An introduction to GPHS

Contents

Introduction

Hyperparameter selection in Bayesian inference is very crucial as it specifies the covariance structure. Typically, we integrate over these hyperparameters but they often correspond to high-dimensional integrals which becomes analytically intractable. This integration is often performed by MCMC, but MCMC converges very slowly and it needs expert tuning. Hence, we need special techniques to handle these situations.
`GPHS` is an R package that provides various methods to perform hyperparameter selection in Gaussian processes. It has three algorithms to perform the update of hyperparameters. We have Metropolis-Hastings algorithm and the Slice sampling algorithm for a Surrogate data model as discussed in Murray and Adams (2010). We also have another function that implements the Elliptical Slice Sampling as given in Murray et al (2010) and Nishihara, Murray and Adams (2014). In addition to these algorithms, `GPHS` also has a function to invert symmetric matrices based on Cholesky decomposition, which is faster than the default `solve()` function in R.

Elliptical Slice Sampling (ESS)

Elliptical Slice Sampling, first discussed by Murray and Adams in 2010, is a MCMC technique that provides a way to sample along an ellipse with an adaptive step-size. It is designed to sample from posterior of the form $\pi(x) \propto L(x)N(x;\mu,\Sigma)$, where $L(\cdot)$ denotes the likelihood function and $N(\mu,\Sigma)$ denotes a multivariate Gaussian prior. The major advantage of this method is that it has no free tuning parameters. It relies on the fact that Gaussian prior mixes rapidly even when there is strong dependency induced by the covariance structure.
ESS uses the invariance property that when x and ν are drawn independently from $N(\mu,\Sigma)$ distribution then the linear combination:

$$x' = (x-\mu)\cos(\theta) + (\nu-\mu)\sin(\theta)$$

is marginally distributed as $N(\mu,\Sigma)$ for any $\theta \in [0, 2\pi]$, although x' is still correlated with x and ν.

Surrogate Data model:

As discussed in Murray and Adams(2010), we consider the following generic form of the model:

$$\begin{aligned} \text{covariance hyperparameters } \theta &\propto p_h \\ \text{latent variable } f &\propto N(0, \Sigma_\theta) \\ \text{conditional likelihood } P(data|f) &= L(f) \end{aligned}$$

We assume that the covariance matrix Σ_θ is arbitrary positive definite matrix. For non-Gaussian likelihoods, we sample from the joint posterior:

$$P(f, \theta) \propto L(f) N(f; 0, \Sigma_\theta) p_h(\theta)$$

To guide joint proposals of the hyperparameters and the latent variables, we create an auxiliary variable that introduces a surrogate gaussian observation. The Gaussian model is augmented with a noisy version of the true latent variable, g:

$$P(g|f, \theta) = N(g; f, S_\theta)$$

Here, S_θ can be set by hand to a fixed value or a value depending on the current hyperparameters. We first draw the auxiliary variable from the marginal distribution:

$$P(g|\theta) = N(g; 0, \Sigma_\theta + S_\theta)$$

and then sample the latent values conditioned on the auxiliary variables:

$$P(f|g, \theta) = N(f; m_{\theta,g}, R_\theta), \text{ where } R_\theta = \left(\Sigma_\theta^{-1} + S_\theta^{-1}\right)^{-1} \text{ and } m_{\theta,g} = R_\theta S_\theta^{-1} g$$

The sampling process can be described like a draw from a spherical Gaussian as follows:

$$\eta \sim N(0, I), \ f = L_{R_\theta} \eta + m_{\theta,g}, \text{ where } L_{R_\theta} L_{R_\theta}^T = R_\theta$$

We update the hyperparameters θ conditioned on the "whitened" variables η and the surrogate data g.

$$P(\theta|\eta, g, data) \propto P(\theta, \eta, g, data) \propto L(f(\theta, \eta, g)) N(g; 0, \Sigma_\theta + S_\theta) p_h(\theta)$$

.

The acceptance rule in the Metropolis-Hastings algorithm contains a ratio of the above. We implement this algorithm in the function `SurrogateMH`.

However, the major problem with the Metropolis-Hastings algorithm is that the proposal distribution needs to be set and tuned properly. The efficiency of the algorithms depend on the scale σ of the proposal distribution. Slice sampling, a family of adaptive procedures, is much more robust to the choice of scale parameter. We implement a possible type of slice sampling in the function `SurrogateSS`.

Installation

You can install the most recent version of 'GPHS' package from GitHub using the following commands:

```r
# Plain installation
devtools::install_github("niladrik/GPHS")

# For installation with vignette
devtools::install_github("niladrik/GPHS", build_vignettes = TRUE)
```

Quick Start

This section will convey a idea of the contents and the usage of the functions in the package. We would briefly mention the functions that are there in the package, their inputs and their outputs.
Firstly, we load the package.

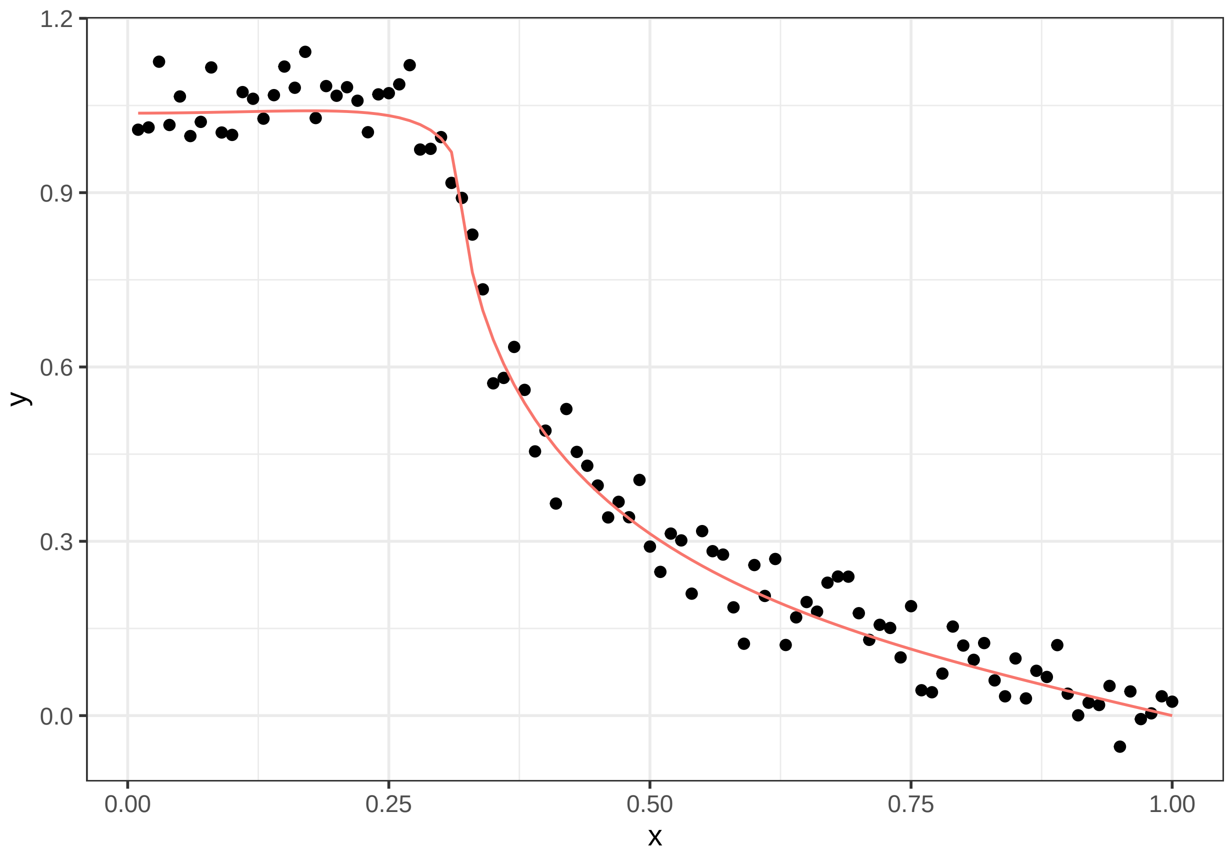
```r
library(GPHS)
```

We consider the model: $y = f + \epsilon$ where y is the observed value, f is the latent variable, and ϵ is a noise term. We now create a dataset that we are going to use to depict the functionality of our functions:

```r
# loading ggplot for plots
library(ggplot2)

# setting the seed for reproducibility of the data
set.seed(12345)

f_data <- function(x){
  func <- 0
  for(j in 1:5000){
    func <- func + j^(-1.7) * sin(j) * cos(pi * (j - 0.5) * x)
  }
  return(func)
}

x = seq(1, 1e-3, -0.01)
# true value (latent variable)
ff = f_data(x)

# observed value is anoisy version of the true value
y = ff + rnorm(length(x), 0, sd(ff)/10)
```

Thus, the function and the data points we generated look like:

```r
  ggplot(data = as.data.frame(cbind(x, y, ff)), aes(x = x, y = y)) + geom_point() +
  ↪  geom_line(aes(x = x, y = ff, color = "red")) + theme_bw() + theme(legend.position =
  ↪  "none")
```



For the purpose of illustration, we consider a sample of 10 data points from this set as our data in hand.

```
# selecting a sample of 100 data points
x.train = sample(x = x, size = 10, replace = FALSE)
y.train = sample(x = y, size = 10, replace = FALSE)

# dataset in hand
training_data = cbind(x.train, y.train)
```

We consider the matern covariance function for the covariance of the latent variable, and the length scale l is our hyperparameter of interest. We consider the prior of l to be a log-normal distribution, i.e, $log(l) \sim N(0, 1)$.

```
# prior function
prior <- function(x, mu = 0, sigma = 1){
  return(dnorm(log(x), mu, sigma))
}
```

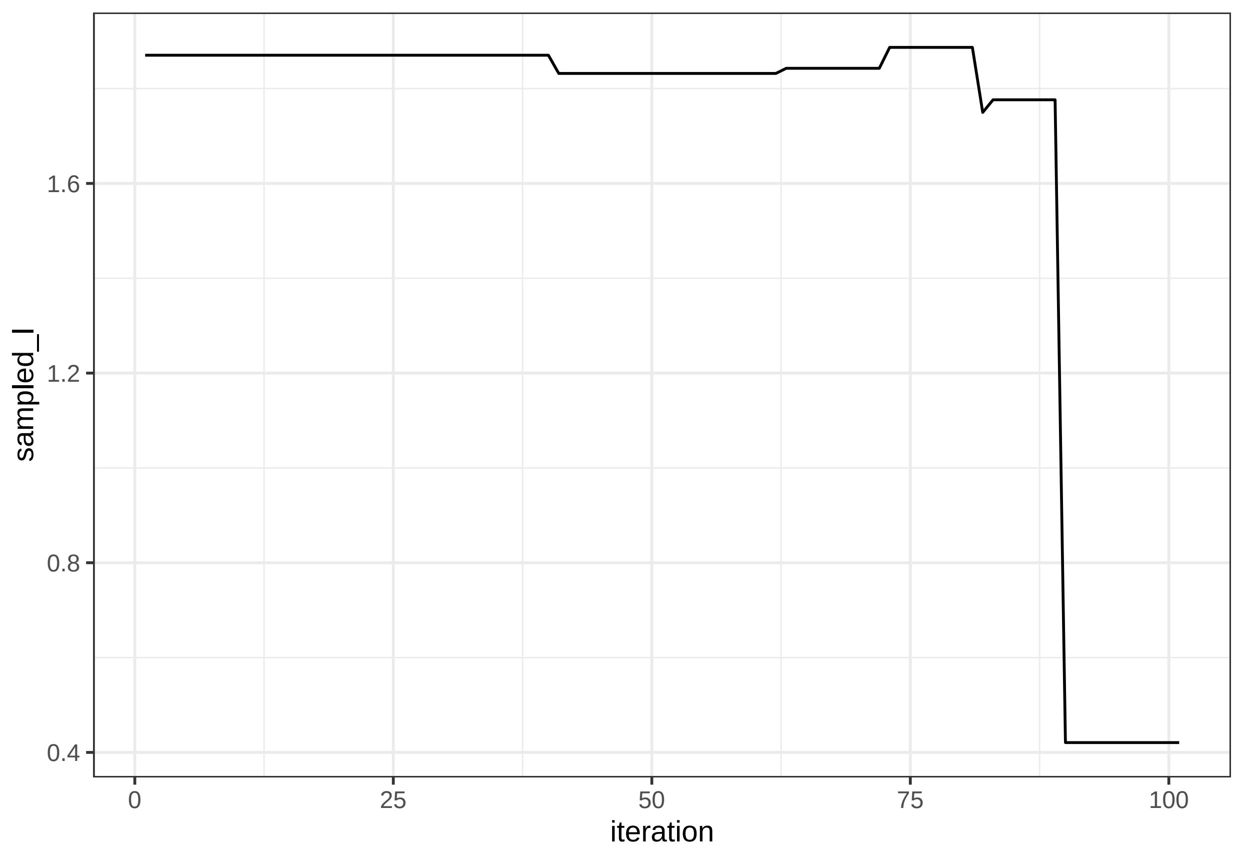
We also wish to pre-compute a range of l such that the matern correlation lies between 0.05 and 0.95.

```
## matern covariance function

matern = function(d , phi, nu = 3){ # previous value of nu = 3
  ifelse(d > 0, (sqrt(2 * nu) * d * phi)^nu / (2^(nu - 1) * gamma(nu)) *
           besselK(x = d * phi * sqrt(2 * nu), nu = nu), 1.0)
}

## check for 0.05 <= k_mat(1/l) / k(0) <= 0.95
## Here we fix the maximum distance to be 1 since x varies from (0.01, 1)
## and calculate these values
l_range = seq(0.2, 6.36, 0.01)
k_range = sapply(l_range, function(l){return(matern(1, 1/l, 3) / matern(0, 1/l, 3))})

l_min = min(l_range[k_range > 0.05])
l_max = max(l_range[k_range < 0.95])
c(l_min, l_max)
#> [1] 0.39 3.78
```

SurrogateMH

This function provides an update of the latent variable f and the hyperparameter θ using the Metropolis-Hastings algorithm. It takes as input the initial state of the hyperparameter θ(scalar or vector) and the latent variable f(scalar or vector), the data in hand(matrix or data frame), the prior of the hyperparameter(a function), and the conditional likelihood of the data conditioned on the latent variable(a function). It returns the updated θ and f
We run the update for 100 times and then make a trace plot of the l sampled from the posterior distribution.

```
n = 1e2
theta = vector("numeric", n)
f = vector("list", n)
# randomly selecting a start point between l_min and l_max
theta[1] = runif(1, l_min, l_max)
# choosing the starting point of f to be y/2
f[[1]] = training_data[, 2]/2

# We choose the conditional distribution of the data|f as N(f, tau*I)
l <- function(x, mu = rep(0, length(x)), sigma = diag(1, length(x)), tau = 0.1){
  return(dmvnorm_own(x, mu, sigma, tau))
}
for(i in 1:n){
  result = SurrogateMH(theta = theta[i], f = f[[i]], data = training_data, p = prior, l)
  theta[i + 1] = result$theta
  f[[i + 1]] = result$f
}
# plot(theta, type = "l")
ggplot(data = data.frame(iteration = 1:(n+1), sampled_l = theta), aes(x = iteration, y =
↪  sampled_l)) + geom_line() + theme_bw()
```



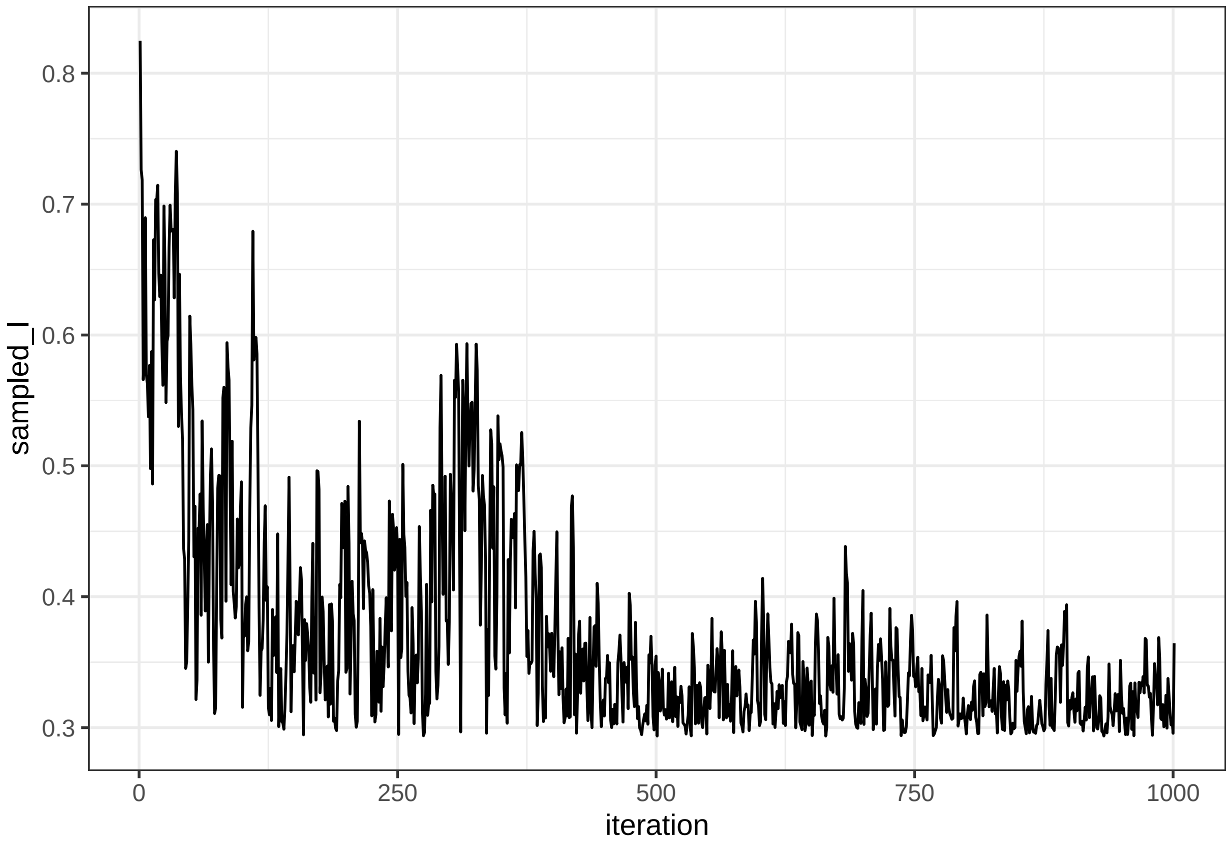
As is evident from the plot, the hyperparameter learning is not that good. The major problem with this algorithm is that it needs expert tuning. Hence, we go for the Slice sampling of the hyperparameters and the latent variable.

SurrogateSS

This function updates the hyperparameter and the latent variable according to the slice sampling algorithm provided in Adams and Murray(2010). It takes as input the initial state of the hyperparameter θ(scalar or vector) and the latent variable f(scalar or vector), the data in hand(matrix or data frame), scale parameter σ (scalar), the prior of the hyperparameter(a function), the conditional likelihood of the data conditioned on the latent variable(a function) and the maximum of number of iterations per update (scalar). It returns the updated θ and f
We run the algorithm for 1000 times. The user can set n to be any higher value to get greater number of updates. Generally speaking, the Slice Sampling update converges much faster than the Metropolis-Hastings algorithm.

```
# set the number of updates needed
n = 1e3
f = vector("list", length = n + 1)
th = vector("numeric", n + 1)

## choosing the starting points
f[[1]] = training_data[, 1]
th[1] = runif(1, l_min, l_max)

## choosing the scale parameter
sigma = 0.2
```

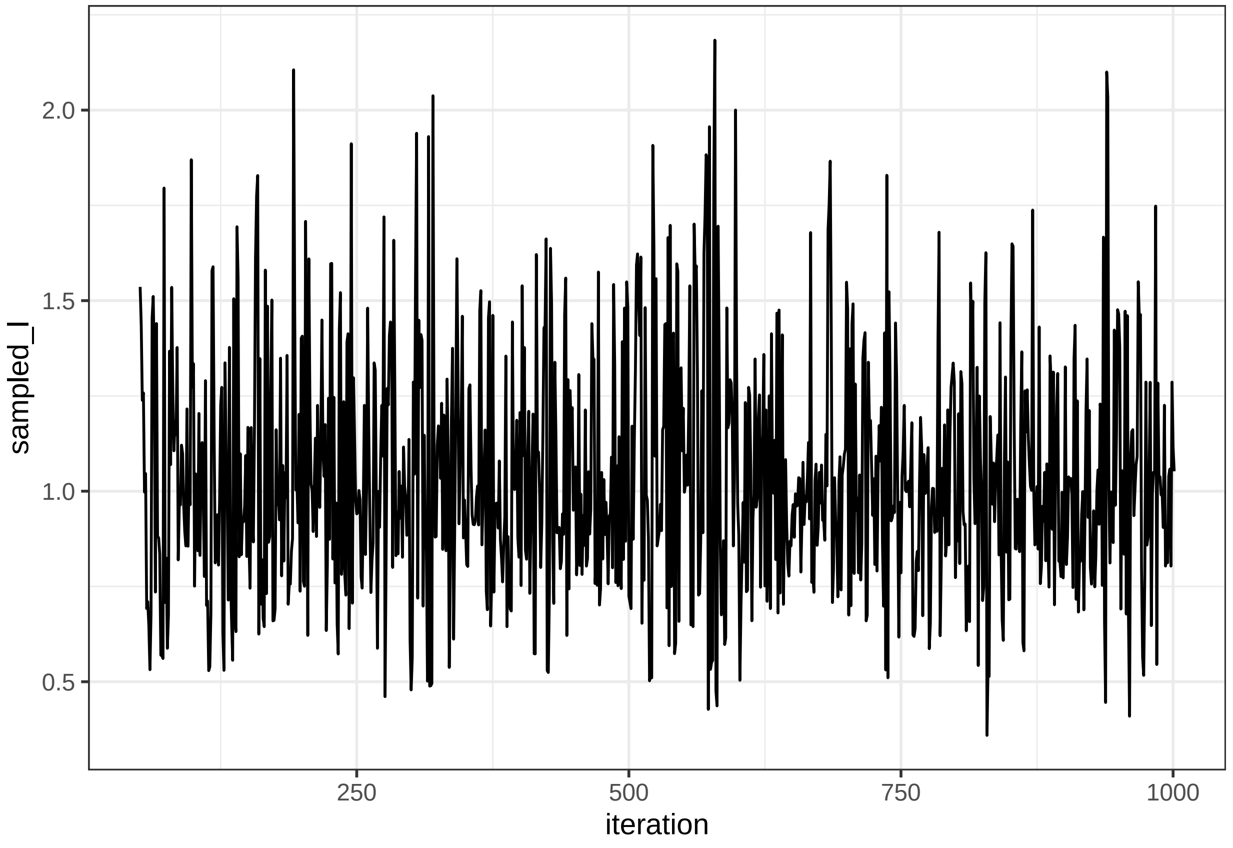
```r
i = 1
for(i in 1:n){
results = SurrogateSS(theta = th[i], f = f[[i]], sigma = sigma, data =   training_data, l
↪  = dmvnorm_own, p = prior, niter = 100)
f[[i+1]] = results$f
th[i + 1] = results$theta
}
# traceplot of sampled l
ggplot(data = data.frame(iteration = 1:(n+1), sampled_l = th), aes(x = iteration, y =
↪  sampled_l)) + geom_line() + theme_bw()
```



`ESS`

The function ESS is defined according to the Elliptical Slice Sampling algorithm as given in Nishihara, Murray and Adams(2014). For simplicity, we consider the log-likelihood function $L(x) \sim N(\mu, \Sigma)$, given by the `log.L()` function. If μ and σ are missing then we assume the standard multivariate normal distribution. We run the chain for $n = 1000$ times and the trace plot for the sampled `l`(the length scale hyperparameter) are plotted.

```r
log.L = function(y, mu = NULL, sigma = NULL){

  p = length(y)
  if(is.null(mu)){
    mu = rep(0, p)
  }
  if(is.null(sigma)){
    sigma = diag(1, p)
  }
  d = mvtnorm::dmvnorm(y, mu, sigma)
  log.val = log(d)
  return(log.val)
}
# choosing the number of updates
n = 1e3
f = vector("numeric", n + 1)

# initial value
f[1] = 5

# dimension of the vector f
p = 1

mu = rep(0, p)
sigma = diag(0.1, p)
for(i in 1:n){
  f[i + 1] = ESS(f[i], mu,  sigma, log.L, 100)
}

ggplot(data = data.frame(iteration = 51:(n+1), sampled_l = exp(f[-(1:50)])), aes(x =
↪  iteration, y = sampled_l)) + geom_line() + theme_bw()
```



mySolve

Often while performing computation, there might be certain small numerical errors that might result in a matrix no longer being positive definite. This function is useful to invert the matrix in such scenario. It generates the Eigen decomposition of the matrix and then calculates the inverse by suitably truncating the decomposition. In addition to the inverse, it also returns the determinant and the square root decomposition of the matrix

```
## a matrix to invert
A = diag(2, nrow = 2)
mySolve(A)
#> $inv
#>      [,1] [,2]
#> [1,]  0.5  0.0
#> [2,]  0.0  0.5
#>
#> $det
#> [1] 4
#>
#> $sqrtDeco
#>          [,1]     [,2]
#> [1,] 1.414214 0.000000
#> [2,] 0.000000 1.414214
```

dmvnorm_own

This function calculates the density of a multivariate normal distribution with the help of `mySolve` function to invert the covariance matrix of the form $\tau\Sigma$.

```
## defining the quantile vector
x = c(0, 0)
dmvnorm_own(y = x)
#> [1] 0.1575713
```

References

Murray, I., & Adams, R. P. (2010). Slice sampling covariance hyperparameters of latent Gaussian models. *Advances in neural information processing systems, 23.* https://doi.org/10.48550/arXiv.1006.0868

Murray, I., Adams, R., & MacKay, D. (2010, March). Elliptical slice sampling. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 541-548). JMLR Workshop and Conference Proceedings. https://doi.org/10.48550/arxiv.1001.0175

Nishihara, R., Murray, I., & Adams, R. P. (2014). Parallel MCMC with generalized elliptical slice sampling. *The Journal of Machine Learning Research, 15(1)*, 2087-2112. https://doi.org/10.48550/arXiv.1210.7477