HMC Diagnostics

This notebook runs a few statistical diagnostics on the logged variables of an HMC run. The experiment is defined through the `path` variable which defines the directory where the logging was made. This notebook assumes that the files provided are:

- `log_gamma.npy`
- `log_lambda.npy`
- `parameter*.npy`

```
library(RcppCNPy)
library(coda)
library(ggplot2)

path <- "../etc/diagnostic/flux_nn"
files <- list.files(path=path, pattern="*.npy", full.names=TRUE, recursive=FALSE)
files <- files[files != "../etc/diagnostic/flux_nn/full_parameters.npy"]
files
```

```
## [1] "../etc/diagnostic/flux_nn/log_gamma.npy"
## [2] "../etc/diagnostic/flux_nn/log_lambda.npy"
## [3] "../etc/diagnostic/flux_nn/parameter0.npy"
## [4] "../etc/diagnostic/flux_nn/parameter1.npy"
## [5] "../etc/diagnostic/flux_nn/parameter2.npy"
## [6] "../etc/diagnostic/flux_nn/parameter3.npy"
## [7] "../etc/diagnostic/flux_nn/parameter4.npy"
```

Geweke Z-score

We first look at the Geweke Z-score for all our variables. This test compares the distribution between the beginning and the end of each Markov chain.

```
for(file in files){
  x <- npyLoad(file)
  y <- mcmc.list(apply(x, 2, function(col) {mcmc(col)}, simplify = FALSE))
  name <- basename(file)
  name <- substr(name, 1, nchar(name) - 4)
  cat("===== Test for", name, "=====\n")
  print(geweke.diag(y[[1]]))
}
```

```
## ===== Test for log_gamma =====
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1
## -1.335
##
## ===== Test for log_lambda =====
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1
## -1.006
##
## ===== Test for parameter0 =====
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##     var1
## -0.04147
##
## ===== Test for parameter1 =====
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1
## -1.157
##
```

```
## ===== Test for parameter2 =====
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1
## 0.6449
##
## ===== Test for parameter3 =====
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1
## 0.4811
##
## ===== Test for parameter4 =====
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    var1
## -0.9189
```

This test is a success if all the `var1` values respect $-2 \leq var1 \leq 2$.

Gelman-Rubin test

The Gelman-Rubin test uses several Markov chains for each variable. It computes a ratio using both the within-chain variance and the between-chain variance. If the chain converges correctly, it should be close to 1.

```
for(file in files){
  x <- npyLoad(file)
  y <- mcmc.list(apply(x, 2, function(col) {mcmc(col)}, simplify = FALSE))
  name <- basename(file)
  name <- substr(name, 1, nchar(name) - 4)
  cat("===== Test for", name, "=====\n")
  print(gelman.diag(y))
}
```

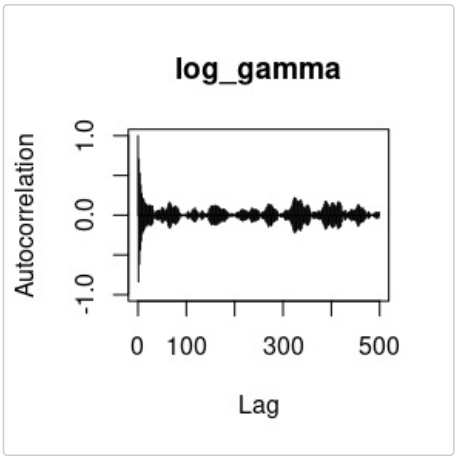
```
## ===== Test for log_gamma =====
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
##
## ===== Test for log_lambda =====
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.12       1.43
##
## ===== Test for parameter0 =====
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
##
## ===== Test for parameter1 =====
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
##
## ===== Test for parameter2 =====
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.01       1.01
##
## ===== Test for parameter3 =====
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

```
##
## ===== Test for parameter4 =====
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

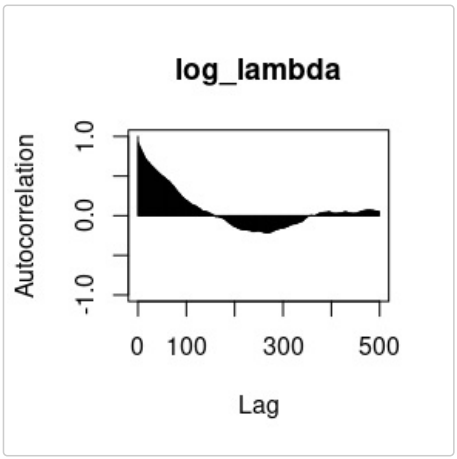
Auto-correlation

Finally, for each parameter we plot its auto-correlation for different lag values:

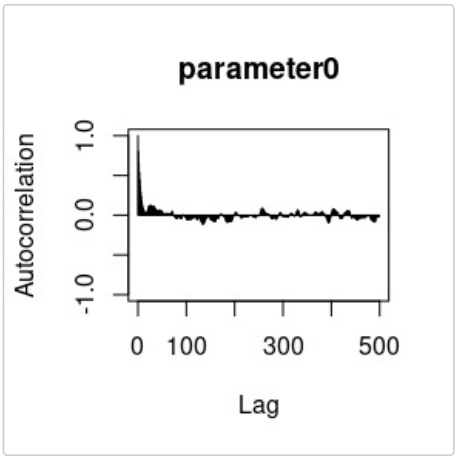
```
file <- files[1]
x <- npyLoad(file)
y <- mcmc.list(apply(x, 2, function(col) {mcmc(col)}, simplify = FALSE))
name <- basename(file)
name <- substr(name, 1, nchar(name) - 4)
autocorr.plot(y[[1]],lag.max=dim(x)[1]/4, main=name)
```




```
file <- files[2]
x <- npyLoad(file)
y <- mcmc.list(apply(x, 2, function(col) {mcmc(col)}, simplify = FALSE))
name <- basename(file)
name <- substr(name, 1, nchar(name) - 4)
autocorr.plot(y[[1]],lag.max=dim(x)[1]/4, main=name)
```




```
file <- files[3]
x <- npyLoad(file)
y <- mcmc.list(apply(x, 2, function(col) {mcmc(col)}, simplify = FALSE))
name <- basename(file)
name <- substr(name, 1, nchar(name) - 4)
autocorr.plot(y[[1]],lag.max=dim(x)[1]/4, main=name)
```




```
file <- files[4]
x <- npyLoad(file)
y <- mcmc.list(apply(x, 2, function(col) {mcmc(col)}, simplify = FALSE))
name <- basename(file)
name <- substr(name, 1, nchar(name) - 4)
autocorr.plot(y[[1]],lag.max=dim(x)[1]/4, main=name)
```