

# Package ‘rlfsm’

April 16, 2021

**Type** Package

**Title** Simulations and Statistical Inference for Linear Fractional Stable Motions

**Version** 1.0.3

**Maintainer** Dmitry Otryakhin <d.otryakhin.acad@protonmail.ch>

**Author** Dmitry Otryakhin [aut, cre] (<<https://orcid.org/0000-0002-4700-7221>>),  
Stepan Mazur [aut] (<<https://orcid.org/0000-0002-1395-9427>>),  
Mathias Ljungdahl [ctb]

## Description

Contains functions for simulating linear fractional stable motions, according to techniques developed by Stoev and Taqqu (2004) <doi:10.1142/S0218348X04002379>, as well as functions for computing important statistics used with these processes introduced by Mazur, Otryakhin and Podolskij (2018) <arXiv:1802.06373>, and also different quantities related to those statistics.

**License** GPL-3

**URL** [https://gitlab.com/Dmitry\\_Otryakhin/  
Tools-for-parameter-estimation-of-the-linear-fractional-stable-motion](https://gitlab.com/Dmitry_Otryakhin/Tools-for-parameter-estimation-of-the-linear-fractional-stable-motion)

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**Depends** methods, foreach, doParallel

**Imports** ggplot2, stabledist, reshape2, plyr, Rdpack, Rcpp

**Suggests** elliptic, spatstat.core, spatstat (>= 2.0-0), testthat, stringi

**RdMacros** Rdpack

**LinkingTo** Rcpp

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2021-04-16 13:10:02 UTC

**R topics documented:**

|                  |           |
|------------------|-----------|
| alpha_hat        | 2         |
| a_p              | 4         |
| a_tilda          | 4         |
| CLT              | 5         |
| ContinEstim      | 6         |
| GenHighEstim     | 7         |
| GenLowEstim      | 8         |
| H_hat            | 9         |
| h_kr             | 10        |
| increment        | 11        |
| MCestimLFSM      | 12        |
| MinContrastEstim | 14        |
| m_pk             | 15        |
| Norm_alpha       | 16        |
| path             | 17        |
| paths            | 19        |
| Path_array       | 20        |
| phi              | 20        |
| phi_of_alpha     | 21        |
| Plot_dens        | 22        |
| Plot_list_paths  | 23        |
| Plot_vb          | 23        |
| Retrieve_stats   | 24        |
| R_hl             | 25        |
| sf               | 26        |
| sigma_hat        | 28        |
| theta            | 29        |
| U_g              | 29        |
| U_gh             | 30        |
| U_ghuv           | 30        |
| <b>Index</b>     | <b>32</b> |

---

|           |  |
|-----------|--|
| alpha_hat | <i>Statistical estimator for alpha</i> |
|-----------|--|

---

**Description**

Defined for the two frequencies as

$$\hat{\alpha}_{high} := \frac{\log |\log \varphi_{high}(t_2; \hat{H}_{high}(p, k)_n, k)_n| - \log |\log \varphi_{high}(t_1; \hat{H}_{high}(p, k)_n, k)_n|}{\log t_2 - \log t_1}$$

$$\hat{\alpha}_{low} := \frac{\log |\log \varphi_{low}(t_2; k)_n| - \log |\log \varphi_{low}(t_1; k)_n|}{\log t_2 - \log t_1}$$

**Usage**

```
alpha_hat(t1, t2, k, path, H, freq)
```

**Arguments**

|      |  |
|------|--|
| t1   | real number such that $t_2 > t_1 > 0$  |
| t2   | real number such that $t_2 > t_1 > 0$  |
| k    | increment order  |
| path | sample path of lfsm on which the inference is to be performed  |
| H    | Hurst parameter  |
| freq | Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting. |

**Details**

The function triggers function [phi](#), thus Hurst parameter is required only in high frequency case. In the low frequency, there is no need to assign H a value because it will not be evaluated.

**References**

Mazur S, Otryakhin D, Podolskij M (2020). “Estimation of the linear fractional stable motion.” *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

**Examples**

```
m<-45; M<-60; N<-2^14-M
alpha<-1.8; H<-0.8; sigma<-0.3
freq='H'
r=1; k=2; p=0.4; t1=1; t2=2

# Estimating alpha in the high frequency case
# using preliminary estimation of H
lfsm<-path(N=N,m=m,M=M,alpha=alpha,H=H,
          sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm

H_est<-H_hat(p=p,k=k,path=lfsm)
H_est
alpha_est<-alpha_hat(t1=t1,t2=t2,k=k,path=lfsm,H=H_est,freq=freq)
alpha_est
```

---

|     |                      |
|-----|----------------------|
| a_p | <i>Function a_p.</i> |
|-----|----------------------|

---

### Description

Computes the corresponding function value from Mazur et al. 2018.

### Usage

a\_p(p)

### Arguments

p power, real number from (-1,1)

### References

Mazur S, Otryakhin D, Podolskij M (2020). “Estimation of the linear fractional stable motion.” *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

---

|         |  |
|---------|--|
| a_tilda | <i>Creates the corresponding value from the paper by Stoev and Taqqu (2004).</i> |
|---------|--|

---

### Description

a\_tilda triggers a\_tilda\_cpp which is written in C++ and essentially performs the computation of the value.

### Usage

a\_tilda(N, m, M, alpha, H)

a\_tilda\_R(N, m, M, alpha, H)

### Arguments

N a number of points of the lfsm.  
m discretization. A number of points between two nearby motion points  
M truncation parameter. A number of points at which the integral representing the definition of lfsm is calculated. So, after M points back we consider the rest of the integral to be 0.  
alpha self-similarity parameter of alpha stable random motion.  
H Hurst parameter

## Functions

- `a_tilda_R`: Old version of `a_tilda` function. Written entirely in R. It was left for comparison, testing and backward compatibility.

## References

Stoev S, Taqqu MS (2004). "Simulation methods for linear fractional stable motion and FARIMA using the Fast Fourier Transform." *Fractals*, **95**(1), 95–121. doi: [10.1142/S0218348X04002379](https://doi.org/10.1142/S0218348X04002379), <https://doi.org/10.1142/S0218348X04002379>.

---

|     |  |
|-----|--|
| CLT | <i>The function explores numerical properties of statistical estimators operating on random processes. Deprecated.</i> |
|-----|--|

---

## Description

The function is left for backward compatibility. The newer version of it is `MCestimLFSM`. The function is useful, for instance, when one needs to compute standard deviation of  $\hat{\alpha}_{high}$  estimator given a fixed set of parameters.

## Usage

```
CLT(Nmc, s, m, M, alpha, H, sigma, fr, Inference, ...)
```

## Arguments

|                        |   |
|------------------------|---|
| <code>Nmc</code>       | Number of Monte Carlo repetitions   |
| <code>s</code>         | sequence of path lengths  |
| <code>m</code>         | discretization. A number of points between two nearby motion points   |
| <code>M</code>         | truncation parameter. A number of points at which the integral representing the definition of <code>lfsm</code> is calculated. So, after <code>M</code> points back we consider the rest of the integral to be 0. |
| <code>alpha</code>     | self-similarity parameter of alpha stable random motion.  |
| <code>H</code>         | Hurst parameter   |
| <code>sigma</code>     | Scale parameter of <code>lfsm</code>  |
| <code>fr</code>        | frequency. Either "H" or "L"  |
| <code>Inference</code> | statistical function to apply to sample paths   |
| <code>...</code>       | parameters to pass to <code>Inference</code>  |

## Details

CLT performs Monte-Carlo experiments to compute parameters according to procedure Inference. More specifically, for each element of  $s$  it generates  $N_{mc}$  Ifsm sample paths with length equal to  $s[i]$ , performs the statistical inference on each, obtaining the estimates, and then returns their different statistics. It is vital that the estimator returns a list of named parameters (one or several of 'sigma', 'alpha' and 'H'). CLT uses the names to lookup the true parameter value and compute its bias.

For sample path generation CLT uses a light-weight version of path, path\_fast. In order to be applied, function Inference must accept argument 'path' as a sample path.

## Value

It returns a list containing the following components:

|                 |   |
|-----------------|---|
| CLT_dataset     | a data frame, standardized values of the estimates depending on path length $s$ |
| BSdM            | a data frame, means, biases and standard deviations depending on $s$            |
| Inference       | a closure used to obtain estimates  |
| alpha, H, sigma | the parameters for which CLT performs path generation                           |
| freq            | frequency, either 'L' for low- or 'H' for high frequency                        |

---

ContinEstim

*Parameter estimation procedure in continuous case.*

---

## Description

Parameter freq is preserved to allow for investigation of the inference procedure in high frequency case.

## Usage

```
ContinEstim(t1, t2, p, k, path, freq)
```

## Arguments

|        |  |
|--------|--|
| t1, t2 | real number such that $t_2 > t_1 > 0$  |
| p      | power  |
| k      | increment order  |
| path   | sample path of Ifsm on which the inference is to be performed  |
| freq   | Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting. |

## References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

**Examples**

```

m<-45; M<-60; N<-2^10-M
alpha<-0.8; H<-0.8; sigma<-0.3
p<-0.3; k=3; t1=1; t2=2

lfsm<-path(N=N,m=m,M=M,alpha=alpha,H=H,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm
ContinEstim(t1,t2,p,k,path=lfsm,freq='L')

```

GenHighEstim

*High frequency estimation procedure for lfsm.***Description**

General estimation procedure for high frequency case when  $1/\alpha$  is not a natural number. "Un-necessary" parameter freq is preserved to allow for investigation of the inference procedure in low frequency case

**Usage**

```
GenHighEstim(p, p_prime, path, freq, low_bound = 0.01, up_bound = 4)
```

**Arguments**

|           |  |
|-----------|--|
| p         | power  |
| p_prime   | power  |
| path      | sample path of lfsm on which the inference is to be performed  |
| freq      | Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting. |
| low_bound | positive real number   |
| up_bound  | positive real number   |

**Details**

In this algorithm the preliminary estimate of alpha is found via using `uniroot` function. The latter is given the lower and the upper bounds for alpha via `low_bound` and `up_bound` parameters. It is not possible to pass 0 as the lower bound because there are numerical limitations on the alpha estimate, caused by the length of the sample path and by numerical errors. `p` and `p_prime` must belong to the interval  $(0,1/2)$  (in the notation kept in `rlfsm` package) The two powers cannot be equal.

**References**

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

**Examples**

```

m<-45; M<-60; N<-2^10-M
sigma<-0.3
p<-0.2; p_prime<-0.4

#### Continuous case
lfsm<-path(N=N,m=m,M=M,alpha=1.8,H=0.8,
          sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm

GenHighEstim(p=p,p_prime=p_prime,path=lfsm,freq="H")

#### H-1/alpha<0 case
lfsm<-path(N=N,m=m,M=M,alpha=0.8,H=0.8,
          sigma=sigma,freq='H',disable_X=FALSE,seed=3)$lfsm

GenHighEstim(p=p,p_prime=p_prime,path=lfsm,freq="H")

```

---

GenLowEstim

*Low frequency estimation procedure for lfsm.*


---

**Description**

General estimation procedure for low frequency case when  $1/\alpha$  is not a natural number.

**Usage**

```
GenLowEstim(t1, t2, p, path, freq = "L")
```

**Arguments**

|      |  |
|------|--|
| t1   | real number such that $t_2 > t_1 > 0$  |
| t2   | real number such that $t_2 > t_1 > 0$  |
| p    | power  |
| path | sample path of lfsm on which the inference is to be performed  |
| freq | Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting. |

**References**

Mazur S, Otryakhin D, Podolskij M (2020). “Estimation of the linear fractional stable motion.” *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.



**Examples**

```

m<-45; M<-60; N<-2^10-M
sigma<-0.3
p<-0.3; k=3; t1=1; t2=2

#### Continuous case
lfsm<-path(N=N,m=m,M=M,alpha=1.8,H=0.8,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm

GenLowEstim(t1=t1,t2=t2,p=p,path=lfsm,freq="L")

#### H-1/alpha<0 case
lfsm<-path(N=N,m=m,M=M,alpha=0.8,H=0.8,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm

GenLowEstim(t1=t1,t2=t2,p=p,path=lfsm,freq="L")

#### The procedure works also for high frequency case
lfsm<-path(N=N,m=m,M=M,alpha=1.8,H=0.8,
           sigma=sigma,freq='H',disable_X=FALSE,seed=3)$lfsm

GenLowEstim(t1=t1,t2=t2,p=p,path=lfsm,freq="H")

```

H\_hat

*Statistical estimator of H in high/low frequency setting***Description**

The statistic is defined as

$$\hat{H}_{\text{high}}(p, k)_n := \frac{1}{p} \log_2 R_{\text{high}}(p, k)_n, \quad \hat{H}_{\text{low}}(p, k)_n := \frac{1}{p} \log_2 R_{\text{low}}(p, k)_n$$

**Usage**

H\_hat(p, k, path)

**Arguments**

|      |   |
|------|---|
| p    | power   |
| k    | increment order   |
| path | sample path of lfsm on which the inference is to be performed |

**References**

Mazur S, Otryakhin D, Podolskij M (2020). “Estimation of the linear fractional stable motion.” *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

h\_kr

Function h\_kr

**Description**

Function  $h_{k,r} : R \rightarrow R$  is given by

$$h_{k,r}(x) = \sum_{j=0}^k (-1)^j \binom{k}{j} (x - rj)_+^{H-1/\alpha}, \quad x \in R$$

**Usage**

`h_kr(k, r, x, H, alpha, l = 0)`

**Arguments**

|                    |   |
|--------------------|---|
| <code>k</code>     | order of the increment, a natural number  |
| <code>r</code>     | difference step, a natural number   |
| <code>x</code>     | real number   |
| <code>H</code>     | Hurst parameter   |
| <code>alpha</code> | self-similarity parameter of alpha stable random motion.  |
| <code>l</code>     | a value by which we shift x. Is used for computing function <code>f_{.}+l</code> and is passed to integrate function. |

**References**

Mazur S, Otryakhin D, Podolskij M (2020). “Estimation of the linear fractional stable motion.” *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

**Examples**

```
#### Plot h_kr ####
s<-seq(0,10, by=0.01)
h_val<-sapply(s,h_kr, k=5, r=1, H=0.3, alpha=1)
plot(s,h_val)
```

---

|           |                                |
|-----------|--------------------------------|
| increment | <i>Higher order increments</i> |
|-----------|--------------------------------|

---

### Description

Difference of the kth order. Defined as following:

$$\Delta_{i,k}^{n,r} X := \sum_{j=0}^k (-1)^j \binom{k}{j} X_{(i-rj)/n}, i \geq rk.$$

Index  $i$  here is a coordinate in terms of `point_num`. Although R uses vector indexes that start from 1, `increment` has  $i$  varying from 0 to  $N$ , so that a vector has a length  $N+1$ . It is done in order to comply with the notation of the paper. This function doesn't allow for choosing frequency  $n$ . The frequency is determined by the path supplied, thus  $n$  equals to either the length of the path in high frequency setting or 1 in low frequency setting. `increment()` gives increments at certain point passed as  $i$ , which is a vector here. `increments()` computes high order increments for the whole sample path. The first function evaluates the formula above, while the second one uses structure `diff(diff(...))` because the formula is slower at higher  $k$ .

### Usage

```
increment(r, i, k, path)
```

```
increments(k, r, path)
```

### Arguments

|                   |  |
|-------------------|--|
| <code>r</code>    | difference step, a natural number  |
| <code>i</code>    | index of the point at which the increment is to be computed, a natural number. |
| <code>k</code>    | order of the increment, a natural number                                       |
| <code>path</code> | sample path for which a kth order increment is computed                        |

### References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

### Examples

```
m<-45; M<-60; N<-2^10-M
alpha<-0.8; H<-0.8; sigma<-0.3

lfsm<-path(N=N,m=m,M=M,alpha=alpha,H=H,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm

tryCatch(
```

```

increment(r=1,i=length(lfsm),k=length(lfsm)+100,path=lfsm),
error=function(c) 'An error occurs when k is larger then the length of the sample path')

increment(r=3,i=50,k=3,path=lfsm)

path=c(1,4,3,6,8,5,3,5,8,5,1,8,6)

r=2; k=3
n <- length(path) - 1
DeltaX = increment(seq(r*k, n), path = path, k = k, r = r)
DeltaX == increments(k=k,r=r,path)

```

---

|             |   |
|-------------|---|
| MCestimLFSM | <i>Numerical properties of statistical estimators operating on the linear fractional stable motion.</i> |
|-------------|---|

---

### Description

The function is useful, for instance, when one needs to compute standard deviation of  $\hat{\alpha}_{high}$  estimator given a fixed set of parameters.

### Usage

```
MCestimLFSM(Nmc, s, m, M, alpha, H, sigma, fr, Inference, ...)
```

### Arguments

|           |   |
|-----------|---|
| Nmc       | Number of Monte Carlo repetitions   |
| s         | sequence of path lengths  |
| m         | discretization. A number of points between two nearby motion points   |
| M         | truncation parameter. A number of points at which the integral representing the definition of lfsm is calculated. So, after M points back we consider the rest of the integral to be 0. |
| alpha     | self-similarity parameter of alpha stable random motion.  |
| H         | Hurst parameter   |
| sigma     | Scale parameter of lfsm   |
| fr        | frequency. Either "H" or "L"  |
| Inference | statistical function to apply to sample paths   |
| ...       | parameters to pass to Inference   |

## Details

MCestimLFSM performs Monte-Carlo experiments to compute parameters according to procedure Inference. More specifically, for each element of  $s$  it generates  $N_{mc}$  lfsm sample paths with length equal to  $s[i]$ , performs the statistical inference on each, obtaining the estimates, and then returns their different statistics. It is vital that the estimator returns a list of named parameters (one or several of 'sigma', 'alpha' and 'H'). MCestimLFSM uses the names to lookup the true parameter value and compute its bias.

For sample path generation MCestimLFSM uses a light-weight version of `path`, `path_fast`. In order to be applied, function `Inference` must accept argument 'path' as a sample path.

## Value

It returns a list containing the following components:

|                                 |   |
|---------------------------------|---|
| <code>data</code>               | a data frame, values of the estimates depending on path length $s$                    |
| <code>data_nor</code>           | a data frame, normalized values of the estimates depending on path length $s$         |
| <code>means, biases, sds</code> | data frames: means, biases and standard deviations of the estimators depending on $s$ |
| <code>Inference</code>          | a function used to obtain estimates   |
| <code>alpha, H, sigma</code>    | the parameters for which MCestimLFSM performs path generation                         |
| <code>freq</code>               | frequency, either 'L' for low- or 'H' for high frequency                              |

## Examples

```
#### Set of global parameters ####
m<-25; M<-60
p<-.4; p_prime<-.2; k<-2
t1<-1; t2<-2
NmonteC<-5e1
S<-c(1e2,3e2)
alpha<-1.8; H<-0.8; sigma<-0.3

# How to plot empirical density

theor_3_1_H_clt<-MCestimLFSM(s=S, fr='H', Nmc=NmonteC,
                           m=m, M=M, alpha=alpha, H=H,
                           sigma=sigma, ContinEstim,
                           t1=t1, t2=t2, p=p, k=k)
l_plot<-Plot_dens(par_vec=c('sigma', 'alpha', 'H'),
                 MC_data=theor_3_1_H_clt, Nnorm=1e7)

# For MCestimLFSM() it is vital that the estimator returns a list of named parameters

H_hat_f <- function(p,k,path) {hh<-H_hat(p,k,path); list(H=hh)}
theor_3_1_H_clt<-MCestimLFSM(s=S, fr='H', Nmc=NmonteC,
```

```

m=m,M=M,alpha=alpha,H=H,
sigma=sigma,H_hat=f,
p=p,k=k)

# The estimator can return one, two or three of the parameters.

est_1 <- function(path) list(H=1)
theor_3_1_H_clt<-MCestimLFSM(s=S,fr='H',Nmc=NmonteC,
m=m,M=M,alpha=alpha,H=H,
sigma=sigma,est_1)

est_2 <- function(path) list(H=0.8, alpha=1.5)
theor_3_1_H_clt<-MCestimLFSM(s=S,fr='H',Nmc=NmonteC,
m=m,M=M,alpha=alpha,H=H,
sigma=sigma,est_2)

est_3 <- function(path) list(sigma=5, H=0.8, alpha=1.5)
theor_3_1_H_clt<-MCestimLFSM(s=S,fr='H',Nmc=NmonteC,
m=m,M=M,alpha=alpha,H=H,
sigma=sigma,est_3)

```

---

|                  |  |
|------------------|--|
| MinContrastEstim | <i>Statistical estimator of sigma, alpha and H in low frequency setting based on minimal contrast estimation comparing the empirical characteristic function with the true one</i> |
|------------------|--|

---

## Description

Estimates H using the [H\\_hat](#) function while sigma and alpha are obtained via

$$\arg \min_{\sigma, \alpha} \int_0^{\infty} (\varphi_n(t) - \varphi_{\sigma, \alpha, H_{hat}}(t))^2 \exp(-t^2/2) dt$$

, where  $\varphi_n$  is the empirical characteristic function, see [phi](#), and  $\varphi_{\sigma, \alpha, H_{hat}}$  is the characteristic function of the kth order increment wrt the parameters  $\sigma, \alpha, H_{hat}$ , see also [increment](#).

## Usage

```
MinContrastEstim(path, k, p, order_GH)
```

## Arguments

|          |   |
|----------|---|
| path     | low frequency sample path from which the parameters should be estimated.  |
| k        | order of increments.  |
| p        | any real number, the power used for <a href="#">H_hat</a> .   |
| order_GH | number of weights in the Gauss-Hermite approximation of the integral, see the <code>gauss.hermite</code> function from the <code>spatstat</code> package. |

**Details**

This algorithm approximates the above integral using Gauss-Hermite quadrature and uses the L-BFGS-B method from the `optim` function to minimize over the parameters `sigma` and `alpha`. Due to numerical problems estimation of `sigma` below 0.01 and `alpha` or `H` below 0.05 is currently not possible.

**References**

Ljungdahl MM, Podolskij M (2019). “A Minimal Contrast Estimator for the Linear Fractional Stable Motion.” *Statistical Inference for Stochastic Processes*, 1–33.

**Examples**

```
m0 = 256
M0 = 600
alpha0 = 1.8
H0 = 0.8
sigma0 = 0.3
n = 100
```

```
X <- path(N = n, m = m0, M = M0, alpha = alpha0, H = H0, sigma = sigma0, freq = 'L', seed = 1)$lfsm
MinContrastEstim(path = X, k = 2, p = 0.4, order_GH = 8)
```

---

m\_pk

 $m(-p,k)$ 


---

**Description**

defined as  $m_{p,k} := E[|\Delta_{k,k}X|^p]$  for positive powers. When `p` is negative (`-p` is positive) the equality does not hold.

**Usage**

```
m_pk(k, p, alpha, H, sigma)
```

**Arguments**

|                    |  |
|--------------------|--|
| <code>k</code>     | increment order  |
| <code>p</code>     | a positive number  |
| <code>alpha</code> | self-similarity parameter of alpha stable random motion. |
| <code>H</code>     | Hurst parameter  |
| <code>sigma</code> | Scale parameter of lfsm                                  |

**Details**

The following identity is used for computations:

$$m_{-p,k} = \frac{(\sigma \|h_k\|_\alpha)^{-p}}{a_{-p}} \int_{\mathbf{R}} \exp(-|y|^\alpha) |y|^{-1+p} dy = \frac{2(\sigma \|h_k\|_\alpha)^{-p}}{\alpha a_{-p}} \Gamma(p/\alpha)$$

**References**

Mazur S, Otryakhin D, Podolskij M (2020). “Estimation of the linear fractional stable motion.” *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

---

 Norm\_alpha

*Alpha-norm of an arbitrary function*


---

**Description**

Alpha-norm of an arbitrary function

**Usage**

Norm\_alpha(fun, alpha, ...)

**Arguments**

|       |  |
|-------|--|
| fun   | a function to compute a norm                             |
| alpha | self-similarity parameter of alpha stable random motion. |
| ...   | a set of parameters to pass to integrate                 |

**Details**

fun must accept a vector of values for evaluation. See ?integrate for further details. Most problems with this function appear because of rather high precision. Try to tune rel.tol parameter first.

**References**

Mazur S, Otryakhin D, Podolskij M (2020). “Estimation of the linear fractional stable motion.” *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

**Examples**

```
Norm_alpha(h_kr, alpha=1.8, k=2, r=1, H=0.8, l=4)
```



---

path *Generator of linear fractional stable motion*

---

### Description

The function creates a 1-dimensional lfsm sample path using the numerical algorithm from the paper by Stoev and Taqqu. Linear fractional stable motion is defined as

$$X_t = \int_{\mathbf{R}} \left\{ (t-s)_+^{H-1/\alpha} - (-s)_+^{H-1/\alpha} \right\} dL_s$$

### Usage

```
path(
  N = NULL,
  m,
  M,
  alpha,
  H,
  sigma,
  freq,
  disable_X = FALSE,
  levy_increments = NULL,
  seed = NULL
)
```

### Arguments

|                 |  |
|-----------------|--|
| N               | a number of points of the lfsm.  |
| m               | discretization. A number of points between two nearby motion points  |
| M               | truncation parameter. A number of points at which the integral representing the definition of lfsm is calculated. So, after M points back we consider the rest of the integral to be 0.  |
| alpha           | self-similarity parameter of alpha stable random motion.   |
| H               | Hurst parameter  |
| sigma           | Scale parameter of lfsm  |
| freq            | Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting.   |
| disable_X       | is needed to disable computation of X. The default value is FALSE. When it is TRUE, only a levy motion is returned, which in turn reduces the computation time. The feature is particularly useful for reproducibility when combined with seeding. |
| levy_increments | increments of Levy motion underlying the lfsm.   |
| seed            | this parameter performs seeding of path generator  |

**Value**

It returns a list containing the motion, the underlying Levy motion, the point number of the motions from 0 to N and the corresponding coordinate (which depends on the frequency), the parameters that were used to generate the lfsm, and the predefined frequency.

**References**

Stoev S, Taqqu MS (2004). "Simulation methods for linear fractional stable motion and FARIMA using the Fast Fourier Transform." *Fractals*, **95**(1), 95–121. doi: [10.1142/S0218348X04002379](https://doi.org/10.1142/S0218348X04002379), <https://doi.org/10.1142/S0218348X04002379>.

**See Also**

[paths](#) simulates a number of lfsm sample paths.

**Examples**

```
# Path generation

m<-256; M<-600; N<-2^10-M
alpha<-1.8; H<-0.8; sigma<-0.3
seed=2

List<-path(N=N,m=m,M=M,alpha=alpha,H=H,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)

# Normalized paths
Norm_lfsm<-List[['lfsm']]/max(abs(List[['lfsm']]))
Norm_oLm<-List[['levy_motion']]/max(abs(List[['levy_motion']]))

# Visualization of the paths
plot(Norm_lfsm, col=2, type="l", ylab="coordinate")
lines(Norm_oLm, col=3)
leg.txt <- c("lfsm", "oLm")
legend("topright",legend = leg.txt, col =c(2,3), pch=1)

# Creating Levy motion
levyIncrems<-path(N=N, m=m, M=M, alpha, H, sigma, freq='L',
                 disable_X=TRUE, levy_increments=NULL, seed=seed)

# Creating lfsm based on the levy motion
lfsm_full<-path(m=m, M=M, alpha=alpha,
               H=H, sigma=sigma, freq='L',
               disable_X=FALSE,
               levy_increments=levyIncrems$levy_increments,
               seed=seed)

sum(levyIncrems$levy_increments==
    lfsm_full$levy_increments)==length(lfsm_full$levy_increments)
```

---

|       |  |
|-------|--|
| paths | <i>Generator of a set of lfsm paths.</i> |
|-------|--|

---

### Description

It is essentially a wrapper for [path](#) generator, which exploits the latest to create a matrix with paths in its columns.

### Usage

```
paths(N_var, parallel, seed_list = rep(x = NULL, times = N_var), ...)
```

### Arguments

|           |   |
|-----------|---|
| N_var     | number of lfsm paths to generate  |
| parallel  | a TRUE/FALSE flag which determines if the paths will be created in parallel or sequentially |
| seed_list | a numerical vector of seeds to pass to <a href="#">path</a>                                 |
| ...       | arguments to pass to path   |

### See Also

[path](#)

### Examples

```
m<-45; M<-60; N<-2^10-M
alpha<-1.8; H<-0.8; sigma<-0.3
freq='L'
r=1; k=2; p=0.4

Y<-paths(N_var=10,parallel=TRUE,N=N,m=m,M=M,
         alpha=alpha,H=H,sigma=sigma,freq='L',
         disable_X=FALSE,levy_increments=NULL)

Hs<-apply(Y,MARGIN=2,H_hat,p=p,k=k)
hist(Hs)
```

---

|            |                             |
|------------|-----------------------------|
| Path_array | <i>Path array generator</i> |
|------------|-----------------------------|

---

### Description

The function takes a list of parameters (alpha, H) and uses `expand.grid` to obtain all possible combinations of them. Based on each combination, the function simulates an lfsm sample path. It is meant to be used in conjunction with function `Plot_list_paths`.

### Usage

```
Path_array(N, m, M, l, sigma)
```

### Arguments

|       |   |
|-------|---|
| N     | a number of points of the lfsm.   |
| m     | discretization. A number of points between two nearby motion points   |
| M     | truncation parameter. A number of points at which the integral representing the definition of lfsm is calculated. So, after M points back we consider the rest of the integral to be 0. |
| l     | a list of parameters to expand  |
| sigma | Scale parameter of lfsm   |

### Value

The returned value is a data frame containing paths and the corresponding values of alpha, H and frequency.

### Examples

```
l=list(H=c(0.2,0.8),alpha=c(1,1.8), freq="H")
arr<-Path_array(N=300,m=30,M=100,l=l,sigma=0.3)
str(arr)
head(arr)
```

---

|     |            |
|-----|------------|
| phi | <i>Phi</i> |
|-----|------------|

---

### Description

Defined as

$$\varphi_{\text{high}}(t; H, k)_n := V_{\text{high}}(\psi_t; k)_n \quad \text{and} \quad \varphi_{\text{low}}(t; k)_n := V_{\text{low}}(\psi_t; k)_n$$

, where  $\psi_t(x) := \cos(tx)$

**Usage**

phi(t, k, path, H, freq)

**Arguments**

|      |  |
|------|--|
| t    | positive real number   |
| k    | increment order  |
| path | sample path of lfsm on which the inference is to be performed  |
| H    | Hurst parameter  |
| freq | Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting. |

**Details**

Hurst parameter is required only in high frequency case. In the low frequency, there is no need to assign H a value because it will not be evaluated.

**References**

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

---

|              |                                |
|--------------|--------------------------------|
| phi_of_alpha | <i>Inverse alpha estimator</i> |
|--------------|--------------------------------|

---

**Description**

A function from a general estimation procedure which is defined as  $m^{\wedge}p_{-p}'_k / m^{\wedge}p'_{-p}_k$ , originally proposed in [13].

**Usage**

phi\_of\_alpha(p, p\_prime, alpha)

**Arguments**

|         |  |
|---------|--|
| p       | power  |
| p_prime | power  |
| alpha   | self-similarity parameter of alpha stable random motion. |

**References**

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

---

|           |                                      |
|-----------|--------------------------------------|
| Plot_dens | <i>(alpha,H,sigma)- density plot</i> |
|-----------|--------------------------------------|

---

### Description

Plots the densities of the parameters (alpha,H,sigma) estimated in Monte-Carlo experiment. Works in conjunction with [MCestimLFSM](#) function.

### Usage

```
Plot_dens(par_vec = c("alpha", "H", "sigma"), MC_data, Nnorm = 1e+07)
```

### Arguments

|         |   |
|---------|---|
| par_vec | vector of parameters which are to be plotted              |
| MC_data | a list created by <a href="#">MCestimLFSM</a>             |
| Nnorm   | number of point sampled from standard normal distribution |

### See Also

[Plot\\_vb](#) to plot variance- and bias dependencies on n.

### Examples

```
m<-45; M<-60

p<-.4; p_prime<-.2
t1<-1; t2<-2; k<-2

NmonteC<-5e2
S<-c(1e3,1e4)
alpha<-.8; H<-0.8; sigma<-0.3
theor_4_1_clt_new<-MCestimLFSM(s=S, fr='L', Nmc=NmonteC,
                             m=m, M=M,
                             alpha=alpha, H=H, sigma=sigma,
                             GenLowEstim, t1=t1, t2=t2, p=p)
l_plot<-Plot_dens(par_vec=c('sigma', 'alpha', 'H'), MC_data=theor_4_1_clt_new, Nnorm=1e7)
l_plot
```

---

|                 |                                  |
|-----------------|----------------------------------|
| Plot_list_paths | <i>Rendering of path lattice</i> |
|-----------------|----------------------------------|

---

**Description**

Rendering of path lattice

**Usage**

```
Plot_list_paths(arr)
```

**Arguments**

arr                    a data frame produced by [Path\\_array](#).

**Examples**

```
l=list(H=c(0.2,0.8),alpha=c(1,1.8), freq="H")
arr<-Path_array(N=300,m=30,M=100,l=1,sigma=0.3)
p<-Plot_list_paths(arr)
p
```

---

|         |   |
|---------|---|
| Plot_vb | <i>A function to plot variance- and bias dependencies of estimators on the lengths of sample paths. Works in conjunction with <a href="#">MCestimLFSM</a> function.</i> |
|---------|---|

---

**Description**

A function to plot variance- and bias dependencies of estimators on the lengths of sample paths. Works in conjunction with [MCestimLFSM](#) function.

**Usage**

```
Plot_vb(data)
```

**Arguments**

data                    a list created by [MCestimLFSM](#)

**Value**

The function returns a ggplot2 graph.

**See Also**

[Plot\\_dens](#)

**Examples**

```

# Light weight computaions

m<-25; M<-50
alpha<-1.8; H<-0.8; sigma<-0.3
S<-c(1:3)*1e2
p<-0.4; p_prime<-0.2; t1<-1; t2<-2
k<-2; NmonteC<-50

# Here is the continuous H-1/alpha inference procedure
theor_3_1_H_clt<-MCestimLFSM(s=S, fr='H', Nmc=NmonteC,
                           m=m, M=M, alpha=alpha, H=H,
                           sigma=sigma, ContinEstim,
                           t1=t1, t2=t2, p=p, k=k)
Plot_vb(theor_3_1_H_clt)

# More demanding example (it is better to use multicore setup)
# General low frequency inference

m<-45; M<-60
alpha<-0.8; H<-0.8; sigma<-0.3
S<-c(1:15)*1e2
p<-0.4; t1<-1; t2<-2
NmonteC<-50

# Here is the continuous H-1/alpha inference procedure
theor_4_1_H_clt<-MCestimLFSM(s=S, fr='H', Nmc=NmonteC,
                           m=m, M=M, alpha=alpha, H=H,
                           sigma=sigma, GenLowEstim,
                           t1=t1, t2=t2, p=p)
Plot_vb(theor_4_1_H_clt)

```

---

Retrieve\_stats

*Retrieve statistics(bias, variance) of estimators based on a set of paths*


---

**Description**

Retrieve statistics(bias, variance) of estimators based on a set of paths

**Usage**

```
Retrieve_stats(paths, true_val, Est, ...)
```



**Arguments**

|          |  |
|----------|--|
| paths    | real-valued matrix representing sample paths of the stochastic process being studied |
| true_val | true value of the estimated parameter  |
| Est      | estimator (i.e. H_hat)   |
| ...      | parameters to pass to Est  |

**Examples**

```

m<-45; M<-60; N<-2^10-M
alpha<-1.8; H<-0.8; sigma<-0.3
freq='L'; t1=1; t2=2
r=1; k=2; p=0.4

Y<-paths(N_var=10,parallel=TRUE,N=N,m=m,M=M,
         alpha=alpha,H=H,sigma=sigma,freq='L',
         disable_X=FALSE,levy_increments=NULL)

Retrieve_stats(paths=Y,true_val=sigma,Est=sigma_hat,t1=t1,k=2,alpha=alpha,H=H,freq="L")

```

R\_hl

*R high /low***Description**

Defined as

$$R_{\text{high}}(p, k)_n := \frac{\sum_{i=2k}^n \left| \Delta_{i,k}^{n,2} X \right|^p}{\sum_{i=k}^n \left| \Delta_{i,k}^{n,1} X \right|^p},$$

$$R_{\text{low}}(p, k)_n := \frac{\sum_{i=2k}^n \left| \Delta_{i,k}^2 X \right|^p}{\sum_{i=k}^n \left| \Delta_{i,k}^1 X \right|^p}$$

**Usage**

R\_hl(p, k, path)

**Arguments**

|      |   |
|------|---|
| p    | power   |
| k    | increment order   |
| path | sample path of lfsm on which the inference is to be performed |

## Details

The computation procedure for high- and low frequency cases is the same, since there is no way to control frequency given a sample path.

## References

Mazur S, Otryakhin D, Podolskij M (2020). “Estimation of the linear fractional stable motion.” *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

## Examples

```
m<-45; M<-60; N<-2^10-M
alpha<-0.8; H<-0.8; sigma<-0.3
p<-0.3; k=3

lfsm<-path(N=N,m=m,M=M,alpha=alpha,H=H,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm
R_hl(p=p,k=k,path=lfsm)
```

---

 sf

*Statistic V*


---

## Description

Statistic of the form

$$V_{\text{high}}(f; k, r)_n := \frac{1}{n} \sum_{i=rk}^n f \left( n^H \Delta_{i,k}^{n,r} X \right),$$

$$V_{\text{low}}(f; k, r)_n := \frac{1}{n} \sum_{i=rk}^n f \left( \Delta_{i,k}^r X \right)$$

## Usage

```
sf(path, f, k, r, H, freq, ...)
```

## Arguments

|      |  |
|------|--|
| path | sample path for which the statistic is to be calculated. |
| f    | function applied to high order increments.               |
| k    | order of the increments.                                 |
| r    | step of high order increments.                           |
| H    | Hurst parameter.   |
| freq | frequency.   |
| ...  | parameters to pass to function f                         |

## Details

Hurst parameter is required only in high frequency case. In the low frequency, there is no need to assign H a value because it will not be evaluated.

## References

Mazur S, Otryakhin D, Podolskij M (2020). “Estimation of the linear fractional stable motion.” *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

## See Also

[phi](#) computes V statistic with  $f(\cdot)=\cos(t)$

## Examples

```
m<-45; M<-60; N<-2^10-M
alpha<-1.8; H<-0.8; sigma<-0.3
freq='L'
r=1; k=2; p=0.4
S<-(1:20)*100

path_lfsm<-function(...){

  List<-path(...)
  List$lfsm

}

Pths<-lapply(X=S,FUN=path_lfsm,
            m=m, M=M, alpha=alpha, sigma=sigma, H=H,
            freq=freq, disable_X = FALSE,
            levy_increments = NULL, seed = NULL)

f_phi<-function(t,x) cos(t*x)
f_pow<-function(x,p) (abs(x))^p

V_cos<-sapply(Pths,FUN=sf,f=f_phi,k=k,r=r,H=H,freq=freq,t=1)
ex<-exp(-(abs(sigma*Norm_alpha(h_kr,alpha=alpha,k=k,r=r,H=H,l=0)$result)^alpha))

# Illustration of the law of large numbers for phi:
plot(y=V_cos, x=S, ylim = c(0,max(V_cos)+0.1))
abline(h=ex, col='brown')

# Illustration of the law of large numbers for power functions:
Mpk<-m_pk(k=k, p=p, alpha=alpha, H=H, sigma=sigma)

sf_mod<-function(Xpath,...) {

  Path<-unlist(Xpath)
  sf(path=Path,...)

}
```

```
V_pow<-sapply(Pths,FUN=sf_mod,f=f_pow,k=k,r=r,H=H,freq=freq,p=p)
plot(y=V_pow, x=S, ylim = c(0,max(V_pow)+0.1))
abline(h=Mpk, col='brown')
```

---

|           |  |
|-----------|--|
| sigma_hat | <i>Statistical estimator for sigma</i> |
|-----------|--|

---

### Description

Statistical estimator for sigma

### Usage

```
sigma_hat(t1, k, path, alpha, H, freq)
```

### Arguments

|       |  |
|-------|--|
| t1    | real number such that $t_2 > t_1 > 0$  |
| k     | increment order  |
| path  | sample path of lfsm on which the inference is to be performed  |
| alpha | self-similarity parameter of alpha stable random motion.   |
| H     | Hurst parameter  |
| freq  | Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting. |

### Examples

```
m<-45; M<-60; N<-2^14-M
alpha<-1.8; H<-0.8; sigma<-0.3
freq='H'
r=1; k=2; p=0.4; t1=1; t2=2

# Reproducing the work of ContinEstim
# in high frequency case
lfsm<-path(N=N,m=m,M=M,alpha=alpha,H=H,
           sigma=sigma,freq='L',disable_X=FALSE,seed=1)$lfsm

H_est<-H_hat(p=p,k=k,path=lfsm)
H_est
alpha_est<-alpha_hat(t1=t1,t2=t2,k=k,path=lfsm,H=H_est,freq=freq)
alpha_est

sigma_est<-tryCatch(
  sigma_hat(t1=t1,k=k,path=lfsm,
            alpha=alpha_est,H=H_est,freq=freq),
  error=function(c) 'Impossible to compute sigma_est')
sigma_est
```

theta

*Function of the form*

$$\theta(g, h)_p = a_p^{-2} \int_{\mathbb{R}^2} |xy|^{-1-p} U_{g, h}(x, y) dx dy$$

**Description**

Function of the form

$$\theta(g, h)_p = a_p^{-2} \int_{\mathbb{R}^2} |xy|^{-1-p} U_{g, h}(x, y) dx dy$$

**Usage**

theta(p, alpha, sigma, g, h)

**Arguments**

|       |  |
|-------|--|
| p     | power, real number from (-1,1)   |
| alpha | self-similarity parameter of alpha stable random motion.   |
| sigma | Scale parameter of lfsm  |
| g     | functions $g, h : \mathbb{R} \text{ to } \mathbb{R}$ with finite alpha-norm (see <a href="#">Norm_alpha</a> ). |
| h     | functions $g, h : \mathbb{R} \text{ to } \mathbb{R}$ with finite alpha-norm (see <a href="#">Norm_alpha</a> ). |

**References**

Mazur S, Otryakhin D, Podolskij M (2020). “Estimation of the linear fractional stable motion.” *Bernoulli*, **26**(1), 226–252. doi: [10.3150/19BEJ1124](https://doi.org/10.3150/19BEJ1124), <https://doi.org/10.3150/19-BEJ1124>.

U\_g

*alpha norm of u\*g***Description**

alpha norm of u\*g

**Usage**

U\_g(g, u, ...)

**Arguments**

|     |  |
|-----|--|
| g   | functions $g, h : \mathbb{R} \text{ to } \mathbb{R}$ with finite alpha-norm (see <a href="#">Norm_alpha</a> ). |
| u   | real numbers   |
| ... | additional parameters to pass to <a href="#">Norm_alpha</a>  |

**Examples**

```
g<-function(x) exp(-x^2)
g<-function(x) exp(-abs(x))
U_g(g=g,u=4,alpha=1.7)
```

---

U\_gh *alpha-norm of u\*g + v\*h.*

---

**Description**

alpha-norm of  $u*g + v*h$ .

**Usage**

```
U_gh(g, h, u, v, ...)
```

**Arguments**

g functions  $g, h : \mathbb{R} \text{ to } \mathbb{R}$  with finite alpha-norm (see [Norm\\_alpha](#)).

h functions  $g, h : \mathbb{R} \text{ to } \mathbb{R}$  with finite alpha-norm (see [Norm\\_alpha](#)).

u real numbers

v real numbers

... additional parameters to pass to [Norm\\_alpha](#)

**Examples**

```
g<-function(x) exp(-x^2)
h<-function(x) exp(-abs(x))
U_gh(g=g, h=h, u=4, v=3, alpha=1.7)
```

---

U\_ghuv *A dependence structure of 2 random variables.*

---

**Description**

It is used when random variables do not have finite second moments, and thus, the covariance matrix is not defined. For  $X = \int_{\mathbb{R}} g_s dL_s$  and  $Y = \int_{\mathbb{R}} h_s dL_s$  with  $\|g\|_{\alpha}, \|h\|_{\alpha} < \infty$ . Then the measure of dependence is given by  $U_{g,h} : \mathbb{R}^2 \text{ to } \mathbb{R}$  via

$$U_{g,h}(u, v) = \exp(-\sigma^{\alpha} \|ug + vh\|_{\alpha}^{\alpha}) - \exp(-\sigma^{\alpha} (\|ug\|_{\alpha}^{\alpha} + \|vh\|_{\alpha}^{\alpha}))$$

**Usage**

```
U_ghuv(alpha, sigma, g, h, u, v, ...)
```

**Arguments**

|                    |  |
|--------------------|--|
| <code>alpha</code> | self-similarity parameter of alpha stable random motion.   |
| <code>sigma</code> | Scale parameter of lfsm  |
| <code>g, h</code>  | functions $g, h : \mathbb{R} \rightarrow \mathbb{R}$ with finite alpha-norm (see <a href="#">Norm_alpha</a> ). |
| <code>v, u</code>  | real numbers   |
| <code>...</code>   | additional parameters to pass to <code>U_gh</code> and <code>U_g</code>  |

**Examples**

```
g<-function(x) exp(-x^2)
h<-function(x) exp(-abs(x))
U_ghuv(alpha=1.5, sigma=1, g=g, h=h, u=10, v=15,
rel.tol = .Machine$double.eps^0.25, abs.tol=1e-11)
```

# Index

a\_p, 4  
a\_tilda, 4  
a\_tilda\_R(a\_tilda), 4  
alpha\_hat, 2

CLT, 5  
ContinEstim, 6

expand.grid, 20

GenHighEstim, 7  
GenLowEstim, 8

H\_hat, 9, 14  
h\_kr, 10

increment, 11, 14  
increments(increment), 11

m\_pk, 15  
MCestimLFSM, 5, 12, 22, 23  
MinContrastEstim, 14

Norm\_alpha, 16, 29–31

path, 17, 19  
Path\_array, 20, 23  
paths, 18, 19  
phi, 3, 14, 20, 27  
phi\_of\_alpha, 21  
Plot\_dens, 22, 23  
Plot\_list\_paths, 20, 23  
Plot\_vb, 22, 23

R\_h1, 25  
Retrieve\_stats, 24

sf, 26  
sigma\_hat, 28

theta, 29

U\_g, 29  
U\_gh, 30  
U\_ghuv, 30  
uniroot, 7