The Hamiltonian Monte Carlo and the No-U-Turn Sampler

Jingyue Lu Marco Palma

October 18, 2017

Abstract

We present the R package 'NUTS', which contains functions for Hamiltonian Monte Carlo and one of its extensions called No-U-Turn Sampler with Dual Averaging.

1 Introduction

This project investigates the No-U-Turn Sampler (NUTS). M.D. Hoffman and A. Gelman introduced NUTS in 2011 to address the tuning issues involved in Hamiltonian Monte Carlo (HMC) algorithm. To be more specific, we start by introducing the theoretical ideas behind HMC and NUTS. Then we implement both methods in R to compare their performances. Especially, we are interested in how effective NUTS is as an extension of HMC.

2 Hamiltonian Monte Carlo

The Hamiltonian Monte Carlo (also known as hybrid Monte Carlo, introduced by Duane et al., 1987) is a Markov chain Monte Carlo method that avoids simple random work behaviour by proposing remote states for Metropolis algorithm, thereby achieving rapid convergence to a target distribution. The idea of HMC is to introduce a d-dimensional vector r called momentum (where d is the dimension of the parameter vector θ), independently drawn from a distribution we can easily sample from (usually a standard multivariate normal). The unnormalized joint density can be written as

$$p(\theta, r) \propto \exp\left\{\mathcal{L}(\theta) - \frac{1}{2} r \cdot r\right\}$$

where $\mathcal{L}$ is the unnormalized logarithm of the joint posterior density of $\{\theta_k\}$ and $(\cdot)$ is an inner product.

Samples of (θ, r) are obtained by using the Störmer-Verlet *leapfrog integrator*. Given the gradient $\nabla_\theta \mathcal{L}$ and the step size ϵ, the updates proceed as follows:

$$r^{t+\epsilon/2} = r^t + \epsilon/2 \nabla_\theta \mathcal{L}(\theta^t) \quad \theta^{t+\epsilon} = \theta^t + \epsilon r^{t+\epsilon/2} \quad r^{t+\epsilon} = r^{t+\epsilon/2} + (\epsilon/2)\nabla_\theta \mathcal{L}(\theta^{t+\epsilon}).$$

The leapfrog procedure is applied L times in order to generate a pair $(\tilde{\theta}, \tilde{r})$. Then the proposal for the m-th iteration of the Markov chain is $(\tilde{\theta}, -\tilde{r})$ (where the minus sign for r is only used to guarantee time reversibility). According to Metropolis ratio, the proposal is accepted with a probability

$$\alpha = \min\left\{1, \frac{p(\tilde{\theta}, \tilde{r})}{p(\theta^{m-1}, r^*)}\right\}$$

where r^* is the resampled momentum at the m-th iteration.

Despite the increase in efficiency, the usability and the performances of HMC are affected not only by the computation of $\nabla_\theta \mathcal{L}$ (that can be addressed via numerical procedures) but also by the step size ϵ and number of steps L chosen within the leapfrog. Indeed, when ϵ is too large the acceptance rates will be low, whereas for small values of ϵ there will be a waste of computation because of the tiny steps. In terms of L, if it is too small the samples will be close to each other, but if it is too large the trajectory in the parameters space will loop back to the previous steps.

3 No-U-Turn Sampler

NUTS addresses the problem of tuning the number of steps L. In short, NUTS repeatedly doubles the length of the current trajectory until increasing L no longer leads to an increased distance between the initial θ and a newly proposed $\tilde{\theta}$. That is, the $\tilde{\theta}$ makes a "U-turn".

3.1 Derivation of simplified NUTS algorithm

The derivation of NUTS algorithm can be divided into two parts: the conditions this algorithm has to satisfy in order to be theoretically sound and the criteria NUTS uses to stop the doubling procedure.

To simplify the derivation of NUTS, Hoffman and Gelman introduced a slice variable u. Simplified NUTS considers the augmented model

$$p(\theta, r, u) \propto \mathbb{I}\left[u \in \left[0, \exp\left\{\mathcal{L}(\theta) - \frac{1}{2} r \cdot r\right\}\right]\right],$$

where $\mathbb{I}[\cdot]$ is 1 if $u \in \left[0, \exp\{\mathcal{L}(\theta) - \frac{1}{2} r \cdot r\}\right]$ is true and 0 otherwise. Introducing u renders the conditional probabilities $p(u|\theta, r) \sim \text{Unif}(u; [0, \exp\{\mathcal{L}(\theta) - \frac{1}{2} r \cdot r\}])$ and $p(\theta, r|u) \sim \text{Unif}(\theta', r'| \exp\{\mathcal{L}(\theta) - \frac{1}{2} r \cdot r\} \geq u)$ and hence simplifies the simulation.

In terms of theoretical requirements, the simplified NUTS algorithm must not only leave the target distribution invariant but also guarantee the time reversibility. Under several mild conditions, NUTS uses the following procedure to sample θ^{t+1} from θ^t to achieve invariant target distribution:

1. sample $r \sim \mathcal{N}(0, I)$,
2. sample $u \sim \text{Unif}\left(\left[0, \exp\left\{\mathcal{L}(\theta^t) - \frac{1}{2} r \cdot r\right\}\right]\right)$,
3. sample $\mathcal{B}$, $\mathcal{C}$ from their conditional distribution $p(\mathcal{B}, \mathcal{C}|\theta^t, r, u, \epsilon)$,
4. sample $\left(\theta^{t+1}, r\right)$ uniformly from the set $\mathcal{C}$.

Here, $\mathcal{C}$ is a set of candidate position-momentum states while $\mathcal{B}$ is the set of all position-momentum states computed by leapfrog integrator during each NUTS iteration. Clearly, $\mathcal{C} \subseteq \mathcal{B}$. For the purpose of this project, we omit the proof of the validity of the procedure but only state the key observations and results. We first point out that steps 1, 2, 3 constitute a valid Gibbs sampling update for r, u, $\mathcal{B}$, $\mathcal{C}$. Secondly, as a result of the prequiresites of the above procedure, we use the following condition to determine whether a state in $\mathcal{B}$ is also in $\mathcal{C}$:

$$(\theta', r') \in \mathcal{C}, \qquad \text{if} \quad u \leq \exp\left\{\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r'\right\} \tag{C.1}$$

For what concerns time reversibility, it is important as it ensures the algorithm converge to the target distribution. NUTS uses a recursive algorithm to preserve it. Recall that, to find a trajectory length for each NUTS iteration, NUTS doubles the current trajectory repeatedly until an U-turn is encountered. To simulate forward and backward movement in time, during each doubling, a new subtrajectory is traced by leapfrog either forward from the rightmost state or backward from the leftmost state until the stopping criteria are met. To illustrate, we assume the starting point is (θ_0^1, r), where the subscript is the number of doubling steps and the superscript is the index of the state in that doubling step. Also, let $v \in \{1, -1\}$ be randomly chosen in each doubling step to determine the direction of movement. We are interested in the path for θ:
Step j=1: (v= 1) $\Rightarrow \theta_0^1 \rightarrow \theta_1^1$.
Step j=2: (v= 1) $\Rightarrow \theta_0^1 \rightarrow \theta_1^1 \rightarrow \theta_2^1 \rightarrow \theta_2^2$.
Step j=3: (v=-1) $\Rightarrow \theta_3^4 \leftarrow \theta_3^3 \leftarrow \theta_3^2 \leftarrow \theta_3^1 \leftarrow \theta_0^1 \rightarrow \theta_1^1 \rightarrow \theta_2^1 \rightarrow \theta_2^2$.
In the example above, we build a three-step path. We can also think the path after each step j as a binary

tree of height j, so final path is a binary tree of height 3

$$\underbrace{\underbrace{\underbrace{\theta_3^4 \leftarrow \theta_3^3}_{\text{level 1}} \leftarrow \underbrace{\theta_3^2 \leftarrow \theta_3^1}_{\text{level 1}}}_{\text{level 2}} \leftarrow \underbrace{\underbrace{\theta_0^1 \rightarrow \theta_1^1}_{\text{level 1}} \rightarrow \underbrace{\theta_2^1 \rightarrow \theta_2^2}_{\text{level 1}}}_{\text{level 2}}}_{\text{level 3}}.$$

Finally, we discuss the stopping criteria used in NUTS. A straightforward and essential stopping condition for NUTS implements the idea of no-U-turn. We observe that when $\tilde{\theta}$ makes a U-turn, the following derivative should be less than 0:

$$\frac{d}{dt}\frac{(\tilde{\theta}-\theta)\cdot(\tilde{\theta}-\theta)}{2} = (\tilde{\theta}-\theta)\cdot\frac{d}{dt}(\tilde{\theta}-\theta) = (\tilde{\theta}-\theta)\cdot r < 0. \tag{C.2}$$

The second equality is due to a property of Hamiltonian system in physics. This condition is checked for each subtree and also for the whole tree. In addition, NUTS also stops expanding $\mathcal{B}$ when any newly discovered states in the continuing process has extremely low probability to be in $\mathcal{C}$. To formulate, NUTS develops the following condition based on (C.1):

$$\mathcal{L}(\theta) - \frac{1}{2}r\cdot r - \log u > -\Delta_{max}. \tag{C.3}$$

In NUTS, Δ_{max} is set to 1000, so the algorithm continues as long as the simulation is moderately accurate.

So far, we have addressed all aspects needed for deriving the simplified NUTS algorithm. We summarise the simplified NUTS algorithm below.

Algorithm 1 Sample for a point $\tilde{\theta}$ using a NUTS iteration

Require: The initial position θ_0^1.
  Resample $r \sim \mathcal{N}(0, I)$.
  Resample $u \sim \text{Unif}([0, \exp\{\mathcal{L}(\theta^t) - \frac{1}{2}r\cdot r\}])$.
  {j is the number of doubling steps we take to build a trajectory path.}
  {s is an indicator variable. It becomes 0 when a stopping criterion is met.}
  Initialise $j = 0$, $s = 1$, $\mathcal{C} = \{(\theta_0^1, r)\}$
  {Set the rightmost (+) and leftmost (-) points of the trajectory path.}
  Initialise $\theta^+ = \theta_0^1$, $\theta^- = \theta_0^1$.
  while s=1 **do**
    Build a binary tree of height j
    while Build a binary tree of height j **do**
      For each new node: check C.1 to determine whether or not to add the new node into $\mathcal{C}$.
      For each new node: check C.3. If Condition Three is met, set $s = 0$.
      For each subtree: check C.2. If Condition Two is met, set $s = 0$.
    end while
    Update θ^+ and θ^- to be the rightmost and leftmost point of the whole path.
    Check C.2 for the whole path.If it is met, set $s = 0$.
    j= j+1.
  end while
  Sample $\tilde{\theta}$ uniformly from $\mathcal{C}$.

3.2 Efficient NUTS

Hoffman and Gelman (2011) also proposed an efficient version of the NUTS algorithm. Firstly, the execution is halted exactly once a stopping criterion is met instead of at the end of the corresponding doubling step. Secondly, simplified NUTS requires to store 2^j states to perform uniform sampling at the end. A solution for reducing this memory requirement (from $O(2^j)$ to $O(j)$, see Hoffman and Gelman, 2011) is to use a

more sophisticated transition kernel and to exploit the binary tree structure of the trajectory path. We refer interested reader to **?** for details. The NUTS function included in the package is an efficient algorithm with these improvements implemented.

4 Dual averaging

In Hoffman and Gelman (2011) a method based on the primal-dual algorithm by Nesterov (2009) is also provided for setting the step size ϵ for both HMC and NUTS. In the context of MCMC, considered the statistic $H_t = \delta - \alpha_t$ where δ is a specified average acceptance probability and α_t is the observed Metropolis one at time t. The key idea of the dual averaging algorithm is that, under some specific conditions, $H_t = 0$ can be approached (that is, to approximate the desired average acceptance probability) by tuning $\log(\epsilon)$ using an iterative procedure. In the implementation, ϵ is tuned during the warmup phase (the user needs to specify the number of iterations of the warmup phase) and kept constant for the subsequent iterations. The user needs to specify the target mean acceptance rate δ and the number of iterations of the warmup phase.

5 NUTS Package and evaluations

We first introduce the NUTS Package and how we can use this package to generate samples from a posterior distribution, using HMC, modified versions of HMC and efficient NUTS. We will present results by sampling from a simple distribution, which is a 2-dimensional double exponential (Laplace) distribution with zero mean and unit scale parameter. The distiribution has the density

$$f(x|\mu, b) = \frac{1}{2b} \exp(-\frac{|x-\mu|}{b}),$$

with the parameter $\theta = (\mu, b) = (0, 1)$. We ran HMC and NUTS for 5000 iterations (M=5000) after 2500 burn-in iterations (Madapt = 2500).

```
>  L <- function(x) {
+    return(list(- sum(abs(x)), - sign(x)))
+  }
>  samples.hmc <- Hmc(theta0 = c(1,1), epsilon = 0.2, leap.nsteps = 10, L = L, M = 5000)
>  samples.hmcdual <- HmcDual(theta0=c(1,1), delta = 0.65, lambda = 1.5, L = L, M = 5000, Madapt = 2500)
>  samples.nuts <- NutsDual(theta0 = c(1,1), delta = 0.6, L = L, M = 5000, Madapt = 2500)
```

Depending on the method, each result is assigned with a class and its own print function. For example, for the samples generated by NUTS, we have

```
>   load("/homes/palma/Documents/NUTS/data/SAMPLE.RData")
>    #load("https://github.com/jodie0399/NUTS/DoubExp.RData")
>   print(samples.nuts)

Dimension of the parameter: 2
Sample size generated: M =  5000

theta 1 :  -1.055986 -0.788904  0.387217  0.387217  0.469840 -0.111665 -0.111665 -0.091766 ...
theta 2 :  -1.025440 -1.132669 -0.470981 -0.470981 -0.543014  1.444288  1.444288  1.616248 ...

Acceptance Rate:  1.000000 0.408421 0.312188 1.000000 0.204909 0.250144 1.000000 0.533262 ...

Values of epsilon bar, normalised by the final value of epsilon bar:
11.938494  4.248787  1.539326  0.592077  0.437687  0.454401  0.611903  0.503058 ...

Height of trajectory tree:  3 2 2 4 6 5 3 3 4 3 2 1 1 4 ...
```

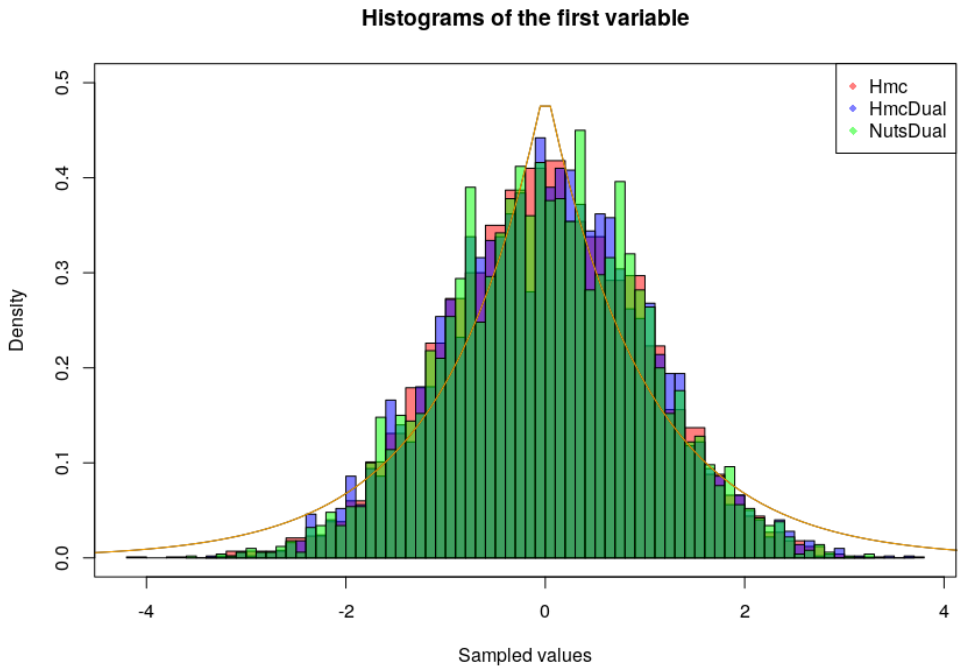

Figure 1: Evaluation of the functions for the Double Exponential example

To view the outputs from these three methods, we can generate for example the histogram of the observed samples for θ_1 in 1.

The histograms are substantially overlapping and they are consistent with the shape of the target distribution.

5.1 Evaluations: HMC

In this section, we will use a 30-dimensional multivariate normal distribution. Since HMC is invariant to rotation and NUTS is simply an extension of HMC, the performance of each method will be the same for any 30-dimensional normal distribution, having the same squre roots of the eigenvalues of their covariance matrix. Hence, it is sufficient for us to consider a distribution with independent components. In this project, we follow the example provided by Radford Neal's blog Neal (2012). Assume the target distribution is composed of 30 independent normal distributed variables $\theta_1, \theta_2, \ldots, \theta_30$, with standard deviations of 110, 100, 26 equally spaced values between 16 and 8, 1.1, 1.0 respectively. For our experiment, we used a burnin period of 2000 iterations and sampled 12500 points afterwards.

To evaluate the performances of these methods, we divide the 12500 sample points into 20 batches with 625 observations in each batch. We are interested in the variance of the mean and variances estimates of these batches. In terms of HMC methods, different values for leapfrog steps (100, 170, 200) are used.

The results for the standard HMC with dual averaging are reported in the following plots.

From the plot in 2, we see that for any leapfrog step value, the variance of the mean estimates changes drastically for different variable θ, ranging approximately from 0.01 to 100. This is rather unexpected. A similar evolution is observed for the variance of the variance estimates, although at a reduced scale. According to Neal (2012), this type of behaviour is caused by the fact that Hamiltonian dynamics is periordic for each variable. If the length of the period is similar to the trajectory length, the ending point will be close to the starting point, leading to poor sample quality. A possible solution to this issue is to introduce randomness when deciding the number of leapfrog steps. In the following experiment, we implemented Neal's proposal and chose a number of leapfrog step randomly from the uniform distribution between 0.9 and 1.1 times the original number of leapfrog steps. The results are reported in 3.

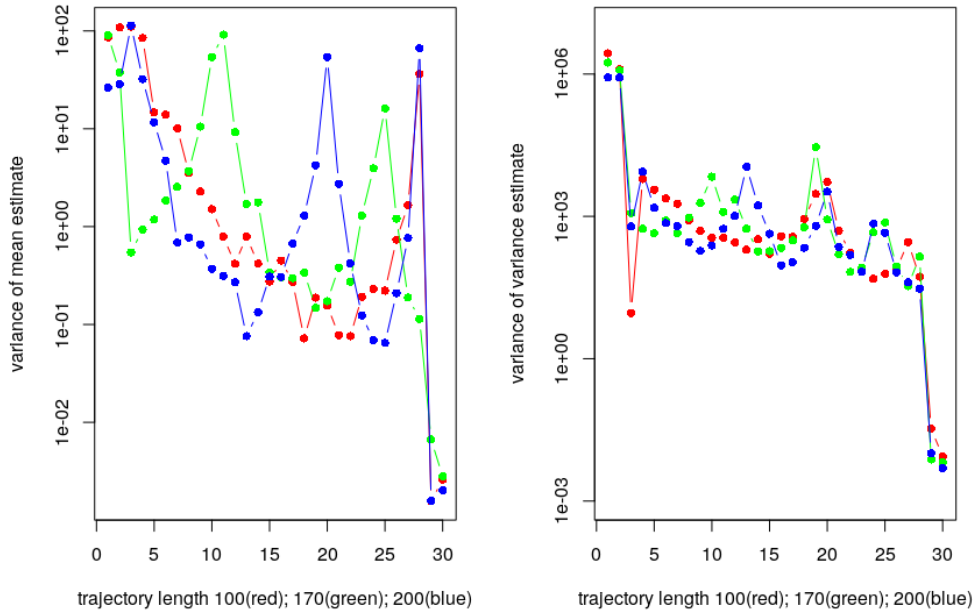

Figure 2: Results for HMC with Dual Averaging

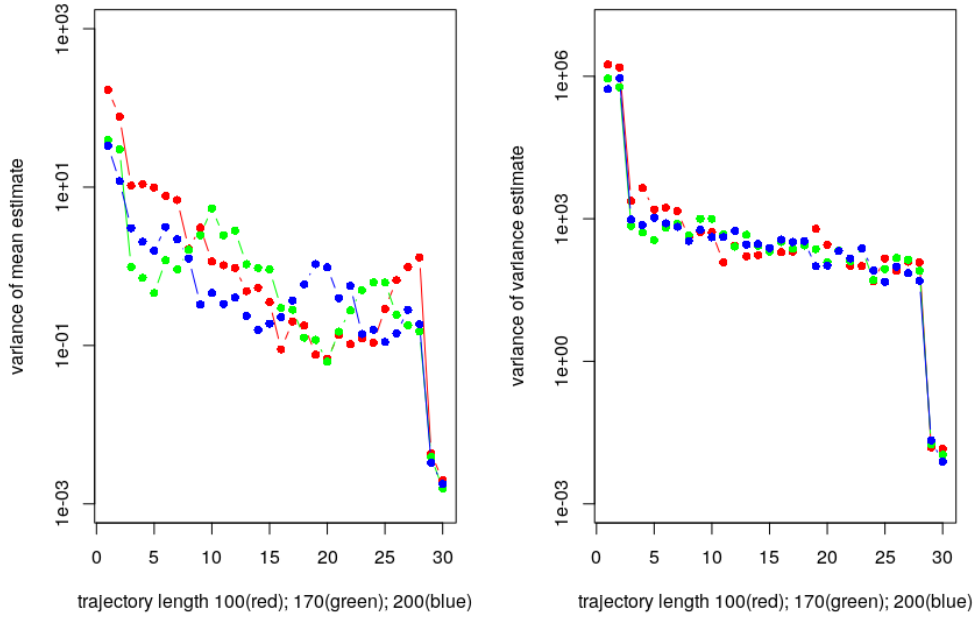

Figure 3: Results for HMC with Dual Averaging with modified leapfrog number of steps

After the modification, the issue caused by the periodic behaviour of Hamiltonial dynamics is partially addressed. The variance of means still exhibit a fluatuting pattern but at a lower magnitude. For this reason, we employ the modified HMC for the following experiments.

Another issue about HMC is that the performance of HMC is largely affected by the choice of the trajectory length λ, which we will discuss in detail when comparing it with NUTS.

5.2 Evaluations: HMC vs. NUTS

Evaluations of NUTS are conducted based on the approach proposed by Hoffman and Gelman (2011). The first experiment is aimed at assessing whether the implementation of dual averaging is effective. The criterion used is the discrepancy between the desired average acceptance rate δ and the mean of the acceptance rate α. For each targer distribution, we implemented NUTS and modified HMC with Dual Averaging to generate 2000 samples after a 1000 iteration burn-in period. The experiment is replicated for 15 values (NUTS) and 8 values (HMC) of δ equally spaced between 0.25 and 0.95. In terms of the trajectory length for HMC, we tried 4 different values, each of which is 1.5 times larger than the previous one. For each combination of δ and λ, 10 simulations with different random seeds have been produced. Assessments are done for both the double exponential distribution and the 30-dimensional normal distribution (introduced previously)

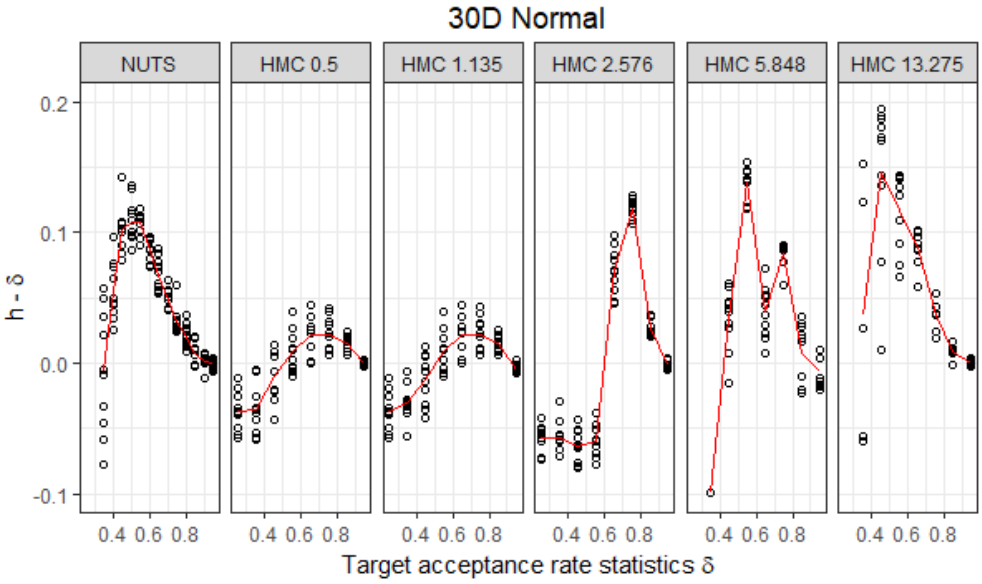

Figure 4: Evaluation of Dual Averaging performances for the normal case

The results shown in 5 and 5 partly confirms the findings in the original paper. For small values of λ in HMC, the average acceptance rate is close to the desired δ for all δ tested. When the trajectory length increases, the difference between the average acceptance rates in HMC and the specified δ increases as well. It is difficult to acheive the desired acceptance rate with relatetively large λ. For what concerns NUTS, the dual averaging performances are rather unsatisfactory. Instead of a line closing to 0, the line of difference fluctuates with majority points far from 0. In addition, we notice that the variance of the discrepancies are large for high value of λ, which might be caused by the fact that HMC is highly sensitive to the trajectory length.

Hoffman and Gelman (2011) also computed the effective sample size (ESS) to show the effectiveness of NUTS. However, the formula for ESS reported in the paper has been shown to be incorrect because it does not take into account negative autocorrelations which in HMC are quite possible. For this reason we used a standard ESS function from the package "mcmcse".

For what concerns the normal case, the ESS for HMC is extremely small for the first four trajectory length considered but an increasing trend of ESS can be observed when trajectory length increases. We expect high

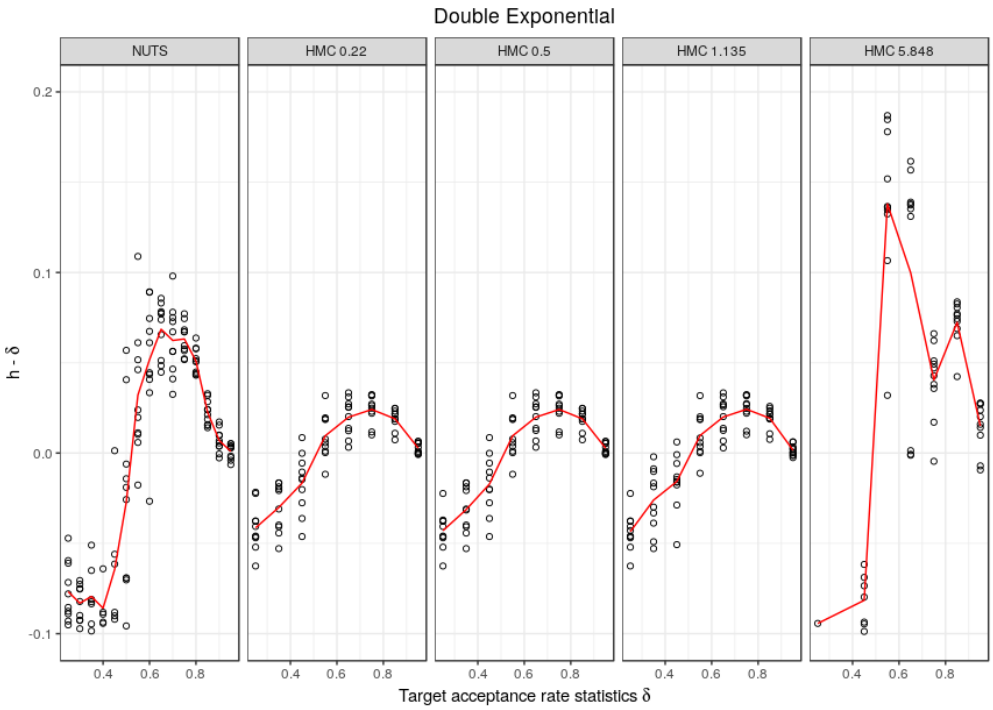

Figure 5: Evaluation of Dual Averaging performances for the double exponential case

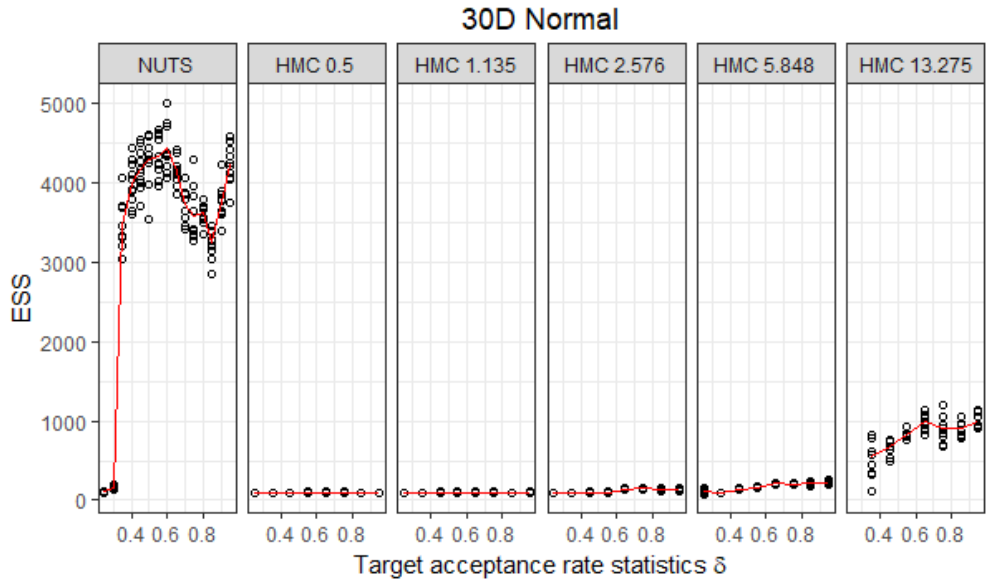

Figure 6: Evaluation of ESS performances for the normal case

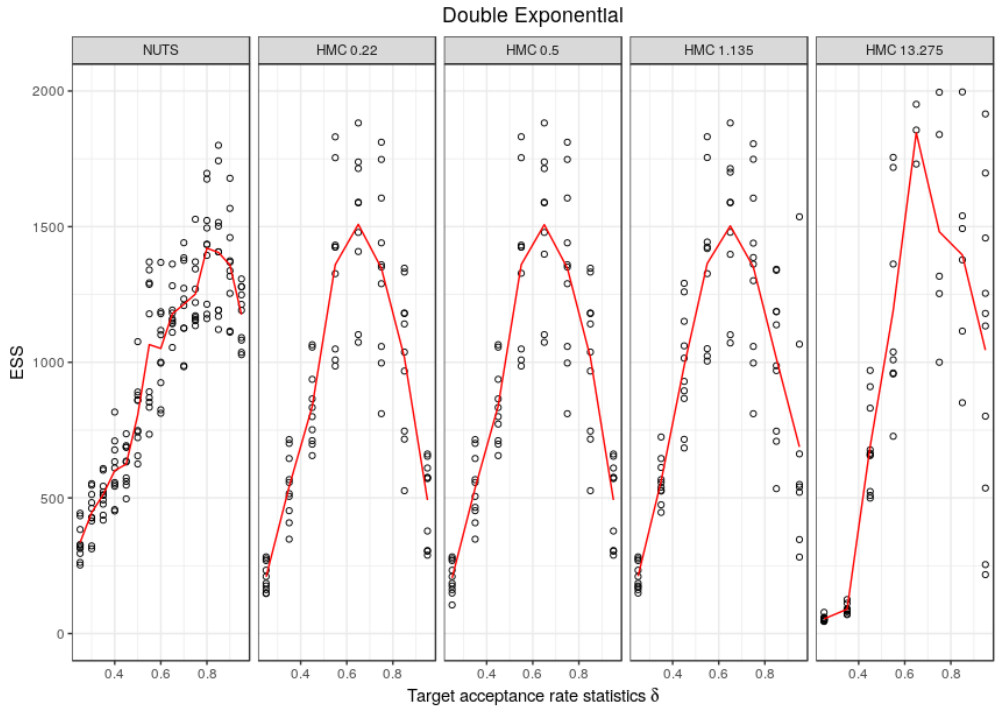

Figure 7: Evaluation of ESS performances for the double exponential case

trajectory length would be a better option for the 30-dimensional normal distribution. The ESS for NUTS also shows a peculiar pattern with ESS number increases again after $\delta > 0.8$. With these unexpected results, we propose further analysis to be made and more experiments with much larger trajectory lengths to be carried on. The double exponential scenario is instead close to our expectations. For HMC, our results confirmed the statement made by Hoffman and Gelman (2011) that HMC performs better at the optimal value $\delta = 0.65$. Moreover, from the simple visualization of ESS, NUTS performed similar to HMC but not better. Still, further studies should be done.

Finally, we conduct an investigation into the possible premature U-turn behaviour of NUTS. This defect of NUTS was first discovered and discussed in Neal (2012). In plain words, premature U-turn happens when some directions are much more constrained with respect to some other directions. Therefore, the trajectory of intermediate constrained direction reverses long before the least constrained direction is fully explored, which means NUTS stop long before achieving the optimal trajectory length. To study this behaviour, we consider again the 30-dimensinal normal distribution used in section 5.1. Results are shown in the following plots. An independent sample using built in function rnorm is also included. We see from 8 that the NUTS is not as efficient as HMC (by a factor of 2) in estimating variables with large standard deviation. On the contrary, NUTS outperforms HMC for variables with relatively small standard deviations. This inefficiency of NUTS in sampling variables of large standard deviation is consistent with the pre-mature U-turn behaviour suggested by Neal (2012). The better performance of NUTS for variables with small standard deviations may be due to its automatic tuning procedure.

6 Conclusion

In this report, we have discussed some features about the No-U-Turn Sampler in comparison with standard and modified versions of HMC. In general, some of the results presented in Hoffman and Gelman (2011) are not confirmed by our analysis, especially for what concerns the claimed advantages of NUTS over the other HMC algorithms. We acknowledged the limits of our studies and some experiments results should be revised. The study is inconclusive regarding the effectiveness of NUTS. Further investigations should be carried out.

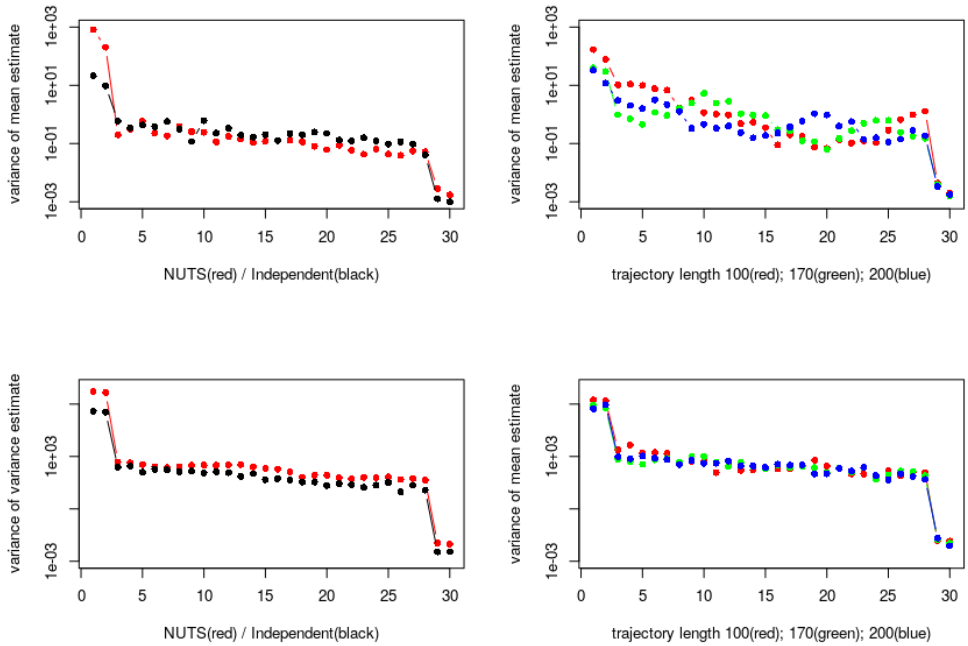

Figure 8: NUTS vs. independent case

References

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *arXiv preprint arXiv:1111.4246*, 2011.

Radford Neal. Evaluation of nuts — more comments on the paper by hoffman and gelman, 2012. URL `https://radfordneal.wordpress.com/2012/01/27/evaluation-of-nuts-more-comments-on-the-paper-by-hoffman-and-gelman/`.

Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1): 221–259, 2009.