

- boxmuller
  - mod\_random\_number\_generator, [10](#)
- compute\_diffusion
  - mod\_stochastic\_logistic\_model, [12](#)
- compute\_drift
  - mod\_stochastic\_logistic\_model, [13](#)
- get\_brownian\_increment
  - mod\_milstein\_solver, [8](#)
- get\_milstein\_iteration
  - mod\_milstein\_solver, [8](#)
- mod\_milstein\_solver, [7](#)
  - get\_brownian\_increment, [8](#)
  - get\_milstein\_iteration, [8](#)
- mod\_random\_number\_generator, [9](#)
  - boxmuller, [10](#)
  - normalvar, [10](#)
  - unif, [11](#)
- mod\_stochastic\_logistic\_model, [11](#)
  - compute\_diffusion, [12](#)
  - compute\_drift, [13](#)
- normalvar
  - mod\_random\_number\_generator, [10](#)
- src/mod\_milstein\_solver.f95, [15](#)
- src/mod\_milstein\_solver.f95.d, [15](#)
- src/mod\_random\_number\_generator.f95, [15](#)
- src/mod\_random\_number\_generator.f95.d, [16](#)
- src/mod\_stochastic\_logistic\_model.f95, [16](#)
- src/mod\_stochastic\_logistic\_model.f95.d, [16](#)
- unif
  - mod\_random\_number\_generator, [11](#)