Package: levyBridge (via r-universe)

November 29, 2026

Title Stochastic Bridges for Levy Degradation Processes

Version 0.1.0

Description Exact simulation of the bridges of Levy degradation processes (gamma, Wiener, inverse Gaussian, variance-gamma and Meixner processes) and resampling of trajectories observed at regular or irregular inspection times onto a regular time grid, following the interpolation algorithm of Grall-Maes (2012) <doi:10.1177/1748006X12462522>.

License GPL-3

URL https://github.com/rytracetito/levy_packages/tree/main/levyBridge, https://github.com/rytracetito/levy_packages

BugReports https://github.com/rytracetito/levy_packages/issues

Encoding UTF-8

Language en

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests DegradSim, statmod, testthat (>= 3.0.0)

Remotes github::rytracetito/levy_packages/levyfit

Config/testthat/edition 3

Imports ggplot2, GIGrvg, gsl, levyfit, Runuran, stats, utils

Config/pak/sysreqs libgsl0-dev

Repository https://rytracetito.r-universe.dev

Date/Publication 2026-11-29 04:07:38 UTC

RemoteUrl https://github.com/rytracetito/levy_packages

RemoteRef HEAD

RemoteSha 87869bea20605b72514e4cbc3e354acda28553a8

RemoteSubdir levyBridge

Contents

levyBridge-package . . . 2
gamma_bridge . . . 7
gamma_resampling . . . 9
ig_bridge . . . 12
ig_resampling . . . 14
meixner_bridge . . . 17
meixner_resampling . . . 20
plot_bridge . . . 23
vg_bridge . . . 24
vg_resampling . . . 27
wiener_bridge . . . 30
wiener_resampling . . . 31

Index **35**

`levyBridge-package` *levyBridge: stochastic bridges for Levy degradation processes*

Description

`levyBridge` simulates exactly the bridges of Levy degradation processes, and uses them to resample degradation trajectories, observed at regular or irregular inspection times, onto a regular time grid.

Details

The package has two kinds of functions, for each process:

- the **bridge** functions `gamma_bridge()`, `wiener_bridge()`, `ig_bridge()`, `vg_bridge()` and `meixner_bridge()` draw the value X_t of the process at one time t, given its levels at two times $t_A < t < t_B$;
- the **resampling** functions `gamma_resampling()`, `wiener_resampling()`, `ig_resampling()`, `vg_resampling()` and `meixner_resampling()` resample whole trajectories onto a regular grid, by calling the bridge function at each grid point.

All these functions take the vector Θ of all the parameters of the process, as a named vector `theta`, for instance `c(a = 2, b = 3)` for the gamma process.

Bridge of a Levy process

Let X be a Levy process such that X_s has the density f_s, and let $t_A < t < t_B$. Given $X_{t_A} = x_A$ and $X_{t_B} = x_B$, the value X_t has the density

$$f_{t|t_A,t_B}(x) = \frac{f_{t-t_A}(x - x_A)\, f_{t_B-t}(x_B - x)}{f_{t_B-t_A}(x_B - x_A)}$$

(Hoyle, Hughston and Macrina, 2011). The bridge functions draw exactly from this distribution. The functions of each process are:

Process	Bridge	Resampling
gamma	`gamma_bridge()`	`gamma_resampling()`
Wiener	`wiener_bridge()`	`wiener_resampling()`
inverse Gaussian	`ig_bridge()`	`ig_resampling()`
variance-gamma	`vg_bridge()`	`vg_resampling()`
Meixner	`meixner_bridge()`	`meixner_resampling()`

and the parameters in Θ (argument `theta`) are:

Process	`theta`	Used by the bridge
gamma	`a`, `b`	`a`
Wiener	`mu`, `sigma`	`sigma`
inverse Gaussian	`mu`, `eta`	`eta`
variance-gamma	`theta`, `sigma`, `nu`	all
Meixner	`a`, `b`, `d`, `m`	`a`, `d`, `m`

The distribution of the bridge does not always depend on all the parameters: the parameters that it does not use are ignored by the bridge functions, and used by the resampling functions only to extend the trajectories (option 2 of the method `"MultiBridge"`). The help page of each bridge function gives the distribution of its bridge and the way it is drawn.

Data

The argument `x` of the resampling functions accepts three formats:

- **one trajectory**: a data frame whose first column contains the inspection times and second column the degradation levels;
- **several trajectories, each with its own inspection times**: a list of such data frames, one per trajectory, for instance the result of a simulation function of `DegradSim` when the trajectories have different inspection times;
- **several trajectories with the same inspection times**: a matrix whose first column contains the times and the other columns the levels, one column per trajectory, for instance the result of a simulation function of `DegradSim` when all the trajectories have the same inspection times.

The columns are read by position, not by name. The inspection times of a trajectory start at 0 and are strictly increasing.

Exact increments. The observed increment between two inspections is the difference of the two levels. A level only keeps about 16 significant digits, so that an increment very small compared with the level (for instance a gamma increment over a short time step) can be found equal to 0. The simulation functions of `DegradSim` keep the increments exactly as drawn, in `attr(x, "dx")`: when `x` has this attribute, a list with one vector of increments per trajectory whose cumulated sums give the levels, these increments are used instead of the differences of the levels. The resampling is otherwise the same.

Resampling algorithm

Let $0 = t_0 < t_1 < \cdots < t_N$ be the inspection times of a trajectory and $x_0, \ldots, x_N$ its observed levels. The method (section Resampling methods) gives a regular grid $0 = t'_0 < t'_1 < \ldots$, and the resampled levels x'_k are computed in increasing order of k:

- if t'_k is an inspection time, x'_k is the observed level;
- otherwise, x'_k is drawn by the bridge function between the last known point on the left (the inspection time or the grid point t'_{k-1}, whichever is closer to t'_k) and the next inspection time on the right.

This is the interpolation algorithm of Grall-Maës (2012), proposed for the gamma process and used here for all the processes.

A grid point that differs from an inspection time only by a rounding error of the computer (relative difference below 10^{-9}) is set to that inspection time.

Resampling methods

Every resampling function has an argument `method`, which chooses the grid.

`"Bridge"` (default). The grid of Grall-Maës (2012):

$$t'_k = k\,\frac{t_N}{N}, \qquad k = 0, \ldots, N,$$

with as many points as observations and the same end points. Each trajectory has its own grid, of step t_N/N.

`"ExtBridge"`. A grid that contains all the inspection times. The times are taken with d decimal places (argument `digits`) and

$$h = 10^{-d}\,\mathrm{gcd}\big(\lfloor 10^d t_1 \rfloor, \ldots, \lfloor 10^d t_N \rfloor\big), \qquad t'_k = k\,h, \quad k = 0, \ldots, t_N/h,$$

where $\lfloor u \rfloor$ is the largest integer not greater than u. The step h is the largest one such that every inspection time, truncated to d decimal places, is a grid point: all the observations are kept. The grid has $t_N/h \geq N$ steps, with equality if and only if the inspection times are regular; the grid is then the inspection times and the data are returned unchanged.

With `digits = NULL` (default), d is the number of decimal places of the inspection times, that is the smallest integer d such that all the times are multiples of 10^{-d}, up to rounding errors of the computer. This package looks for d between 0 and 6, a limit chosen to avoid very fine grids; if the times have more decimal places, an error asks for `digits`. When `digits` is given and some times have more decimal places, they are truncated to `digits` decimal places, with a warning; two times equal after truncation give an error. A warning is also given when the grid has more than 10^5 steps, since the resampling then takes time.

Each trajectory has its own step h. With `common_step = TRUE`, d and h are computed from the inspection times of all the trajectories, which then have the same step.

`"MultiBridge"`. One step common to all the trajectories: the trajectory i, observed on $[0, t_{i,max}]$ with m_i increments, is resampled on $t'_k = kh$ with

$$h = \frac{t_{1,max} + \cdots + t_{n,max}}{m_1 + \cdots + m_n},$$

the total observed duration divided by the total number of increments. Grall-Maës (2012, eq. (7)) proposes this step for two trajectories. The number of reconstructed increments is then about the number of observed ones; for one trajectory, $h = t_N/N$ and the method is `"Bridge"`. The grid does not necessarily contain the inspection times.

- `option = 1` (default): the trajectory i is resampled on the grid points $kh \leq t_{i,max}$, that is up to its own last inspection. The end of the trajectory, between its last grid point and $t_{i,max}$, is not used. A trajectory shorter than h gives no increment, with a warning.
- `option = 2`: all the trajectories are resampled up to the last inspection time over all of them, $T = \max_i t_{i,max}$. The grid ends exactly at T: the number of steps T/h is rounded to the nearest integer N, and the step becomes $h^* = T/N$, close to h; the grid is $t'_k = kh^*$, $k = 0, \ldots, N$, the same for all the trajectories, and `attr(res, "teq")` gives h^*. Beyond its last inspection, a trajectory is extended by increments of the process drawn with the given parameters, starting from its last observed level. This needs all the parameters of the process (for instance the rate `b` of the gamma process), and the extended values do not come from the data.

Choice of the method.

- `"Bridge"`: one grid per trajectory, with as many points as observations; the steps of two trajectories are different in general.
- `"ExtBridge"`: all the observations are kept on the grid, which can have many more points than observations.
- `"MultiBridge"`, or `"ExtBridge"` with `common_step = TRUE`: the same step for all the trajectories.

Results

A resampling function returns an object of class `"levybridge"`, a list with one data frame per trajectory:

- `res[[k]]` is the k-th trajectory on its regular grid, with the columns `t` (grid) and `X` (resampled levels);
- `attr(res, "teq")` gives the step of the grid of each trajectory;
- `attr(res, "dx")` gives, for each trajectory, the increments between consecutive grid points. They are computed directly by the bridges, not as differences of the resampled levels: a very small increment keeps its significant digits instead of being rounded to 0;
- `attr(res, "observed")` gives the observed data, one data frame per trajectory (with the method `"ExtBridge"`, the truncated times when a truncation was needed);
- `attr(res, "model")`, `attr(res, "parameters")` and `attr(res, "method")` recall the call.

Printing `res` shows the model, the parameters, the steps and the first resampled levels. The function `plot_bridge()` compares the observed and resampled levels. The bridge values are random: use `set.seed()` before a resampling to make it reproducible.

Author(s)

Maintainer: Amedée Rytrace Tito <rytracetito5@gmail.com>

Authors:

- Christian Paroissin <christian.paroissin@univ-pau.fr>

References

Grall-Maës, E. (2012). Use of the Kolmogorov-Smirnov test for gamma process. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 226(6), 624-634. doi:10.1177/1748006X12462522

Hoyle, E., Hughston, L. P. and Macrina, A. (2011). Lévy random bridges and the modelling of financial information. *Stochastic Processes and their Applications*, 121(4), 856-884.

See Also

Useful links:

- https://github.com/rytracetito/levy_packages/tree/main/levyBridge
- https://github.com/rytracetito/levy_packages
- Report bugs at https://github.com/rytracetito/levy_packages/issues

Examples

```
# 1. The bridge: draws of X_2 given X_1 = 3 and X_4 = 6 (gamma, a = 2, b = 3)
set.seed(1)
gamma_bridge(t_A = 1, x_A = 3, t_B = 4, x_B = 6, t = 2, theta = c(a = 2, b = 3), n = 5)

# 2. Resampling of one gamma trajectory observed at 11 irregular times
t <- c(0, 1.2, 2, 3.8, 5.5, 7.1, 10.4, 12.9, 15.3, 18.7, 22.5)
x <- DegradSim::sim_gamma(t, a = 2, b = 3)

# on the regular grid 0, 2.25, 4.5, ..., 22.5
r <- gamma_resampling(x, theta = c(a = 2, b = 3))
r
r[[1]]
attr(r, "teq")
attr(r, "dx")
plot_bridge(r)

# 3. Five trajectories with the same inspection times (a matrix)
xm <- DegradSim::sim_gamma(t, a = 2, b = 3, n_traj = 5)
head(xm)
plot_bridge(gamma_resampling(xm, theta = c(a = 2, b = 3)), trajectory = 1:5)

# 4. Three trajectories, each with its own inspection times (a list)
plan <- DegradSim::inspection_times("decreasing", n = 10, n_traj = 3,
                                    tmax = c(20, 30, 40), digits = 1)
xs <- DegradSim::sim_gamma(plan, a = 2, b = 3)

# One grid per trajectory, then the two methods with a common step
theta <- c(a = 2, b = 3)
attr(gamma_resampling(xs, theta), "teq")
attr(gamma_resampling(xs, theta, method = "ExtBridge", common_step = TRUE), "teq")
attr(gamma_resampling(xs, theta, method = "MultiBridge"), "teq")
plot_bridge(gamma_resampling(xs, theta, method = "MultiBridge"), trajectory = 1:3)

# "MultiBridge", option 2: the grid ends exactly at the last inspection
```

```
# time (40), and the shorter trajectories are extended up to it
r2 <- gamma_resampling(xs, theta, method = "MultiBridge", option = 2)
sapply(r2, function(d) max(d$t))
plot_bridge(r2, trajectory = 1:3)
```

`gamma_bridge` *Gamma bridge*

Description

Draws the value at time t of a gamma process known at two times $t_A < t < t_B$, from the distribution of the gamma bridge. This function is used by `gamma_resampling()` at each point of the grid.

Usage

```
gamma_bridge(t_A, x_A, t_B, x_B, t, theta, n = 1)
```

Arguments

`t_A`, `x_A`	Time and level of the known point on the left.
`t_B`, `x_B`	Time and level of the known point on the right, $t_B > t_A$.
`t`	Time of the draw, $t_A < t < t_B$.
`theta`	Vector $\Theta = (a, b)$ of the parameters of the process, a numeric vector named `a` and `b`, for instance the estimate returned by `gamma_mle()` of the package `levyfit`. The bridge uses only `a`.
`n`	Number of independent draws, a positive integer (default 1).

Value

A numeric vector of `n` independent draws of X_t given $X_{t_A} = x_A$ and $X_{t_B} = x_B$.

Model

The gamma process with shape $a > 0$ and rate $b > 0$ has $X_0 = 0$, independent increments and $X_t - X_s \sim \Gamma(a(t-s), b)$ for $s < t$ (shape $a(t-s)$, rate b). Its trajectories are increasing. The vector of its parameters is $\Theta = (a, b)$.

Distribution of the bridge

Given $X_{t_A} = x_A$ and $X_{t_B} = x_B$,

$$X_t = x_A + (x_B - x_A)\, Y, \qquad Y \sim \mathrm{Beta}\big(a(t-t_A),\ a(t_B-t)\big)$$

(Grall-Maës, 2012, eq. (3)). This distribution does not depend on the rate b. When $x_B = x_A$ (two equal levels, which can happen after rounding), the bridge is constant.

References

Grall-Maës, E. (2012). Use of the Kolmogorov-Smirnov test for gamma process. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 226(6), 624-634. doi:10.1177/1748006X12462522

See Also

- gamma_resampling(): resampling of trajectories on a regular grid;
- wiener_bridge(), ig_bridge(), vg_bridge(), meixner_bridge(): the bridges of the other processes.

Examples

```
# 1. One draw of X_2, given X_1 = 3 and X_4 = 6 (a = 2, b = 3)
set.seed(1)
gamma_bridge(t_A = 1, x_A = 3, t_B = 4, x_B = 6, t = 2, theta = c(a = 2, b = 3))

# 2. Many draws: the histogram of the draws and, in red, the density of
#    the bridge, f_{t - t_A}(x - x_A) f_{t_B - t}(x_B - x) / f_{t_B - t_A}(x_B - x_A)
#    (see ?levyBridge), where f_s is the density of X_s
f <- function(x, s) dgamma(x, shape = 2 * s, rate = 3)
dens <- function(x) f(x - 3, 1) * f(6 - x, 2) / f(3, 3)
y <- gamma_bridge(1, 3, 4, 6, 2, theta = c(a = 2, b = 3), n = 10000)
hist(y, breaks = 50, freq = FALSE)
curve(dens, add = TRUE, col = "red")

# The rate b does not change the bridge
set.seed(2); y1 <- gamma_bridge(1, 3, 4, 6, 2, theta = c(a = 2, b = 3), n = 5)
set.seed(2); y2 <- gamma_bridge(1, 3, 4, 6, 2, theta = c(a = 2, b = 100), n = 5)
identical(y1, y2)

# 3. Equal levels give a constant bridge; a decrease is impossible
gamma_bridge(0, 1, 2, 1, 0.5, theta = c(a = 2, b = 3))
try(gamma_bridge(0, 2, 2, 1, 0.5, theta = c(a = 2, b = 3)))

# Errors: t outside (t_A, t_B), theta without all the parameters
try(gamma_bridge(1, 3, 4, 6, 5, theta = c(a = 2, b = 3)))
try(gamma_bridge(1, 3, 4, 6, 2, theta = c(a = 2)))

# 4. Parameters estimated with levyfit on a simulated trajectory
t <- c(0, 1.2, 2, 3.8, 5.5, 7.1, 10.4, 12.9, 15.3, 18.7, 22.5)
x <- DegradSim::sim_gamma(t, a = 2, b = 3)
est <- levyfit::gamma_mle(x)
est
# Three draws at t = 1.5, between the inspections 1.2 and 2
gamma_bridge(1.2, x[2, 2], 2, x[3, 2], 1.5, theta = est, n = 3)
```

gamma_resampling *Gamma process: resampling on a regular grid*

Description

Resamples one or several trajectories of a gamma process onto a regular time grid: each grid point that is not an inspection time is drawn with `gamma_bridge()` between the known levels around it. The grids and the resampling algorithm are described in levyBridge-package.

Usage

```
gamma_resampling(
  x,
  theta,
  method = "Bridge",
  digits = NULL,
  common_step = FALSE,
  option = 1
)
```

Arguments

`x`	The observed trajectories: a data frame whose first column contains the inspection times (starting at 0, strictly increasing) and second column the degradation levels; a list of such data frames, one per trajectory; or a matrix whose first column contains the times and the other columns the levels, one column per trajectory. The results of the simulation functions of `DegradSim` are accepted. See the section Data of levyBridge-package.
`theta`	Vector $\Theta = (a, b)$ of the parameters of the process, a numeric vector named `a` and `b`, for instance the estimate returned by `gamma_mle()` of the package `levyfit`. The bridge uses only `a`; `b` is used by the option 2 of the method `"MultiBridge"`, to extend the trajectories.
`method`	Grid of the resampling: `"Bridge"` (default), `"ExtBridge"` or `"MultiBridge"`, see the section Resampling methods of levyBridge-package.
`digits`	Used only with `method = "ExtBridge"`: number of decimal places of the inspection times, which sets the step of the grid. `NULL` (default) takes the number of decimal places of the times.
`common_step`	Used only with `method = "ExtBridge"`: `FALSE` (default) gives each trajectory its own step, `TRUE` gives the same step to all the trajectories.
`option`	Used only with `method = "MultiBridge"`: `1` (default) to resample each trajectory up to its own last inspection, `2` to resample all the trajectories up to the last inspection time over all of them, extending the shorter ones by increments of the process.

Value

An object of class "levybridge": a list with one data frame per trajectory, with the columns t (regular grid) and X (resampled levels), and the attributes teq (step of each grid), dx (increments between consecutive grid points), observed, model, parameters and method. See the section Results of levyBridge-package.

References

Grall-Maës, E. (2012). Use of the Kolmogorov-Smirnov test for gamma process. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 226(6), 624-634. doi:10.1177/1748006X12462522

See Also

gamma_bridge() for the distribution of the bridge, levyBridge-package for the grids and the algorithm, plot_bridge().

Examples

```
# 1. One trajectory observed at 11 irregular times (a = 2, b = 3)
t <- c(0, 1.2, 2, 3.8, 5.5, 7.1, 10.4, 12.9, 15.3, 18.7, 22.5)
set.seed(1)
x <- DegradSim::sim_gamma(t, a = 2, b = 3)
theta <- c(a = 2, b = 3)
r <- gamma_resampling(x, theta)
r

# The resampled trajectory, on the regular grid 0, 2.25, 4.5, ..., 22.5
r[[1]]
attr(r, "teq")
plot_bridge(r)

# 2. One trajectory given as a data frame (times, then levels): the grid is
#    0, 2, 4, 6, 8, 10, and the inspection times 2 and 6 keep their levels
x2 <- data.frame(t = c(0, 2, 3, 6, 7, 10), X = c(0, 1.4, 2.0, 4.1, 4.6, 6.8))
gamma_resampling(x2, theta)[[1]]

# Equal observed levels give a constant bridge; a decrease is impossible
gamma_resampling(data.frame(t = c(0, 1, 2.5, 3), X = c(0, 1, 1, 2)), theta)[[1]]
try(gamma_resampling(data.frame(t = c(0, 1, 2, 3), X = c(0, 2, 1.5, 3)), theta))

# 3. Method "ExtBridge": the times have 1 decimal place, so the grid
#    0, 0.1, ..., 22.5 contains all of them
r3 <- gamma_resampling(x, theta, method = "ExtBridge")
attr(r3, "teq")
all(t %in% r3[[1]]$t)
plot_bridge(r3)

# digits = 0 truncates the times to integers, with a warning
attr(gamma_resampling(x, theta, method = "ExtBridge", digits = 0), "teq")
```

```
# 4. Five trajectories with the same inspection times (a matrix)
xm <- DegradSim::sim_gamma(t, a = 2, b = 3, n_traj = 5)
head(xm)
r4 <- gamma_resampling(xm, theta)
attr(r4, "teq")
plot_bridge(r4, trajectory = 1:5)

# 5. Three trajectories, each with its own inspection times (a list)
plan <- DegradSim::inspection_times("decreasing", n = 10, n_traj = 3,
                                    tmax = c(20, 30, 40), digits = 1)
xs <- DegradSim::sim_gamma(plan, a = 2, b = 3)

# One grid and one step per trajectory
attr(gamma_resampling(xs, theta), "teq")
attr(gamma_resampling(xs, theta, method = "ExtBridge"), "teq")

# The same step for all the trajectories
attr(gamma_resampling(xs, theta, method = "ExtBridge", common_step = TRUE), "teq")
r5 <- gamma_resampling(xs, theta, method = "MultiBridge")
attr(r5, "teq")
sapply(r5, nrow)
plot_bridge(r5, trajectory = 1:3)

# Option 2: all the trajectories up to 40; beyond its last inspection, each
#    one is extended by increments of the process (this uses b)
r6 <- gamma_resampling(xs, theta, method = "MultiBridge", option = 2)
sapply(r6, nrow)
plot_bridge(r6, trajectory = 1:3)

# 6. The bridge values are random: set.seed() makes them reproducible
set.seed(2); y1 <- gamma_resampling(x, theta)[[1]]$X
set.seed(2); y2 <- gamma_resampling(x, theta)[[1]]$X
identical(y1, y2)

# 7. With the parameters estimated by levyfit
theta_hat <- levyfit::gamma_mle(xs)
theta_hat
r7 <- gamma_resampling(xs, theta_hat)
attr(r7, "parameters")

# 8. Four trajectories observed up to 8, 12, 45 and 58, each with its own
#    inspection times, resampled with "MultiBridge"
liste_t <- list(c(0, 3, 8), c(0, 2, 5, 9, 12),
                c(0, 4, 7, 11, 15, 18, 22, 27, 33, 39, 45),
                c(0, 2, 3, 5, 8, 9, 14, 15, 17, 20, 22, 25, 27, 30, 33, 46, 58))

# Gamma process (a = 5, b = 1/3), parameters estimated by levyfit
set.seed(46)
x8 <- DegradSim::sim_gamma(liste_t, a = 5, b = 1/3)
theta8 <- levyfit::gamma_mle(x8)

# Option 1: one grid for all the trajectories; each one stops at the last
# grid point before its last inspection
```

```
r8 <- gamma_resampling(x8, theta8, method = "MultiBridge")
attr(r8, "teq")
sapply(r8, function(d) max(d$t))
plot_bridge(r8, trajectory = 1:4)

# Option 2: the grid ends exactly at 58; beyond its last inspection, each
# shorter trajectory is extended by increments of the process
r8b <- gamma_resampling(x8, theta8, method = "MultiBridge", option = 2)
attr(r8b, "teq")
sapply(r8b, function(d) max(d$t))
plot_bridge(r8b, trajectory = 1:4)
```

`ig_bridge` *Inverse Gaussian bridge*

Description

Draws the value at time t of an inverse Gaussian process known at two times $t_A < t < t_B$, from the distribution of the inverse Gaussian bridge. This function is used by `ig_resampling()` at each point of the grid.

Usage

```
ig_bridge(t_A, x_A, t_B, x_B, t, theta, n = 1)
```

Arguments

`t_A`, `x_A`	Time and level of the known point on the left.
`t_B`, `x_B`	Time and level of the known point on the right, $t_B > t_A$.
`t`	Time of the draw, $t_A < t < t_B$.
`theta`	Vector $\Theta = (\mu, \eta)$ of the parameters of the process, a numeric vector named `mu` and `eta`, for instance the estimate returned by `ig_mle()` of the package `levyfit`. The bridge uses only `eta`.
`n`	Number of independent draws, a positive integer (default 1).

Value

A numeric vector of `n` independent draws of X_t given $X_{t_A} = x_A$ and $X_{t_B} = x_B$.

Model

The inverse Gaussian process with parameters $\mu > 0$ and $\eta > 0$ has $X_0 = 0$, independent increments and $X_t - X_s \sim IG(\mu(t-s), \eta(t-s)^2)$ (mean $\mu(t-s)$, shape $\eta(t-s)^2$). Its trajectories are increasing. The vector of its parameters is $\Theta = (\mu, \eta)$.

Distribution of the bridge

Let $x_B > x_A$. Given $X_{t_A} = x_A$ and $X_{t_B} = x_B$, the ratio $W = (X_t - x_A)/(x_B - X_t)$ has the density

$$f_W(w) = \sqrt{\frac{\lambda}{2\pi}}\,\frac{1}{1+m}\,w^{-3/2}(1+w)\exp\Big(-\frac{\lambda(w-m)^2}{2m^2 w}\Big), \qquad w > 0,$$

where $m = (t - t_A)/(t_B - t)$ and $\lambda = \eta(t - t_A)^2/(x_B - x_A)$. Ribeiro and Webber (2003, section 3.3, eq. (27)) give this density for the inverse ratio $S = 1/W$. Splitting the factor $1 + w$, it is the mixture

$$W \sim IG(m,\lambda) \text{ with probability } \frac{t_B - t}{t_B - t_A}, \qquad W \sim \frac{m^2}{IG(m,\lambda)} \text{ with probability } \frac{t - t_A}{t_B - t_A};$$

then $X_t = x_A + (x_B - x_A)\,W/(1 + W)$. This distribution does not depend on μ. When $x_B = x_A$ (two equal levels, which can happen after rounding), the bridge is constant.

The variable W is drawn from one Gaussian and one uniform variable, by the method of Michael, Schucany and Haas (1976) as adapted by Ribeiro and Webber (2003, sections 3.3 and 3.4).

References

Michael, J. R., Schucany, W. R. and Haas, R. W. (1976). Generating random variates using transformations with multiple roots. *The American Statistician*, 30(2), 88-90. doi:10.1080/00031305.1976.10479147

Ribeiro, C. and Webber, N. (2003). A Monte Carlo method for the normal inverse Gaussian option valuation model using an inverse Gaussian bridge. *Computing in Economics and Finance 2003*, 5.

See Also

- `ig_resampling()`: resampling of trajectories on a regular grid;
- `gamma_bridge()`, `wiener_bridge()`, `vg_bridge()`, `meixner_bridge()`: the bridges of the other processes.

Examples

```
# 1. One draw of X_2, given X_1 = 3 and X_4 = 6 (mu = 1, eta = 3)
set.seed(1)
ig_bridge(t_A = 1, x_A = 3, t_B = 4, x_B = 6, t = 2, theta = c(mu = 1, eta = 3))

# 2. Many draws: the histogram of the draws and, in red, the density of
#    the bridge, f_{t - t_A}(x - x_A) f_{t_B - t}(x_B - x) / f_{t_B - t_A}(x_B - x_A)
#    (see ?levyBridge), where f_s is the density of X_s
#    (inverse Gaussian with mean mu s = s and shape eta s^2 = 3 s^2)
f <- function(x, s) sqrt(3 * s^2 / (2 * pi * x^3)) * exp(-3 * (x - s)^2 / (2 * x))
dens <- function(x) f(x - 3, 1) * f(6 - x, 2) / f(3, 3)
y <- ig_bridge(1, 3, 4, 6, 2, theta = c(mu = 1, eta = 3), n = 10000)
range(y)
hist(y, breaks = 50, freq = FALSE)
curve(dens, from = 3.001, to = 5.999, add = TRUE, col = "red")
```

```
# 3. Equal levels give a constant bridge; a decrease is impossible
ig_bridge(0, 1, 2, 1, 0.5, theta = c(mu = 1, eta = 3))
try(ig_bridge(0, 2, 2, 1, 0.5, theta = c(mu = 1, eta = 3)))

# Errors: t outside (t_A, t_B), theta without all the parameters
try(ig_bridge(1, 3, 4, 6, 5, theta = c(mu = 1, eta = 3)))
try(ig_bridge(1, 3, 4, 6, 2, theta = c(eta = 3)))

# 4. Parameters estimated with levyfit on a simulated trajectory
t <- c(0, 1.2, 2, 3.8, 5.5, 7.1, 10.4, 12.9, 15.3, 18.7, 22.5)
x <- DegradSim::sim_ig(t, mu = 1, eta = 3)
est <- levyfit::ig_mle(x)
ig_bridge(1.2, x[2, 2], 2, x[3, 2], 1.5, theta = est, n = 3)
```

`ig_resampling` *Inverse Gaussian process: resampling on a regular grid*

Description

Resamples one or several trajectories of an inverse Gaussian process onto a regular time grid: each grid point that is not an inspection time is drawn with `ig_bridge()` between the known levels around it. The grids and the resampling algorithm are described in levyBridge-package.

Usage

```
ig_resampling(
  x,
  theta,
  method = "Bridge",
  digits = NULL,
  common_step = FALSE,
  option = 1
)
```

Arguments

`x`	The observed trajectories: a data frame whose first column contains the inspection times (starting at 0, strictly increasing) and second column the degradation levels; a list of such data frames, one per trajectory; or a matrix whose first column contains the times and the other columns the levels, one column per trajectory. The results of the simulation functions of `DegradSim` are accepted. See the section Data of levyBridge-package.
`theta`	Vector $\Theta = (\mu, \eta)$ of the parameters of the process, a numeric vector named `mu` and `eta`, for instance the estimate returned by `ig_mle()` of the package `levyfit`. The bridge uses only `eta`; `mu` is used by the option 2 of the method `"MultiBridge"`, to extend the trajectories.

method	Grid of the resampling: "Bridge" (default), "ExtBridge" or "MultiBridge", see the section Resampling methods of levyBridge-package.
digits	Used only with method = "ExtBridge": number of decimal places of the inspection times, which sets the step of the grid. NULL (default) takes the number of decimal places of the times.
common_step	Used only with method = "ExtBridge": FALSE (default) gives each trajectory its own step, TRUE gives the same step to all the trajectories.
option	Used only with method = "MultiBridge": 1 (default) to resample each trajectory up to its own last inspection, 2 to resample all the trajectories up to the last inspection time over all of them, extending the shorter ones by increments of the process.

Value

An object of class "levybridge": a list with one data frame per trajectory, with the columns t (regular grid) and X (resampled levels), and the attributes teq (step of each grid), dx (increments between consecutive grid points), observed, model, parameters and method. See the section Results of levyBridge-package.

See Also

ig_bridge() for the distribution of the bridge, levyBridge-package for the grids and the algorithm, plot_bridge().

Examples

```
# 1. One trajectory observed at 11 irregular times (mu = 1, eta = 3)
t <- c(0, 1.2, 2, 3.8, 5.5, 7.1, 10.4, 12.9, 15.3, 18.7, 22.5)
set.seed(1)
x <- DegradSim::sim_ig(t, mu = 1, eta = 3)
theta <- c(mu = 1, eta = 3)
r <- ig_resampling(x, theta)
r

# The resampled trajectory, on the regular grid 0, 2.25, 4.5, ..., 22.5
r[[1]]
attr(r, "teq")
plot_bridge(r)

# 2. One trajectory given as a data frame (times, then levels): the grid is
#    0, 2, 4, 6, 8, 10, and the inspection times 2 and 6 keep their levels
x2 <- data.frame(t = c(0, 2, 3, 6, 7, 10), X = c(0, 1.4, 2.0, 4.1, 4.6, 6.8))
ig_resampling(x2, theta)[[1]]

# Equal observed levels give a constant bridge; a decrease is impossible
ig_resampling(data.frame(t = c(0, 1, 2.5, 3), X = c(0, 1, 1, 2)), theta)[[1]]
try(ig_resampling(data.frame(t = c(0, 1, 2, 3), X = c(0, 2, 1.5, 3)), theta))

# 3. Method "ExtBridge": the times have 1 decimal place, so the grid
#    0, 0.1, ..., 22.5 contains all of them
```

```
r3 <- ig_resampling(x, theta, method = "ExtBridge")
attr(r3, "teq")
all(t %in% r3[[1]]$t)
plot_bridge(r3)

# digits = 0 truncates the times to integers, with a warning
attr(ig_resampling(x, theta, method = "ExtBridge", digits = 0), "teq")

# 4. Five trajectories with the same inspection times (a matrix)
xm <- DegradSim::sim_ig(t, mu = 1, eta = 3, n_traj = 5)
head(xm)
r4 <- ig_resampling(xm, theta)
attr(r4, "teq")
plot_bridge(r4, trajectory = 1:5)

# 5. Three trajectories, each with its own inspection times (a list)
plan <- DegradSim::inspection_times("decreasing", n = 10, n_traj = 3,
                                    tmax = c(20, 30, 40), digits = 1)
xs <- DegradSim::sim_ig(plan, mu = 1, eta = 3)

# One grid and one step per trajectory
attr(ig_resampling(xs, theta), "teq")
attr(ig_resampling(xs, theta, method = "ExtBridge"), "teq")

# The same step for all the trajectories
attr(ig_resampling(xs, theta, method = "ExtBridge", common_step = TRUE), "teq")
r5 <- ig_resampling(xs, theta, method = "MultiBridge")
attr(r5, "teq")
sapply(r5, nrow)
plot_bridge(r5, trajectory = 1:3)

# Option 2: all the trajectories up to 40; beyond its last inspection, each
#    one is extended by increments of the process (this uses mu)
r6 <- ig_resampling(xs, theta, method = "MultiBridge", option = 2)
sapply(r6, nrow)
plot_bridge(r6, trajectory = 1:3)

# 6. The bridge values are random: set.seed() makes them reproducible
set.seed(2); y1 <- ig_resampling(x, theta)[[1]]$X
set.seed(2); y2 <- ig_resampling(x, theta)[[1]]$X
identical(y1, y2)

# 7. With the parameters estimated by levyfit
theta_hat <- levyfit::ig_mle(xs)
theta_hat
r7 <- ig_resampling(xs, theta_hat)
attr(r7, "parameters")

# 8. Four trajectories observed up to 8, 12, 45 and 58, each with its own
#    inspection times, resampled with "MultiBridge"
liste_t <- list(c(0, 3, 8), c(0, 2, 5, 9, 12),
                c(0, 4, 7, 11, 15, 18, 22, 27, 33, 39, 45),
                c(0, 2, 3, 5, 8, 9, 14, 15, 17, 20, 22, 25, 27, 30, 33, 46, 58))
```

```
# Inverse Gaussian process (mu = 1, eta = 3), parameters estimated by levyfit
set.seed(46)
x8 <- DegradSim::sim_ig(liste_t, mu = 1, eta = 3)
theta8 <- levyfit::ig_mle(x8)

# Option 1: one grid for all the trajectories; each one stops at the last
# grid point before its last inspection
r8 <- ig_resampling(x8, theta8, method = "MultiBridge")
attr(r8, "teq")
sapply(r8, function(d) max(d$t))
plot_bridge(r8, trajectory = 1:4)

# Option 2: the grid ends exactly at 58; beyond its last inspection, each
# shorter trajectory is extended by increments of the process
r8b <- ig_resampling(x8, theta8, method = "MultiBridge", option = 2)
attr(r8b, "teq")
sapply(r8b, function(d) max(d$t))
plot_bridge(r8b, trajectory = 1:4)
```

`meixner_bridge` *Meixner bridge*

Description

Draws the value at time t of a Meixner process known at two times $t_A < t < t_B$, from the distribution of the Meixner bridge. This function is used by `meixner_resampling()` at each point of the grid.

Usage

```
meixner_bridge(t_A, x_A, t_B, x_B, t, theta, n = 1)
```

Arguments

`t_A`, `x_A`	Time and level of the known point on the left.
`t_B`, `x_B`	Time and level of the known point on the right, $t_B > t_A$.
`t`	Time of the draw, $t_A < t < t_B$.
`theta`	Vector $\Theta = (a, b, d, m)$ of the parameters of the process, a numeric vector named `a`, `b`, `d` and `m` (scale, skewness, shape and location per unit of time), for instance the estimate returned by `meixner_mle()` of the package `levyfit`. The bridge uses only `a`, `d` and `m`.
`n`	Number of independent draws, a positive integer (default 1).

Value

A numeric vector of `n` independent draws of X_t given $X_{t_A} = x_A$ and $X_{t_B} = x_B$.

Model

The $Meixner(a, b, d, m)$ distribution, $a > 0$, $b \in]-\pi, \pi[$, $d > 0$, m real, has density

$$f(x) = \frac{(2\cos(b/2))^{2d}}{2a\pi\,\Gamma(2d)} \exp\Big(\frac{b(x-m)}{a}\Big)\Big|\Gamma\Big(d + i\frac{x-m}{a}\Big)\Big|^2$$

(Schoutens and Teugels, 1998). The Meixner process has $X_0 = 0$, independent increments and $X_t - X_s \sim Meixner(a, b, d(t-s), m(t-s))$. Its trajectories are not monotone. The vector of its parameters is $\Theta = (a, b, d, m)$.

Distribution of the bridge

Write $h_\alpha(y) = |\Gamma(\alpha + iy)|^2$ and

$$\alpha_1 = d(t - t_A), \qquad \alpha_2 = d(t_B - t), \qquad z = \frac{(x_B - x_A) - m(t_B - t_A)}{a}.$$

Given $X_{t_A} = x_A$ and $X_{t_B} = x_B$, $X_t = x_A + m(t - t_A) + aY$, where the random variable Y has the density

$$q(y) = \frac{h_{\alpha_1}(y)\, h_{\alpha_2}(z - y)}{2\pi B(2\alpha_1, 2\alpha_2)\, h_{\alpha_1+\alpha_2}(z)},$$

which is the formula of the section Bridge of a Levy process of levyBridge-package with Meixner densities. This distribution does not depend on b.

Simulation of the bridge

The random variable Y is drawn exactly by acceptance-rejection (Devroye, 1986, section II.3): a value is proposed from a simpler density, the envelope, and accepted or rejected so that the accepted values follow exactly the distribution of Y. The envelope is built in three steps.

1. **A bound of the gamma function.** For $\alpha > 0$,

$$|\Gamma(\alpha + iy)|^2 \le \Gamma(\alpha)^2 \frac{\alpha^2}{\alpha^2 + y^2}\Big(1 + \frac{y^2}{(\alpha+1)^2}\Big)^{\alpha+1/2} \exp\Big(-2|y| \arctan\frac{|y|}{\alpha+1}\Big),$$

   a bound close to $|\Gamma(\alpha + iy)|^2$ for all y, which uses only elementary functions.

2. **A piecewise exponential envelope.** The product of the bounds for $h_{\alpha_1}(y)$ and $h_{\alpha_2}(z - y)$ is bounded above by an exponential function on each of a set of intervals, which are halved until this exponential function is close to the product. Beyond these intervals, the two tails are bounded by exponential functions.

3. **Acceptance-rejection.** A piece of the envelope is chosen with a probability proportional to its area, a value is drawn from it by inversion of its distribution function, and this value is accepted with the probability given by the ratio of the density of Y to the envelope.

Guarantee. For all the parameters, a proposed value is accepted with probability at least 0.57, so that at most 1.75 values are proposed on average for each draw. The cost of the envelope changes little with z, and for n draws with the same endpoints it is built only once.

Details. The bound of step 1 follows from the product formula of the gamma function (DLMF, equation 5.8.3) and the partial fraction expansion of $\coth$ (DLMF, equation 4.36.3). The function

$\log|\Gamma(\alpha+iy)|^2$ is computed with `gsl::lngamma_complex()`, and only when the bound of step 1 and a similar lower bound cannot decide the acceptance alone. When α_1 and α_2 are both smaller than about $10^{-15}|z|$, the two peaks of the density cannot be resolved with floating point numbers: the draws stay exact, but the probability of acceptance can be lower.

References

Devroye, L. (1986). *Non-Uniform Random Variate Generation*. Springer.

NIST Digital Library of Mathematical Functions (DLMF), release 1.2.8, 2026-09-15, https://dlmf.nist.gov/: equation 5.8.3 (product formula of the gamma function) and equation 4.36.3 (partial fraction expansion of $\coth$).

Schoutens, W. and Teugels, J. L. (1998). Lévy processes, polynomials and martingales. *Communications in Statistics. Stochastic Models*, 14(1-2), 335-349.

See Also

- `meixner_resampling()`: resampling of trajectories on a regular grid;
- `gamma_bridge()`, `wiener_bridge()`, `ig_bridge()`, `vg_bridge()`: the bridges of the other processes.

Examples

```
# 1. One draw of X_2, given X_1 = 3 and X_4 = 6 (a = 1, b = 0.3, d = 1.5, m = 0.2)
set.seed(1)
meixner_bridge(t_A = 1, x_A = 3, t_B = 4, x_B = 6, t = 2,
               theta = c(a = 1, b = 0.3, d = 1.5, m = 0.2))

# 2. Many draws: the histogram of the draws and, in red, the density of
#    the bridge, f_{t - t_A}(x - x_A) f_{t_B - t}(x_B - x) / f_{t_B - t_A}(x_B - x_A)
#    (see ?levyBridge), where f_s is the density of X_s
#    (see the section Model, computed on the log scale)
f <- function(x, s, a = 1, b = 0.3, d = 1.5, m = 0.2) {
  exp(2 * d * s * log(2 * cos(b / 2)) - log(2 * a * pi) - lgamma(2 * d * s) +
        b * (x - m * s) / a +
        2 * Re(gsl::lngamma_complex(rep(d * s, length(x)), (x - m * s) / a)))
}
dens <- function(x) f(x - 3, 1) * f(6 - x, 2) / f(3, 3)
y <- meixner_bridge(1, 3, 4, 6, 2, theta = c(a = 1, b = 0.3, d = 1.5, m = 0.2), n = 50000)
hist(y, breaks = 50, freq = FALSE)
curve(dens, add = TRUE, col = "red")

# The skewness b does not change the bridge
set.seed(2)
y1 <- meixner_bridge(1, 3, 4, 6, 2, theta = c(a = 1, b = 0.3, d = 1.5, m = 0.2), n = 5)
set.seed(2)
y2 <- meixner_bridge(1, 3, 4, 6, 2, theta = c(a = 1, b = -2, d = 1.5, m = 0.2), n = 5)
identical(y1, y2)

# A bridge far from its typical value: x_B - x_A is more than 1000 times its
# standard deviation, and the draws stay fast and exact
system.time(y <- meixner_bridge(0, 0, 1, 2000, 0.5,
```

```
                                  theta = c(a = 1, b = 0.3, d = 1.5, m = 0.2), n = 1000))
summary(y)

# Errors: t outside (t_A, t_B), theta without all the parameters
try(meixner_bridge(1, 3, 4, 6, 5, theta = c(a = 1, b = 0.3, d = 1.5, m = 0.2)))
try(meixner_bridge(1, 3, 4, 6, 2, theta = c(a = 1, d = 1.5, m = 0.2)))
try(meixner_bridge(1, 3, 4, 6, 2, theta = c(a = 1, b = 4, d = 1.5, m = 0.2)))

# 3. Parameters estimated with levyfit on simulated trajectories
set.seed(1)
plan <- DegradSim::inspection_times("decreasing", n = 20, n_traj = 10, tmax = 40)
x <- DegradSim::sim_meixner(plan, a = 1, b = 0.3, d = 1.5, m = 0.2)
est <- levyfit::meixner_mle(x)
est
meixner_bridge(0, 0, 2, 1, 1, theta = est, n = 3)
```

`meixner_resampling` *Meixner process: resampling on a regular grid*

Description

Resamples one or several trajectories of a Meixner process onto a regular time grid: each grid point that is not an inspection time is drawn with `meixner_bridge()` between the known levels around it. The grids and the resampling algorithm are described in levyBridge-package.

Usage

```
meixner_resampling(
  x,
  theta,
  method = "Bridge",
  digits = NULL,
  common_step = FALSE,
  option = 1
)
```

Arguments

`x`	The observed trajectories: a data frame whose first column contains the inspection times (starting at 0, strictly increasing) and second column the degradation levels; a list of such data frames, one per trajectory; or a matrix whose first column contains the times and the other columns the levels, one column per trajectory. The results of the simulation functions of `DegradSim` are accepted. See the section Data of levyBridge-package.
`theta`	Vector $\Theta = (a, b, d, m)$ of the parameters of the process, a numeric vector named `a`, `b`, `d` and `m`, for instance the estimate returned by `meixner_mle()` of the package `levyfit`. The bridge uses only `a`, `d` and `m`; `b` is used by the option 2 of the method `"MultiBridge"`, to extend the trajectories.

`method`	Grid of the resampling: `"Bridge"` (default), `"ExtBridge"` or `"MultiBridge"`, see the section Resampling methods of levyBridge-package.
`digits`	Used only with `method = "ExtBridge"`: number of decimal places of the inspection times, which sets the step of the grid. `NULL` (default) takes the number of decimal places of the times.
`common_step`	Used only with `method = "ExtBridge"`: `FALSE` (default) gives each trajectory its own step, `TRUE` gives the same step to all the trajectories.
`option`	Used only with `method = "MultiBridge"`: `1` (default) to resample each trajectory up to its own last inspection, `2` to resample all the trajectories up to the last inspection time over all of them, extending the shorter ones by increments of the process.

Value

An object of class `"levybridge"`: a list with one data frame per trajectory, with the columns `t` (regular grid) and `X` (resampled levels), and the attributes `teq` (step of each grid), `dx` (increments between consecutive grid points), `observed`, `model`, `parameters` and `method`. See the section Results of levyBridge-package.

See Also

`meixner_bridge()` for the distribution of the bridge, levyBridge-package for the grids and the algorithm, `plot_bridge()`.

Examples

```
# 1. One trajectory observed at 11 irregular times (a = 1, b = 0.3, d = 1.5, m = 0.2)
t <- c(0, 1.2, 2, 3.8, 5.5, 7.1, 10.4, 12.9, 15.3, 18.7, 22.5)
set.seed(1)
x <- DegradSim::sim_meixner(t, a = 1, b = 0.3, d = 1.5, m = 0.2)
theta <- c(a = 1, b = 0.3, d = 1.5, m = 0.2)
r <- meixner_resampling(x, theta)
r

# The resampled trajectory, on the regular grid 0, 2.25, 4.5, ..., 22.5
r[[1]]
attr(r, "teq")
plot_bridge(r)

# 2. One trajectory given as a data frame (times, then levels): the grid is
#    0, 2, 4, 6, 8, 10, and the inspection times 2 and 6 keep their levels
x2 <- data.frame(t = c(0, 2, 3, 6, 7, 10), X = c(0, 1.4, 0.9, 3.1, 2.6, 4.8))
meixner_resampling(x2, theta)[[1]]

# 3. Method "ExtBridge": the times have 1 decimal place, so the grid
#    0, 0.1, ..., 22.5 contains all of them
r3 <- meixner_resampling(x, theta, method = "ExtBridge")
attr(r3, "teq")
all(t %in% r3[[1]]$t)
plot_bridge(r3)
```

```
# digits = 0 truncates the times to integers, with a warning
attr(meixner_resampling(x, theta, method = "ExtBridge", digits = 0), "teq")

# 4. Five trajectories with the same inspection times (a matrix)
xm <- DegradSim::sim_meixner(t, a = 1, b = 0.3, d = 1.5, m = 0.2, n_traj = 5)
head(xm)
r4 <- meixner_resampling(xm, theta)
attr(r4, "teq")
plot_bridge(r4, trajectory = 1:5)

# 5. Three trajectories, each with its own inspection times (a list)
plan <- DegradSim::inspection_times("decreasing", n = 10, n_traj = 3,
                                    tmax = c(20, 30, 40), digits = 1)
xs <- DegradSim::sim_meixner(plan, a = 1, b = 0.3, d = 1.5, m = 0.2)

# One grid and one step per trajectory
attr(meixner_resampling(xs, theta), "teq")
attr(meixner_resampling(xs, theta, method = "ExtBridge"), "teq")

# The same step for all the trajectories
attr(meixner_resampling(xs, theta, method = "ExtBridge", common_step = TRUE), "teq")
r5 <- meixner_resampling(xs, theta, method = "MultiBridge")
attr(r5, "teq")
sapply(r5, nrow)
plot_bridge(r5, trajectory = 1:3)

# Option 2: all the trajectories up to 40; beyond its last inspection, each
#    one is extended by increments of the process (this uses b)
r6 <- meixner_resampling(xs, theta, method = "MultiBridge", option = 2)
sapply(r6, nrow)
plot_bridge(r6, trajectory = 1:3)

# 6. The bridge values are random: set.seed() makes them reproducible
set.seed(2); y1 <- meixner_resampling(x, theta)[[1]]$X
set.seed(2); y2 <- meixner_resampling(x, theta)[[1]]$X
identical(y1, y2)

# 7. With the parameters estimated by levyfit
set.seed(1)
plan_e <- DegradSim::inspection_times("decreasing", n = 20, n_traj = 10, tmax = 40)
xe <- DegradSim::sim_meixner(plan_e, a = 1, b = 0.3, d = 1.5, m = 0.2)
theta_hat <- levyfit::meixner_mle(xe)
theta_hat
r7 <- meixner_resampling(xe, theta_hat)
attr(r7, "parameters")

# 8. Four trajectories observed up to 8, 12, 45 and 58, each with its own
#    inspection times, resampled with "MultiBridge"
liste_t <- list(c(0, 3, 8), c(0, 2, 5, 9, 12),
                c(0, 4, 7, 11, 15, 18, 22, 27, 33, 39, 45),
                c(0, 2, 3, 5, 8, 9, 14, 15, 17, 20, 22, 25, 27, 30, 33, 46, 58))
```

```
# Meixner process (a = 1, b = 0.3, d = 1.5, m = 0.2). With these 32
# increments, levyfit::meixner_mle() often finds no maximum inside the
# parameter space: the true parameters are used
set.seed(46)
x8 <- DegradSim::sim_meixner(liste_t, a = 1, b = 0.3, d = 1.5, m = 0.2)
theta8 <- c(a = 1, b = 0.3, d = 1.5, m = 0.2)

# Option 1: one grid for all the trajectories; each one stops at the last
# grid point before its last inspection
r8 <- meixner_resampling(x8, theta8, method = "MultiBridge")
attr(r8, "teq")
sapply(r8, function(d) max(d$t))
plot_bridge(r8, trajectory = 1:4)

# Option 2: the grid ends exactly at 58; beyond its last inspection, each
# shorter trajectory is extended by increments of the process
r8b <- meixner_resampling(x8, theta8, method = "MultiBridge", option = 2)
attr(r8b, "teq")
sapply(r8b, function(d) max(d$t))
plot_bridge(r8b, trajectory = 1:4)
```

`plot_bridge` *Plot observed and resampled trajectories*

Description

Draws, for one or several trajectories, the observed values at the inspection times and the values resampled on the regular grid by a resampling function, with `ggplot2`.

Usage

```
plot_bridge(x, trajectory = 1, title = NULL, same_scale = TRUE)
```

Arguments

`x`	An object of class `"levybridge"`, returned by a resampling function of the package, for instance `gamma_resampling()`.
`trajectory`	Indices of the trajectories to plot in `x`, for instance `1` or `1:3` (default: the first one). With several trajectories, one panel is drawn per trajectory.
`title`	Title of the plot, a character string; by default the name of the bridge, for instance `"gamma bridge"`.
`same_scale`	Used with several trajectories: `TRUE` (default) to draw all the panels with the same time and degradation axes, so that the trajectories can be compared (for instance the common grid of the method `"MultiBridge"`); `FALSE` to give each panel its own axes.

Value

A ggplot object. It is displayed when printed, and it can be modified with the usual ggplot2 functions, added with +: for instance + ggplot2::labs(x = "Time (hours)").

See Also

levyBridge-package

Examples

```
t <- c(0, 1.2, 2, 3.8, 5.5, 7.1, 10.4, 12.9, 15.3, 18.7, 22.5)
set.seed(1)
x <- DegradSim::sim_gamma(t, a = 2, b = 3)
r <- gamma_resampling(x, theta = c(a = 2, b = 3))

# Observed levels (red) and resampled levels on the regular grid (blue)
plot_bridge(r)

# Custom title and axis labels
plot_bridge(r, title = "Gamma bridge, a = 2") +
  ggplot2::labs(x = "Time (hours)", y = "Degradation (%)")

# Several trajectories: one panel per trajectory
plan <- DegradSim::inspection_times("decreasing", n = 10, n_traj = 3,
                                    tmax = c(20, 30, 40), digits = 1)
xs <- DegradSim::sim_gamma(plan, a = 2, b = 3)
plot_bridge(gamma_resampling(xs, theta = c(a = 2, b = 3)), trajectory = 1:3)

# Each panel with its own axes
plot_bridge(gamma_resampling(xs, theta = c(a = 2, b = 3)), trajectory = 1:3,
            same_scale = FALSE)
```

---

vg_bridge *Variance-gamma bridge*

Description

Draws the value at time t of a variance-gamma process known at two times $t_A < t < t_B$, from the distribution of the variance-gamma bridge. This function is used by vg_resampling() at each point of the grid.

Usage

```
vg_bridge(t_A, x_A, t_B, x_B, t, theta, n = 1)
```

Arguments

`t_A`, `x_A`	Time and level of the known point on the left.
`t_B`, `x_B`	Time and level of the known point on the right, $t_B > t_A$.
`t`	Time of the draw, $t_A < t < t_B$.
`theta`	Vector $\Theta = (\theta, \sigma, \nu)$ of the parameters of the process, a numeric vector named `theta`, `sigma` and `nu` (drift, volatility of the subordinated Brownian motion and variance rate of the gamma process), for instance the estimate returned by `vg_mle()` of the package `levyfit`.
`n`	Number of independent draws, a positive integer (default 1).

Value

A numeric vector of `n` independent draws of X_t given $X_{t_A} = x_A$ and $X_{t_B} = x_B$.

Model

The variance-gamma process with parameters θ, $\sigma > 0$ and $\nu > 0$ is $X_s = \theta G_s + \sigma W_{G_s}$ (Madan, Carr and Chang, 1998), where G is a gamma process with $G_s \sim \Gamma(s/\nu)$ with scale ν (mean s, variance νs) and W an independent standard Brownian motion. Its trajectories are not monotone. The vector of its parameters is $\Theta = (\theta, \sigma, \nu)$: the drift θ is one of the parameters, not the whole vector Θ.

Distribution of the bridge

Given $X_{t_A} = x_A$ and $X_{t_B} = x_B$,

$$X_t = x_A + (x_B - x_A)\, Y + \sqrt{S\, Y (1 - Y)}\, \xi,$$

where the random variables S, Y and ξ are independent:

- the variable S follows the distribution $\mathrm{GIG}(p, \kappa^2, (x_B - x_A)^2)$ in the notation of Peña and Jauch (2025), with density proportional to

$$s^{p-1} \exp\big(-(\kappa^2 s + (x_B - x_A)^2/s)/2\big), \qquad s > 0,$$

  where $p = (t_B - t_A)/\nu - 1/2$ and $\kappa^2 = (\theta^2 + 2\sigma^2/\nu)/\sigma^4$;
- $Y \sim \mathrm{Beta}((t - t_A)/\nu, (t_B - t)/\nu)$;
- $\xi \sim \mathcal{N}(0, 1)$.

This representation is given by Hoyle (2010, section 2.7.2): equations (2.83) to (2.87) for the standard variance-gamma process, whose parameter m is $1/\nu$ here, and equation (2.77), which writes the bridge of any variance-gamma process as a scaled bridge of the standard one.

When $x_B = x_A$, the last parameter of the GIG distribution is 0: the variable S then has the gamma distribution with shape p and rate $\kappa^2/2$ if $p > 0$; if $p \le 0$, this case has probability zero and the package takes $S = 0$.

The variable S/σ^2 is the increment $G_{t_B} - G_{t_A}$ of the gamma process given the observations, and Y is the fraction of this increment reached at time t, as in the gamma bridge; given G, the Brownian motion W gives the Gaussian term. The bridge uses the three parameters. The GIG variable is drawn with `GIGrvg::rgig()` (Hörmann and Leydold, 2014).

References

Hörmann, W. and Leydold, J. (2014). Generating generalized inverse Gaussian random variates. *Statistics and Computing*, 24, 547-557. doi:10.1007/s1122201393873

Hoyle, A. E. V. (2010). *Information-Based Models for Finance and Insurance*. PhD thesis, Imperial College London. arXiv:1010.0829.

Peña, V. and Jauch, M. (2025). Properties of the generalized inverse Gaussian with applications to Monte Carlo simulation and distribution function evaluation. arXiv:2401.00749.

Madan, D. B., Carr, P. P. and Chang, E. C. (1998). The variance gamma process and option pricing. *European Finance Review*, 2(1), 79-105. doi:10.1023/A:1009703431535

See Also

- vg_resampling(): resampling of trajectories on a regular grid;
- gamma_bridge(), wiener_bridge(), ig_bridge(), meixner_bridge(): the bridges of the other processes.

Examples

```
# 1. One draw of X_2, given X_1 = 3 and X_4 = 6 (theta = 0.5, sigma = 1.5, nu = 2)
set.seed(1)
vg_bridge(t_A = 1, x_A = 3, t_B = 4, x_B = 6, t = 2,
          theta = c(theta = 0.5, sigma = 1.5, nu = 2))

# 2. Many draws of X_3 given X_1 = 3 and X_6 = 6: the histogram of the
#    draws and, in red, the density of the bridge,
#    f_{t - t_A}(x - x_A) f_{t_B - t}(x_B - x) / f_{t_B - t_A}(x_B - x_A)
#    (see ?levyBridge), where f_s is the density of X_s
#    (Madan, Carr and Chang, 1998, computed on the log scale). With nu = 2,
#    the times are such that t - t_A > nu / 2 and t_B - t > nu / 2: otherwise
#    the density of the bridge is infinite at x_A or x_B
f <- function(x, s, th = 0.5, sg = 1.5, nu = 2) {
  k <- sqrt(th^2 + 2 * sg^2 / nu)
  z <- k * abs(x) / sg^2
  exp(log(2) + th * x / sg^2 - (s / nu) * log(nu) - log(sqrt(2 * pi) * sg) -
        lgamma(s / nu) + (s / nu - 0.5) * log(abs(x) / k) +
        log(besselK(z, s / nu - 0.5, expon.scaled = TRUE)) - z)
}
dens <- function(x) f(x - 3, 2) * f(6 - x, 3) / f(3, 5)
y <- vg_bridge(1, 3, 6, 6, 3, theta = c(theta = 0.5, sigma = 1.5, nu = 2), n = 10000)
hist(y, breaks = 50, freq = FALSE)
curve(dens, add = TRUE, col = "red", n = 500)

# Errors: t outside (t_A, t_B), theta without all the parameters
try(vg_bridge(1, 3, 4, 6, 5, theta = c(theta = 0.5, sigma = 1.5, nu = 2)))
try(vg_bridge(1, 3, 4, 6, 2, theta = c(sigma = 1.5, nu = 2)))

# 3. Parameters estimated with levyfit on simulated trajectories
set.seed(1)
plan <- DegradSim::inspection_times("decreasing", n = 20, n_traj = 10, tmax = 40)
x <- DegradSim::sim_vg(plan, theta = 0.5, sigma = 1.5, nu = 2)
```

```
est <- levyfit::vg_mle(x)
est
vg_bridge(0, 0, 2, 1, 1, theta = est, n = 3)
```

---

vg_resampling *Variance-gamma process: resampling on a regular grid*

Description

Resamples one or several trajectories of a variance-gamma process onto a regular time grid: each grid point that is not an inspection time is drawn with `vg_bridge()` between the known levels around it. The grids and the resampling algorithm are described in levyBridge-package.

Usage

```
vg_resampling(
  x,
  theta,
  method = "Bridge",
  digits = NULL,
  common_step = FALSE,
  option = 1
)
```

Arguments

`x`	The observed trajectories: a data frame whose first column contains the inspection times (starting at 0, strictly increasing) and second column the degradation levels; a list of such data frames, one per trajectory; or a matrix whose first column contains the times and the other columns the levels, one column per trajectory. The results of the simulation functions of `DegradSim` are accepted. See the section Data of levyBridge-package.
`theta`	Vector $\Theta = (\theta, \sigma, \nu)$ of the parameters of the process, a numeric vector named `theta`, `sigma` and `nu`, for instance the estimate returned by `vg_mle()` of the package `levyfit`. The bridge and the option 2 of the method `"MultiBridge"` use the three parameters.
`method`	Grid of the resampling: `"Bridge"` (default), `"ExtBridge"` or `"MultiBridge"`, see the section Resampling methods of levyBridge-package.
`digits`	Used only with `method = "ExtBridge"`: number of decimal places of the inspection times, which sets the step of the grid. `NULL` (default) takes the number of decimal places of the times.
`common_step`	Used only with `method = "ExtBridge"`: `FALSE` (default) gives each trajectory its own step, `TRUE` gives the same step to all the trajectories.
`option`	Used only with `method = "MultiBridge"`: `1` (default) to resample each trajectory up to its own last inspection, `2` to resample all the trajectories up to the last inspection time over all of them, extending the shorter ones by increments of the process.

Value

An object of class "levybridge": a list with one data frame per trajectory, with the columns t (regular grid) and X (resampled levels), and the attributes teq (step of each grid), dx (increments between consecutive grid points), observed, model, parameters and method. See the section Results of levyBridge-package.

See Also

vg_bridge() for the distribution of the bridge, levyBridge-package for the grids and the algorithm, plot_bridge().

Examples

```
# 1. One trajectory observed at 11 irregular times (theta = 0.5, sigma = 1.5, nu = 2)
t <- c(0, 1.2, 2, 3.8, 5.5, 7.1, 10.4, 12.9, 15.3, 18.7, 22.5)
set.seed(1)
x <- DegradSim::sim_vg(t, theta = 0.5, sigma = 1.5, nu = 2)
theta <- c(theta = 0.5, sigma = 1.5, nu = 2)
r <- vg_resampling(x, theta)
r

# The resampled trajectory, on the regular grid 0, 2.25, 4.5, ..., 22.5
r[[1]]
attr(r, "teq")
plot_bridge(r)

# 2. One trajectory given as a data frame (times, then levels): the grid is
#    0, 2, 4, 6, 8, 10, and the inspection times 2 and 6 keep their levels
x2 <- data.frame(t = c(0, 2, 3, 6, 7, 10), X = c(0, 1.4, 0.9, 3.1, 2.6, 4.8))
vg_resampling(x2, theta)[[1]]

# 3. Method "ExtBridge": the times have 1 decimal place, so the grid
#    0, 0.1, ..., 22.5 contains all of them
r3 <- vg_resampling(x, theta, method = "ExtBridge")
attr(r3, "teq")
all(t %in% r3[[1]]$t)
plot_bridge(r3)

# digits = 0 truncates the times to integers, with a warning
attr(vg_resampling(x, theta, method = "ExtBridge", digits = 0), "teq")

# 4. Five trajectories with the same inspection times (a matrix)
xm <- DegradSim::sim_vg(t, theta = 0.5, sigma = 1.5, nu = 2, n_traj = 5)
head(xm)
r4 <- vg_resampling(xm, theta)
attr(r4, "teq")
plot_bridge(r4, trajectory = 1:5)

# 5. Three trajectories, each with its own inspection times (a list)
plan <- DegradSim::inspection_times("decreasing", n = 10, n_traj = 3,
                                    tmax = c(20, 30, 40), digits = 1)
xs <- DegradSim::sim_vg(plan, theta = 0.5, sigma = 1.5, nu = 2)
```

```
# One grid and one step per trajectory
attr(vg_resampling(xs, theta), "teq")
attr(vg_resampling(xs, theta, method = "ExtBridge"), "teq")

# The same step for all the trajectories
attr(vg_resampling(xs, theta, method = "ExtBridge", common_step = TRUE), "teq")
r5 <- vg_resampling(xs, theta, method = "MultiBridge")
attr(r5, "teq")
sapply(r5, nrow)
plot_bridge(r5, trajectory = 1:3)

# Option 2: all the trajectories up to 40; beyond its last inspection, each
#    one is extended by increments of the process
r6 <- vg_resampling(xs, theta, method = "MultiBridge", option = 2)
sapply(r6, nrow)
plot_bridge(r6, trajectory = 1:3)

# 6. The bridge values are random: set.seed() makes them reproducible
set.seed(2); y1 <- vg_resampling(x, theta)[[1]]$X
set.seed(2); y2 <- vg_resampling(x, theta)[[1]]$X
identical(y1, y2)

# 7. With the parameters estimated by levyfit
set.seed(1)
plan_e <- DegradSim::inspection_times("decreasing", n = 20, n_traj = 10, tmax = 40)
xe <- DegradSim::sim_vg(plan_e, theta = 0.5, sigma = 1.5, nu = 2)
theta_hat <- levyfit::vg_mle(xe)
theta_hat
r7 <- vg_resampling(xe, theta_hat)
attr(r7, "parameters")

# 8. Four trajectories observed up to 8, 12, 45 and 58, each with its own
#    inspection times, resampled with "MultiBridge"
liste_t <- list(c(0, 3, 8), c(0, 2, 5, 9, 12),
                c(0, 4, 7, 11, 15, 18, 22, 27, 33, 39, 45),
                c(0, 2, 3, 5, 8, 9, 14, 15, 17, 20, 22, 25, 27, 30, 33, 46, 58))

# Variance-gamma process (theta = 0.5, sigma = 1.5, nu = 2), parameters
# estimated by levyfit
set.seed(46)
x8 <- DegradSim::sim_vg(liste_t, theta = 0.5, sigma = 1.5, nu = 2)
theta8 <- levyfit::vg_mle(x8)

# Option 1: one grid for all the trajectories; each one stops at the last
# grid point before its last inspection
r8 <- vg_resampling(x8, theta8, method = "MultiBridge")
attr(r8, "teq")
sapply(r8, function(d) max(d$t))
plot_bridge(r8, trajectory = 1:4)

# Option 2: the grid ends exactly at 58; beyond its last inspection, each
# shorter trajectory is extended by increments of the process
```

```
r8b <- vg_resampling(x8, theta8, method = "MultiBridge", option = 2)
attr(r8b, "teq")
sapply(r8b, function(d) max(d$t))
plot_bridge(r8b, trajectory = 1:4)
```

`wiener_bridge` *Brownian bridge*

Description

Draws the value at time t of a Wiener process with drift known at two times $t_A < t < t_B$, from the distribution of the Brownian bridge. This function is used by `wiener_resampling()` at each point of the grid.

Usage

```
wiener_bridge(t_A, x_A, t_B, x_B, t, theta, n = 1)
```

Arguments

`t_A`, `x_A`	Time and level of the known point on the left.
`t_B`, `x_B`	Time and level of the known point on the right, $t_B > t_A$.
`t`	Time of the draw, $t_A < t < t_B$.
`theta`	Vector $\Theta = (\mu, \sigma)$ of the parameters of the process, a numeric vector named `mu` and `sigma`, with the volatility σ (not the variance). The function `wiener_mle()` of the package `levyfit` returns `mu` and `sigma2`: use `c(mu = est[["mu"]], sigma = sqrt(est[["sigma2"]]))`. The bridge uses only `sigma`.
`n`	Number of independent draws, a positive integer (default 1).

Value

A numeric vector of `n` independent draws of X_t given $X_{t_A} = x_A$ and $X_{t_B} = x_B$.

Model

The Wiener process with drift μ and volatility $\sigma > 0$ is $X_t = \mu t + \sigma B_t$, with B a standard Brownian motion: $X_0 = 0$, independent increments and $X_t - X_s \sim \mathcal{N}(\mu(t-s), \sigma^2(t-s))$. Its trajectories are not monotone. The vector of its parameters is $\Theta = (\mu, \sigma)$.

Distribution of the bridge

Given $X_{t_A} = x_A$ and $X_{t_B} = x_B$,

$$X_t \sim \mathcal{N}\Big(x_A + (x_B - x_A)\frac{t - t_A}{t_B - t_A},\ \sigma^2 \frac{(t - t_A)(t_B - t)}{t_B - t_A}\Big),$$

which is the formula of the section Bridge of a Levy process of levyBridge-package with Gaussian densities. This distribution does not depend on the drift μ.

See Also

- wiener_resampling(): resampling of trajectories on a regular grid;
- gamma_bridge(), ig_bridge(), vg_bridge(), meixner_bridge(): the bridges of the other processes.

Examples

```
# 1. One draw of X_2, given X_1 = 3 and X_4 = 6 (mu = 0.5, sigma = 1.5)
set.seed(1)
wiener_bridge(t_A = 1, x_A = 3, t_B = 4, x_B = 6, t = 2, theta = c(mu = 0.5, sigma = 1.5))

# 2. Many draws: the histogram of the draws and, in red, the density of
#    the bridge, f_{t - t_A}(x - x_A) f_{t_B - t}(x_B - x) / f_{t_B - t_A}(x_B - x_A)
#    (see ?levyBridge), where f_s is the density of X_s
f <- function(x, s) dnorm(x, mean = 0.5 * s, sd = 1.5 * sqrt(s))
dens <- function(x) f(x - 3, 1) * f(6 - x, 2) / f(3, 3)
y <- wiener_bridge(1, 3, 4, 6, 2, theta = c(mu = 0.5, sigma = 1.5), n = 10000)
hist(y, breaks = 50, freq = FALSE)
curve(dens, add = TRUE, col = "red")

# 3. A decrease is possible: the process is not monotone
wiener_bridge(0, 2, 2, 1, 0.5, theta = c(mu = 0.5, sigma = 1.5), n = 3)

# Errors: t outside (t_A, t_B), theta without all the parameters
try(wiener_bridge(1, 3, 4, 6, 5, theta = c(mu = 0.5, sigma = 1.5)))
try(wiener_bridge(1, 3, 4, 6, 2, theta = c(sigma = 1.5)))

# 4. Parameters estimated with levyfit (variance sigma2 turned into sigma)
t <- c(0, 1.2, 2, 3.8, 5.5, 7.1, 10.4, 12.9, 15.3, 18.7, 22.5)
x <- DegradSim::sim_wiener(t, mu = 0.5, sigma = 1.5)
est <- levyfit::wiener_mle(x)
theta <- c(mu = est[["mu"]], sigma = sqrt(est[["sigma2"]]))
wiener_bridge(1.2, x[2, 2], 2, x[3, 2], 1.5, theta = theta, n = 3)
```

---

wiener_resampling *Wiener process: resampling on a regular grid*

Description

Resamples one or several trajectories of a Wiener process with drift onto a regular time grid: each grid point that is not an inspection time is drawn with wiener_bridge() between the known levels around it. The grids and the resampling algorithm are described in levyBridge-package.

Usage

```
wiener_resampling(
  x,
```

```
  theta,
  method = "Bridge",
  digits = NULL,
  common_step = FALSE,
  option = 1
)
```

Arguments

x	The observed trajectories: a data frame whose first column contains the inspection times (starting at 0, strictly increasing) and second column the degradation levels; a list of such data frames, one per trajectory; or a matrix whose first column contains the times and the other columns the levels, one column per trajectory. The results of the simulation functions of `DegradSim` are accepted. See the section Data of levyBridge-package.
theta	Vector $\Theta = (\mu, \sigma)$ of the parameters of the process, a numeric vector named `mu` and `sigma`, with the volatility σ (not the variance, see `wiener_bridge()`). The bridge uses only `sigma`; `mu` is used by the option 2 of the method `"MultiBridge"`, to extend the trajectories.
method	Grid of the resampling: `"Bridge"` (default), `"ExtBridge"` or `"MultiBridge"`, see the section Resampling methods of levyBridge-package.
digits	Used only with `method = "ExtBridge"`: number of decimal places of the inspection times, which sets the step of the grid. `NULL` (default) takes the number of decimal places of the times.
common_step	Used only with `method = "ExtBridge"`: `FALSE` (default) gives each trajectory its own step, `TRUE` gives the same step to all the trajectories.
option	Used only with `method = "MultiBridge"`: `1` (default) to resample each trajectory up to its own last inspection, `2` to resample all the trajectories up to the last inspection time over all of them, extending the shorter ones by increments of the process.

Value

An object of class `"levybridge"`: a list with one data frame per trajectory, with the columns `t` (regular grid) and `X` (resampled levels), and the attributes `teq` (step of each grid), `dx` (increments between consecutive grid points), `observed`, `model`, `parameters` and `method`. See the section Results of levyBridge-package.

See Also

`wiener_bridge()` for the distribution of the bridge, levyBridge-package for the grids and the algorithm, `plot_bridge()`.

Examples

```
# 1. One trajectory observed at 11 irregular times (mu = 0.5, sigma = 1.5)
t <- c(0, 1.2, 2, 3.8, 5.5, 7.1, 10.4, 12.9, 15.3, 18.7, 22.5)
set.seed(1)
```

```
x <- DegradSim::sim_wiener(t, mu = 0.5, sigma = 1.5)
theta <- c(mu = 0.5, sigma = 1.5)
r <- wiener_resampling(x, theta)
r

# The resampled trajectory, on the regular grid 0, 2.25, 4.5, ..., 22.5
r[[1]]
attr(r, "teq")
plot_bridge(r)

# 2. One trajectory given as a data frame (times, then levels): the grid is
#    0, 2, 4, 6, 8, 10, and the inspection times 2 and 6 keep their levels
x2 <- data.frame(t = c(0, 2, 3, 6, 7, 10), X = c(0, 1.4, 0.9, 3.1, 2.6, 4.8))
wiener_resampling(x2, theta)[[1]]

# 3. Method "ExtBridge": the times have 1 decimal place, so the grid
#    0, 0.1, ..., 22.5 contains all of them
r3 <- wiener_resampling(x, theta, method = "ExtBridge")
attr(r3, "teq")
all(t %in% r3[[1]]$t)
plot_bridge(r3)

# digits = 0 truncates the times to integers, with a warning
attr(wiener_resampling(x, theta, method = "ExtBridge", digits = 0), "teq")

# 4. Five trajectories with the same inspection times (a matrix)
xm <- DegradSim::sim_wiener(t, mu = 0.5, sigma = 1.5, n_traj = 5)
head(xm)
r4 <- wiener_resampling(xm, theta)
attr(r4, "teq")
plot_bridge(r4, trajectory = 1:5)

# 5. Three trajectories, each with its own inspection times (a list)
plan <- DegradSim::inspection_times("decreasing", n = 10, n_traj = 3,
                                    tmax = c(20, 30, 40), digits = 1)
xs <- DegradSim::sim_wiener(plan, mu = 0.5, sigma = 1.5)

# One grid and one step per trajectory
attr(wiener_resampling(xs, theta), "teq")
attr(wiener_resampling(xs, theta, method = "ExtBridge"), "teq")

# The same step for all the trajectories
attr(wiener_resampling(xs, theta, method = "ExtBridge", common_step = TRUE), "teq")
r5 <- wiener_resampling(xs, theta, method = "MultiBridge")
attr(r5, "teq")
sapply(r5, nrow)
plot_bridge(r5, trajectory = 1:3)

# Option 2: all the trajectories up to 40; beyond its last inspection, each
#    one is extended by increments of the process (this uses mu)
r6 <- wiener_resampling(xs, theta, method = "MultiBridge", option = 2)
sapply(r6, nrow)
plot_bridge(r6, trajectory = 1:3)
```

```
# 6. The bridge values are random: set.seed() makes them reproducible
set.seed(2); y1 <- wiener_resampling(x, theta)[[1]]$X
set.seed(2); y2 <- wiener_resampling(x, theta)[[1]]$X
identical(y1, y2)

# 7. With the parameters estimated by levyfit
est <- levyfit::wiener_mle(xs)
theta_hat <- c(mu = est[["mu"]], sigma = sqrt(est[["sigma2"]]))
theta_hat
r7 <- wiener_resampling(xs, theta_hat)
attr(r7, "parameters")

# 8. Four trajectories observed up to 8, 12, 45 and 58, each with its own
#    inspection times, resampled with "MultiBridge"
liste_t <- list(c(0, 3, 8), c(0, 2, 5, 9, 12),
                c(0, 4, 7, 11, 15, 18, 22, 27, 33, 39, 45),
                c(0, 2, 3, 5, 8, 9, 14, 15, 17, 20, 22, 25, 27, 30, 33, 46, 58))

# Wiener process (mu = 0.5, sigma = 1.5), parameters estimated by levyfit
set.seed(46)
x8 <- DegradSim::sim_wiener(liste_t, mu = 0.5, sigma = 1.5)
est8 <- levyfit::wiener_mle(x8)
theta8 <- c(mu = est8[["mu"]], sigma = sqrt(est8[["sigma2"]]))

# Option 1: one grid for all the trajectories; each one stops at the last
# grid point before its last inspection
r8 <- wiener_resampling(x8, theta8, method = "MultiBridge")
attr(r8, "teq")
sapply(r8, function(d) max(d$t))
plot_bridge(r8, trajectory = 1:4)

# Option 2: the grid ends exactly at 58; beyond its last inspection, each
# shorter trajectory is extended by increments of the process
r8b <- wiener_resampling(x8, theta8, method = "MultiBridge", option = 2)
attr(r8b, "teq")
sapply(r8b, function(d) max(d$t))
plot_bridge(r8b, trajectory = 1:4)
```

Index

∗ **package**
levyBridge-package, 2

gamma_bridge, 7
gamma_bridge(), *3*, *9, 10*, *13*, *19*, *26*, *31*
gamma_resampling, 9
gamma_resampling(), *3*, *7, 8*, *23*
GIGrvg::rgig(), *25*
gsl::lngamma_complex(), *19*

ig_bridge, 12
ig_bridge(), *3*, *8*, *14, 15*, *19*, *26*, *31*
ig_resampling, 14
ig_resampling(), *3*, *12, 13*

levyBridge *(*levyBridge-package*)*, 2
levyBridge-package, 2, *9, 10*, *14, 15*, *18*, *20, 21*, *24*, *27, 28*, *30–32*

meixner_bridge, 17
meixner_bridge(), *3*, *8*, *13*, *20, 21*, *26*, *31*
meixner_resampling, 20
meixner_resampling(), *3*, *17*, *19*

plot_bridge, 23
plot_bridge(), *5*, *10*, *15*, *21*, *28*, *32*

vg_bridge, 24
vg_bridge(), *3*, *8*, *13*, *19*, *27, 28*, *31*
vg_resampling, 27
vg_resampling(), *3*, *24*, *26*

wiener_bridge, 30
wiener_bridge(), *3*, *8*, *13*, *19*, *26*, *31, 32*
wiener_resampling, 31
wiener_resampling(), *3*, *30, 31*