

Package ‘hdtg’

January 26, 2026

Title Generate Samples from Multivariate Truncated Normal Distributions

Version 0.3.3

Maintainer Zhenyu Zhang <zhangzhenyusa@gmail.com>

Description Efficient sampling from high-dimensional truncated Gaussian distributions, or multivariate truncated normal (MTN). Techniques include zigzag Hamiltonian Monte Carlo as in Akihiko Nishimura, Zhenyu Zhang and Marc A. Suchard (2024) <[doi:10.1080/01621459.2024.2395587](https://doi.org/10.1080/01621459.2024.2395587)>, and harmonic Monte Carlo in Ari Pakman and Liam Paninski (2014) <[doi:10.1080/10618600.2013.788448](https://doi.org/10.1080/10618600.2013.788448)>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports Rcpp, RcppParallel, mgcv, Rdpack

RdMacros Rdpack

LinkingTo Rcpp, RcppEigen, RcppParallel

Suggests TruncatedNormal, testthat (>= 3.0.0)

Config/testthat/edition 3

SystemRequirements CPU with AVX/SSE4.2 (optional for better performance)

Acknowledgements The package uses the following R infrastructure: Rcpp (Eddelbuettel & Francois, 2011), RcppEigen (Bates & Eddelbuettel, 2013), RcppParallel (Allaire et al., 2025), mgcv (Wood, 2017), and Rdpack (Boshnakov, 2023).

NeedsCompilation yes

Author Zhenyu Zhang [aut, cre],
Andrew Chin [aut],
Akihiko Nishimura [aut],
Marc A. Suchard [aut],
John W. Ratcliff et al. [cph, ctb] (authors and copyright holders of see2neon.h under an MIT license)

Repository CRAN
Date/Publication 2026-01-26 04:20:02 UTC

Contents

cholesky	2
createEngine	3
createNutsEngine	4
drawLaplaceMomentum	5
getHarmonicSample	6
getInitialPosition	7
getMarkovianZigzagSample	8
getZigzagSample	9
harmonicHMC	10
markovianZigzag	12
setMean	14
setPrecision	14
zigzagHMC	15
Index	18

cholesky	<i>Efficient Cholesky decomposition</i>
----------	---

Description

Compute Cholesky decomposition of a matrix.

Usage

cholesky(A)

Arguments

A matrix to decompose

Value

upper triangular matrix R such that $A = U'U$.

See Also

[harmonicHMC\(\)](#)

Examples

```
# Larger example
set.seed(123)
B <- matrix(rnorm(16), 4, 4)
B <- t(B) %*% B # Make symmetric positive definite
U <- cholesky(B)
U
# Verify decomposition
all.equal(B, t(U) %*% U)
```

createEngine	<i>Create a Zigzag-HMC engine object</i>
--------------	--

Description

Create the C++ object to set up SIMD vectorization for speeding up calculations for Zigzag-HMC ("Zigzag-HMC engine").

Usage

```
createEngine(
  dimension,
  lowerBounds,
  upperBounds,
  seed,
  mean,
  precision,
  flags = 128L,
  numThreads = 1L
)
```

Arguments

dimension	the dimension of MTN.
lowerBounds	a vector specifying the lower bounds.
upperBounds	a vector specifying the upper bounds.
seed	random seed.
mean	the mean vector.
precision	the precision matrix.
flags	which SIMD instruction set to use. 128 = SSE, 256 = AVX.
numThreads	number of threads for parallel execution (default = 1). Set to 0 for automatic detection of available cores.

Value

a list whose only element is the Zigzag-HMC engine object.

See Also

[setMean\(\)](#), [setPrecision\(\)](#), [zigzagHMC\(\)](#), [markovianZigzag\(\)](#)

Examples

```
# Create a 2D engine with simple bounds
dimension <- 2
lowerBounds <- c(-1, -1)
upperBounds <- c(1, 1)
mean <- c(0, 0)
precision <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
engine <- createEngine(dimension, lowerBounds, upperBounds,
                      seed = 123, mean, precision, flags = 128)
# Check the engine structure
str(engine)
```

createNutsEngine

Create a Zigzag-NUTS engine object

Description

Create the C++ object to set up SIMD vectorization for speeding up calculations for Zigzag-NUTS ("Zigzag-NUTS engine").

Usage

```
createNutsEngine(
  dimension,
  lowerBounds,
  upperBounds,
  seed,
  stepSize,
  mean,
  precision,
  flags = 128L,
  numThreads = 1L
)
```

Arguments

dimension	the dimension of MTN.
lowerBounds	a vector specifying the lower bounds.
upperBounds	a vector specifying the upper bounds.
seed	random seed.
stepSize	the base step size for Zigzag-NUTS.
mean	the mean vector.

precision	the precision matrix.
flags	which SIMD instruction set to use. 128 = SSE, 256 = AVX.
numThreads	number of threads for parallel execution (default = 1). Set to 0 for automatic detection of available cores.

Value

a list whose only element is the Zigzag-NUTS engine object.

See Also

[setMean\(\)](#), [setPrecision\(\)](#), [zigzagHMC\(\)](#), [createEngine\(\)](#)

Examples

```
# Create a Zigzag-NUTS engine for a 2D problem
dimension <- 2
lowerBounds <- c(-2, -2)
upperBounds <- c(2, 2)
stepSize <- 0.1
mean <- c(0.5, -0.5)
precision <- matrix(c(2, 0.3, 0.3, 2), nrow = 2)
nuts_engine <- createNutsEngine(dimension, lowerBounds, upperBounds,
                                seed = 456, stepSize, mean, precision)

str(nuts_engine)
```

drawLaplaceMomentum	<i>Draw a random Laplace momentum</i>
---------------------	---------------------------------------

Description

Generate a d-dimensional momentum where the density of each element is proportional to $\exp(-|p|)$.

Usage

```
drawLaplaceMomentum(d)
```

Arguments

d	dimension of the momentum.
---	----------------------------

Value

a d-dimensional Laplace-distributed momentum.

See Also

[zigzagHMC\(\)](#)

Examples

```
# Draw a 3-dimensional Laplace momentum with reproducible results
set.seed(3)
momentum <- drawLaplaceMomentum(3)
momentum
```

getHarmonicSample	<i>One-step Harmonic HMC Sampler (Whitened Coordinates)</i>
-------------------	---

Description

One-step Harmonic HMC Sampler (Whitened Coordinates)

Usage

```
getHarmonicSample(
  whitenedPosition,
  whitenedConstraints,
  integrationTime,
  diagnosticMode = FALSE,
  seed = NULL
)
```

Arguments

whitenedPosition	Position in whitened coordinates
whitenedConstraints	List from applyWhitenTransform()
integrationTime	Time for dynamics simulation
diagnosticMode	Return bounce diagnostics
seed	random seed

See Also

[harmonicHMC\(\)](#)

Examples

```
# Basic usage with whitened coordinates
set.seed(123)
whitened_pos <- c(0.1, -0.2, 0.3)
# Create example whitened constraints
whitened_constraints <- list(
  direc = matrix(c(1, 0, 0, 0, 1, 0), nrow = 2, byrow = TRUE),
  direcRowNormSq = c(1, 1),
```

```

    bound = c(-0.5, -0.5)
  )
  result <- getHarmonicSample(
    whitenedPosition = whitened_pos,
    whitenedConstraints = whitened_constraints,
    integrationTime = pi/4
  )
  result

# With diagnostics enabled
result_diag <- getHarmonicSample(
  whitenedPosition = whitened_pos,
  whitenedConstraints = whitened_constraints,
  integrationTime = pi/4,
  diagnosticMode = TRUE
)
str(result_diag)

```

getInitialPosition	<i>Get an eligible initial value for a MTN with given mean and truncations</i>
--------------------	--

Description

For a given MTN the function returns an initial vector whose elements are one of: (1) middle point of the truncation interval if both lower and upper bounds are finite (2) lower (upper) bound +0.1 (-0.1) if only the lower (upper) bound is finite (3) the corresponding mean value if lower bound = -Inf are upper bound = Inf.

Usage

```
getInitialPosition(mean, lowerBounds, upperBounds)
```

Arguments

mean	a d-dimensional mean vector.
lowerBounds	a d-dimensional vector specifying the lower bounds.
upperBounds	a d-dimensional vector specifying the upper bounds.

Value

an eligible d-dimensional initial vector.

See Also

[harmonicHMC\(\)](#), [zigzagHMC\(\)](#), [markovianZigzag\(\)](#)

Examples

```
# Example 1: Bounded interval
mean <- c(0, 0)
lower <- c(-1, -2)
upper <- c(1, 2)
getInitialPosition(mean, lower, upper)

# Example 2: Mixed bounds (some finite, some infinite)
mean <- c(0, 0, 0)
lower <- c(-Inf, 0, -1)
upper <- c(Inf, 5, Inf)
getInitialPosition(mean, lower, upper)

# Example 3: All unbounded (returns mean)
mean <- c(1, 2, 3)
lower <- c(-Inf, -Inf, -Inf)
upper <- c(Inf, Inf, Inf)
getInitialPosition(mean, lower, upper)
```

```
getMarkovianZigzagSample
```

Draw one Markovian zigzag sample

Description

Simulate the Markovian zigzag dynamics for a given position over a specified travel time.

Usage

```
getMarkovianZigzagSample(position, velocity = NULL, engine, travelTime)
```

Arguments

position	a d-dimensional position vector.
velocity	optional d-dimensional velocity vector. If NULL, it will be generated within the function.
engine	an object representing the Markovian zigzag engine, typically containing settings and state required for the simulation.
travelTime	the duration for which the dynamics are simulated.

Value

A list containing the position and velocity after simulating the dynamics.

See Also

[markovianZigzag\(\)](#)

Examples

```
# First create an engine
set.seed(123)
engine <- createEngine(
  dimension = 2,
  lowerBounds = c(-1, -1),
  upperBounds = c(1, 1),
  seed = 123,
  mean = c(0, 0),
  precision = diag(2)
)

# Draw a single Markovian zigzag sample
position <- c(0.1, -0.2)
travel_time <- 0.5
sample_result <- getMarkovianZigzagSample(
  position = position,
  engine = engine,
  travelTime = travel_time
)
sample_result
```

getZigzagSample

*Draw one MTN sample with Zigzag-HMC or Zigzag-NUTS***Description**

Simulate the Zigzag-HMC or Zigzag-NUTS dynamics on a given MTN.

Usage

```
getZigzagSample(position, momentum = NULL, nutsFlg, engine, stepSize = NULL)
```

Arguments

position	a d-dimensional initial position vector.
momentum	a d-dimensional initial momentum vector.
nutsFlg	logical. If TRUE the No-U-Turn sampler will be used (Zigzag-NUTS).
engine	list. Its engine element is a pointer to the Zigzag-HMC engine (or Zigzag-NUTS engine) C++ object that implements fast computations for Zigzag-HMC (or Zigzag-NUTS).
stepSize	step size for Zigzag-HMC. If nutsFlg = TRUE, engine contains the base step size for Zigzag-NUTS).

Value

one MCMC sample from the target MTN.

Note

`getZigzagSample` is particularly efficient when the target MTN has a random mean and covariance/precision where one can reuse the Zigzag-HMC engine object while updating the mean and covariance. The following example demonstrates such a use.

See Also

[zigzagHMC\(\)](#), [drawLaplaceMomentum\(\)](#)

Examples

```
set.seed(1)
n <- 1000
d <- 10
samples <- array(0, c(n, d))

# initialize MTN mean and precision
m <- rnorm(d, 0, 1)
prec <- rWishart(n = 1, df = d, Sigma = diag(d))[, , 1]
# call createEngine once
engine <- createEngine(dimension = d, lowerBounds = rep(0, d),
  upperBounds = rep(Inf, d), seed = 1, mean = m, precision = prec)

HZZtime <- sqrt(2) / sqrt(min(mgcv::slanczos(
  A = prec, k = 1,
  kl = 1
)[['values']]))

currentSample <- rep(0.1, d)
for (i in 1:n) {
  m <- rnorm(d, 0, 1)
  prec <- rWishart(n = 1, df = d, Sigma = diag(d))[, , 1]
  setMean(engine = engine, mean = m)
  setPrecision(engine = engine, precision = prec)
  currentSample <- getZigzagSample(position = currentSample,
    nutsFlg = FALSE,
    engine = engine,
    stepSize = HZZtime)
  samples[i,] <- currentSample
}
```

harmonicHMC

Sample from a truncated Gaussian distribution with the harmonic HMC

Description

Generate MCMC samples from a d-dimensional truncated Gaussian distribution with constraints $Fx+g \geq 0$ using the Harmonic Hamiltonian Monte Carlo sampler (Harmonic-HMC).

Usage

```

harmonicHMC(
  nSample,
  burnin = 0,
  mean,
  choleskyFactor,
  constrainDirec,
  constrainBound,
  init,
  time = c(pi/8, pi/2),
  precFlg,
  seed = 1,
  extraOutputs = c()
)

```

Arguments

nSample	number of samples after burn-in.
burnin	number of burn-in samples (default = 0).
mean	a d-dimensional mean vector.
choleskyFactor	upper triangular matrix R from Cholesky decomposition of precision or covariance matrix into $R^T R$.
constrainDirec	the k-by-d F matrix (k is the number of linear constraints).
constrainBound	the k-dimensional g vector.
init	a d-dimensional vector of the initial value. init must satisfy all constraints.
time	HMC integration time for each iteration. Can either be a scalar value for a fixed time across all samples, or a length 2 vector of a lower and upper bound for uniform distribution from which the time is drawn from for each iteration.
precFlg	logical. whether choleskyFactor is from precision (TRUE) or covariance matrix (FALSE).
seed	random seed (default = 1).
extraOutputs	vector of strings. "numBounces" and/or "bounceDistances" can be requested, with the latter containing the distances in-between bounces for each sample and hence incurring significant computational and memory costs.

Value

When extraOutputs is empty (default), returns an nSample-by-d matrix of samples.

When extraOutputs contains "numBounces" and/or "bounceDistances", returns a list with elements:

samples	nSample-by-d matrix of samples
numBounces	Vector of bounce counts per sample (if requested)
bounceDistances	List of bounce distances per sample (if requested)

References

Pakman, A. and Paninski, L. (2014). Exact Hamiltonian Monte Carlo for Truncated Multivariate Gaussians. *Journal of Computational and Graphical Statistics*. doi:10.1080/10618600.2013.788448

See Also

`getHarmonicSample()`, `cholesky()`, `getInitialPosition()`

Examples

```
set.seed(1)
d <- 10
A <- matrix(runif(d^2)*2 - 1, ncol=d)
precMat <- t(A) %*% A
R <- cholesky(precMat)
mu <- rep(0, d)
constrainDirec <- diag(d)
constrainBound <- rep(0,d)
initial <- rep(1, d)
results <- harmonicHMC(1000, 1000, mu, R, constrainDirec, constrainBound, initial, precFlg = TRUE)
```

markovianZigzag

Markovian Zigzag Sampler

Description

Sample from a truncated multivariate normal distribution using the Markovian Zigzag process, a continuous-time, non-reversible Markov chain Monte Carlo method based on piecewise deterministic Markov processes (PDMPs).

Usage

```
markovianZigzag(
  nSample,
  burnin = 0,
  mean,
  prec,
  lowerBounds,
  upperBounds,
  init = NULL,
  stepSize = NULL,
  seed = 1,
  numThreads = 1,
  diagnosticMode = FALSE,
  nStatusUpdate = 0L
)
```

Arguments

nSample	Number of samples after burn-in.
burnin	Number of burn-in samples (default = 0).
mean	A d-dimensional mean vector.
prec	A d-by-d precision matrix of the Gaussian distribution.
lowerBounds	A d-dimensional vector specifying the lower bounds. -Inf is accepted.
upperBounds	A d-dimensional vector specifying the upper bounds. Inf is accepted.
init	A d-dimensional vector of the initial value. <code>init</code> must satisfy all constraints. If <code>init = NULL</code> , a random initial value will be used.
stepSize	Step size for the Markovian Zigzag sampler. Default value is the empirically optimal choice: $\sqrt{2}\lambda^{-1/2}$, where λ is the minimal eigenvalue of the precision matrix.
seed	Random seed (default = 1).
numThreads	number of threads for parallel execution (default = 1). Set to 0 for automatic detection of available cores.
diagnosticMode	Logical. TRUE for also returning diagnostic information such as the step size used.
nStatusUpdate	Number of status updates to print during sampling. If 0 (default), no updates are printed.

Value

An nSample-by-d matrix of samples. If `diagnosticMode` is TRUE, a list with additional diagnostic information is returned.

References

Bierkens, J., Roberts, G. O., and Zitt, P.-A. (2019). Ergodicity of the zigzag process. *The Annals of Applied Probability*, 29(4): 2266-2301.

See Also

[getMarkovianZigzagSample\(\)](#), [createEngine\(\)](#)

Examples

```
set.seed(1)
d <- 5
A <- matrix(runif(d^2)*2-1, ncol=d)
precMat <- t(A) %*% A
initial <- rep(1, d)
results <- markovianZigzag(
  nSample = 1000,
  burnin = 1000,
  mean = rep(0, d),
  prec = precMat,
```

```

    lowerBounds = rep(0, d),
    upperBounds = rep(Inf, d)
  )

```

setMean	<i>Set the mean for the target MTN</i>
---------	--

Description

Set the mean vector for a given Zigzag-HMC engine object.

Usage

```
setMean(engine, mean)
```

Arguments

engine	A Zigzag-HMC engine container object.
mean	the mean vector.

See Also

[createEngine\(\)](#), [createNutsEngine\(\)](#)

Examples

```

# First create an engine
engine <- createEngine(dimension = 2,
  lowerBounds = c(-1, -1),
  upperBounds = c(1, 1),
  seed = 123,
  mean = c(0, 0),
  precision = diag(2))

# Update the mean
setMean(engine, mean = c(0.5, 0.5))

```

setPrecision	<i>Set the precision matrix for the target MTN</i>
--------------	--

Description

Set the precision matrix for a given Zigzag-HMC engine object.

Usage

```
setPrecision(engine, precision)
```

Arguments

engine A Zigzag-HMC engine container object.
precision the precision matrix.

See Also

[createEngine\(\)](#), [createNutsEngine\(\)](#)

Examples

```
# First create an engine
engine <- createEngine(dimension = 2,
                       lowerBounds = c(-1, -1),
                       upperBounds = c(1, 1),
                       seed = 123,
                       mean = c(0, 0),
                       precision = diag(2))
# Update with a correlated precision matrix
new_precision <- matrix(c(2, 0.8, 0.8, 2), nrow = 2)
setPrecision(engine, precision = new_precision)
```

zigzagHMC

Sample from a truncated Gaussian distribution

Description

Generate MCMC samples from a d-dimensional truncated Gaussian distribution with element-wise truncations using the Zigzag Hamiltonian Monte Carlo sampler (Zigzag-HMC).

Usage

```
zigzagHMC(
  nSample,
  burnin = 0,
  mean,
  prec,
  lowerBounds,
  upperBounds,
  init = NULL,
  stepSize = NULL,
  nutsFlg = FALSE,
  precondition = FALSE,
  seed = 1,
  numThreads = 1,
  diagnosticMode = FALSE
)
```

Arguments

nSample	number of samples after burn-in.
burnin	number of burn-in samples (default = 0).
mean	a d-dimensional mean vector.
prec	a d-by-d precision matrix of the Gaussian distribution.
lowerBounds	a d-dimensional vector specifying the lower bounds. -Inf is accepted.
upperBounds	a d-dimensional vector specifying the upper bounds. Inf is accepted.
init	a d-dimensional vector of the initial value. <code>init</code> must satisfy all constraints. If <code>init = NULL</code> , a random initial value will be used.
stepSize	step size for Zigzag-HMC or Zigzag-NUTS (if <code>nutsFlg = TRUE</code>). Default value is the empirically optimal choice: $\sqrt{2}(\lambda)^{-1/2}$ for Zigzag-HMC and $0.1(\lambda)^{-1/2}$ for Zigzag-NUTS, where λ is the minimal eigenvalue of the precision matrix.
nutsFlg	logical. If TRUE the No-U-Turn sampler will be used (Zigzag-NUTS).
precondition	logical. If TRUE, the precision matrix will be preconditioned so that its diagonals (i.e. conditional variances) are all 1.
seed	random seed (default = 1).
numThreads	number of threads for parallel execution (default = 1). Set to 0 for automatic detection of available cores.
diagnosticMode	logical. TRUE for also returning diagnostic information such as the stepsize used.

Value

When `diagnosticMode = FALSE` (default), returns an nSample-by-d matrix of samples.

When `diagnosticMode = TRUE`, returns a list with elements:

samples	nSample-by-d matrix of samples
stepsize	The step size used for sampling

References

Nishimura, A., Zhang, Z., and Suchard, M. A. (2024). Zigzag path connects two Monte Carlo samplers: Hamiltonian counterpart to a piecewise deterministic Markov process. *Journal of the American Statistical Association*, 1-13.

Nishimura, A., Dunson, D. B., and Lu, J. (2020). Discontinuous Hamiltonian Monte Carlo for discrete parameters and discontinuous likelihoods. *Biometrika*, 107(2): 365-380.

See Also

`getZigzagSample()`, `createEngine()`, `createNutsEngine()`, `setMean()`, `setPrecision()`

Examples

```
set.seed(1)
d <- 10
A <- matrix(runif(d^2)*2-1, ncol=d)
precMat <- t(A) %*% A
initial <- rep(1, d)
results <- zigzagHMC(nSample = 1000, burnin = 1000, mean = rep(0, d), prec = precMat,
lowerBounds = rep(0, d), upperBounds = rep(Inf, d))
```

Index

cholesky, [2](#)
cholesky(), [12](#)
createEngine, [3](#)
createEngine(), [5](#), [13–16](#)
createNutsEngine, [4](#)
createNutsEngine(), [14–16](#)

drawLaplaceMomentum, [5](#)
drawLaplaceMomentum(), [10](#)

getHarmonicSample, [6](#)
getHarmonicSample(), [12](#)
getInitialPosition, [7](#)
getInitialPosition(), [12](#)
getMarkovianZigzagSample, [8](#)
getMarkovianZigzagSample(), [13](#)
getZigzagSample, [9](#)
getZigzagSample(), [16](#)

harmonicHMC, [10](#)
harmonicHMC(), [2](#), [6](#), [7](#)

markovianZigzag, [12](#)
markovianZigzag(), [4](#), [7](#), [8](#)

setMean, [14](#)
setMean(), [4](#), [5](#), [16](#)
setPrecision, [14](#)
setPrecision(), [4](#), [5](#), [16](#)

zigzagHMC, [15](#)
zigzagHMC(), [4](#), [5](#), [7](#), [10](#)