

Package ‘hdtg’

October 28, 2025

Title Generate Samples from Multivariate Truncated Normal Distributions

Version 0.2.3

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 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 7.3.2.9000

Imports Rcpp, RcppParallel, mgcv, stats, Rdpack

RdMacros Rdpack

LinkingTo Rcpp, RcppEigen, RcppParallel

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

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

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 2025-10-28 09:40:02 UTC

Contents

cholesky	2
createEngine	2
createNutsEngine	3
drawLaplaceMomentum	4
getInitialPosition	5
getMarkovianZigzagSample	5
getZigzagSample	6
harmonicHMC	7
setMean	8
setPrecision	9
zigzagHMC	9
Index	11

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 $A = U^tU$.

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  
)
```

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.

Value

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

createNutsEngine	<i>Create a Zigzag-NUTS engine object</i>
------------------	---

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  
)
```

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.

Value

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

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.

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 lower bounds.

Value

an eligible d-dimensional initial vector.

getMarkovianZigzagSample	<i>Draw one Markovian zigzag sample</i>
--------------------------	---

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.

getZigzagSample	<i>Draw one MTN sample with Zigzag-HMC or Zigzag-NUTS</i>
-----------------	---

Description

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

Usage

```
getZigzagSample(position, momentum = NULL, nutsFlg, engine, stepZZHMC = 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).
stepZZHMC	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.

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,
  k1 = 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(sexp = engine$engine, mean = m)
  setPrecision(sexp = engine$engine, precision = prec)
  currentSample <- getZigzagSample(position = currentSample, nutsFlg = FALSE,
    engine = engine, stepZZHMC = 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 = NULL,
  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

samples: nSample-by-d matrix of samples or, if extraOutputs is non-empty, a list of samples and the extra outputs.

References

Pakman A, Paninski L (2014). "Exact Hamiltonian Monte Carlo for truncated multivariate Gaussians." *Journal of Computational and Graphical Statistics*, **23**(2), 518–542.

Examples

```
set.seed(1)
d <- 10
A <- matrix(runif(d^2)*2 - 1, ncol=d)
Sigma <- t(A) %*% A
R <- cholesky(Sigma)
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 = FALSE)
```

setMean

Set the mean for the target MTN

Description

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

Usage

```
setMean(sexp, mean)
```

Arguments

sexp	pointer to a Zigzag-HMC engine object.
mean	the mean vector.

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(sexp, precision)
```

Arguments

sexp	pointer to a Zigzag-HMC engine object.
precision	the precision matrix.

zigzagHMC	<i>Sample from a truncated Gaussian distribution</i>
-----------	--

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 = NULL,
  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. $-\text{Inf}$ 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 <code>TRUE</code> the No-U-Turn sampler will be used (Zigzag-NUTS).
precondition	logical. If <code>TRUE</code> , the precision matrix will be preconditioned so that its diagonals (i.e. conditional variances) are all 1.
seed	random seed (default = 1).
diagnosticMode	logical. <code>TRUE</code> for also returning diagnostic information such as the stepsize used.

Value

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

References

- Nishimura A, Zhang Z, Suchard MA (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 DB, Lu J (2020). “Discontinuous Hamiltonian Monte Carlo for discrete parameters and discontinuous likelihoods.” *Biometrika*, **107**(2), 365–380.

Examples

```
set.seed(1)
d <- 10
A <- matrix(runif(d^2)*2-1, ncol=d)
covMat <- t(A) %*% A
precMat <- solve(covMat)
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](#)
createEngine, [2](#)
createNutsEngine, [3](#)

drawLaplaceMomentum, [4](#)

getInitialPosition, [5](#)
getMarkovianZigzagSample, [5](#)
getZigzagSample, [6](#)

harmonicHMC, [7](#)

setMean, [8](#)
setPrecision, [9](#)

zigzagHMC, [9](#)