Is the NUTS algorithm correct?

J.M. Sanz-Serna*

May 5, 2020

Abstract

This paper is devoted to investigate whether the popular No U-turn (NUTS) sampling algorithm is correct, i.e. whether the target probability distribution is exactly conserved by the algorithm. It turns out that one of the Gibbs substeps used in the algorithm cannot always be guaranteed to be correct.

1 Introduction

This paper is devoted to investigate whether the popular No U-turn (NUTS) sampling algorithm [10] is correct, i.e. whether the target probability distribution is exactly conserved by the algorithm. NUTS may be defined as a variant of the Hamiltonian (or Hybrid) Monte Carlo (HMC) Sampler [8, 11] where the length of the integration legs of the Hamiltonian dynamics [13] is adaptively chosen by the algorithm itself, thus freeing the user of one of the most delicate tasks in the practical application of HMC. In fact (see e.g. [4]) increasing the length of the integration legs increases the computational cost of the sampling and simultaneously may degrade the quality of the samples obtained. On the other hand short integration legs will obviously lead to strong correlations between successive samples.

NUTS is a very ingenious and involved algorithm that brings together a number of different ideas. For this reason it is not immediately obvious that it is correct. In earlier articles with different coworkers [1, 2, 3, 5, 6, 7, 9, 12], I have studied the impact of the numerical integration errors on HMC sampling and the present research started when studying the effect of combining advanced numerical integrators [5] and the ideas in NUTS. A salient feature of NUTS is that, unlike most sampling algorithms, it does not incorporate an accept/reject mechanism and I was intrigued by the following question: how is it possible for NUTS to be correct when the leapfrog integrator does not exactly preserve the Boltzmann-Gibbs distribution with density \( \exp(-H) \) in the phase space of position/momentum pairs (\( H \) denotes the Hamiltonian)?

This paper starts from an observation: for the special case of the univariate standard Gaussian distribution and some choices of step-lengths, the recursive procedure used by NUTS to determine the numerical trajectory stops with probability 1 after a single

*Departamento de Matemáticas, Universidad Carlos III de Madrid, Avenida de la Universidad 30, E-28911 Leganés (Madrid). Email: jmsanzserna@gmail.com
time-step (Section 2). It is then possible in those cases (Section 3) to rewrite NUTS in a very short way where things become more transparent. It turns out that one of the Gibbs substeps used by the algorithm cannot be guaranteed to be correct. The final Section 4 discusses the scope of our findings. For brevity, this article does not reproduce the description of NUTS provided in [10].

2 Preliminaries

We study the NUTS algorithm as applied to the model problem where the target is the standard univariate normal distribution with density $p \propto \exp(-\theta^2/2)$. If, as in [10], the kinetic energy is chosen to be $r^2/2$ ($r$ is the momentum variable), a single time-step $(\theta, r) \mapsto (\theta', r')$ of the leapfrog integrator is given by

$$
\theta' = \left(1 - \frac{\epsilon^2}{2}\right) \theta + v r, \quad r' = v \left(-\epsilon + \frac{\epsilon^3}{4}\right) \theta + \left(1 - \frac{\epsilon^2}{2}\right) r,
$$

where $\epsilon > 0$ is the step-length and $v = +1$ for a forward step and $v = -1$ for a backward step. It is important to emphasize that the leapfrog integrator is volume-preserving and time-reversible for all choices of $\epsilon$ [5]. At each step of the Markov chain, NUTS performs forward and backward time-steps to recursively build a binary tree. The construction of the tree is halted as soon as the algorithm detects that a U-turn has taken place and a segment of the numerically computed trajectory folds back. This means that

$$
(\theta' - \theta)(r^-) < 0 \quad \text{or} \quad (\theta' - \theta)(r^+) < 0,
$$

where $(\theta^-, r^-)$ and $(\theta^+, r^+)$ respectively denote the left-most and right-most point in the last computed sub-tree of the binary tree.

In this section we study whether, at a given step of the Markov chain, a U-turn takes place after building the very first tree with two leaves, i.e. after performing a single time-step from the current state of the chain.

We abbreviate as $(\theta, r)$ the current state $(\theta^{m-1}, r^{m-1})$, $m = 1, \ldots, M$, of the Markov chain and denote by $(\theta', r')$ the point obtained by performing a single time-step from the current state of the chain.

We first consider the case $v = 1$ (forward integration). The two-leaf tree then has $(\theta^-, r^-) = (\theta, r)$ and $(\theta^+, r^+) = (\theta', r')$ and the condition (2) becomes:

$$
(\theta' - \theta)r < 0 \quad \text{or} \quad (\theta' - \theta)r' < 0.
$$

Lemma 1 In the above situation:

1. Assume that $\epsilon = 2$, $r \neq (\epsilon/2)\theta$ and $r \neq 0$. Then the U-turn condition (2) holds.

2. Assume that $\epsilon > 2$ and $r \neq (\epsilon/2)\theta$. Then the U-turn condition (2) holds.

Proof: The integrator (1), the condition (2) and the conditions $r \neq (\epsilon/2)\theta$ and $r \neq 0$ are all left invariant when changing $(\theta, r)$ into $-(\theta, r)$, i.e. they are all preserved by the symmetry with respect to the origin of the $(\theta, r)$ plane. Therefore we may assume in the proof that $\theta \geq 0$. 

2
By using the first equation in (1) with \( v = 1 \), we note that the first condition in (3) holds if and only if \( (-\epsilon^2/2)\theta + \epsilon r < 0 \) or \( (-\epsilon^2/2 + \epsilon) \theta + \epsilon r < 0 \). If we introduce a system of polar coordinates \((\rho, \alpha)\), \(-\pi/2 \leq \alpha \leq \pi/2\), in the half-plane \( \theta \geq 0 \), the requirement becomes

\[
0 < \alpha < \arctan(\epsilon/2).
\] (4)

On the other hand, after using (1) with \( v = 1 \), the second condition in (3) may be written as

\[
\frac{\epsilon^3}{2} \left(1 - \frac{\epsilon^2}{4}\right) \theta^2 + \left(-\frac{3\epsilon^2}{2} + \frac{\epsilon^4}{2}\right) \theta r + \epsilon \left(1 - \frac{\epsilon^2}{2}\right) r^2 < 0.
\]

The left hand-side of this inequality vanishes on the lines

\[
\alpha = \arctan(\epsilon/2), \quad \alpha = \arctan(\epsilon \lambda), \quad \lambda = \frac{1 - \epsilon^2/4}{1 - \epsilon^2/2},
\]

and it is then easily checked that, for \( \epsilon \geq 2 \), the second condition in (3) holds if and only if

\[
-\pi/2 \leq \alpha < \arctan(\epsilon \lambda) \quad \text{or} \quad \arctan(\epsilon/2) < \alpha \leq \pi/2.
\] (5)

If \( \epsilon > 2 \), then \( \lambda > 0 \) and either (4) or (5) hold provided that \( \alpha \neq \arctan(\epsilon/2) \). For \( \epsilon = 2 \), we have \( \lambda = 0 \) and (3) takes place except if \( \alpha = 0 \) or \( \alpha = \arctan(\epsilon/2) \). □

Let us now study the case \( v = -1 \), where the two-leaf tree has \((\theta^-, r^-) = (\theta', r')\) and \((\theta^+, r^+) = (\theta, r)\) and the condition (2) reads:

\[
(\theta - \theta') r' < 0 \quad \text{or} \quad (\theta - \theta') r < 0.
\] (6)

Lemma 2 In the above situation:

1. Assume that \( \epsilon = 2 \), \( r \neq -(\epsilon/2)\theta \) and \( r \neq 0 \). Then the U-turn condition (6) holds.

2. Assume that \( \epsilon > 2 \) and \( r \neq -(\epsilon/2)\theta \). Then the U-turn condition (6) holds.

Proof: Since \((\theta, r) \mapsto (\theta', r')\) is a leapfrog step with \( v = -1 \), it follows that \((\theta, -r) \mapsto (\theta', -r')\) is a leapfrog step with \( v = 1 \). The condition (6) holds for the backward step if and only if the condition (5) holds for the forward step and we apply the preceding lemma. □

The lemmas clearly imply:

Proposition 3 Fix \( \epsilon \geq 2 \), \( v (v = 1 \text{ or } v = -1) \) and \( \theta \). If \( r \) is random with distribution \( \mathcal{N}(0, 1) \), then, with probability 1, a U-turn takes place after computing a single time-step from \((\theta, r)\).
Given $\theta^0$, $\epsilon > 0$, $M$:
Set $v = 1$.
for $m = 1$ to $M$ do
    Resample $r^{m-1} \sim \mathcal{N}(0, 1)$.
    Resample $u \sim \text{Uniform}\left([0, \exp\left\{-\frac{1}{2}(\theta^m)^2 - \frac{1}{2}(r^m)^2\right\}\right]$.
    Compute a leapfrog step $(\theta', r') \leftarrow (\theta^{m-1}, r^{m-1}, \epsilon, v)$.
    if $u \leq \exp\left\{-\frac{1}{2}(\theta')^2 - \frac{1}{2}(r')^2\right\}$ (slice condition holds) then
        $(\theta^m, r^m) = (\theta', r')$.
    else
        $(\theta^m, r^m) = (\theta^{m-1}, r^{m-1})$.
    end if
end for

Table 1: Algorithm E to generate samples $\theta^m$, $m = 1, \ldots, M$, from the standard normal distribution. When $\epsilon \geq 2$, it is equivalent to Efficient NUTS.

### 3 Analysis

In this section we analyze the application of the NUTS algorithm in [10] to the model target $\theta \sim \mathcal{N}(0, 1)$.

We begin with some observations that will be used later. The paper [10] gives pseudocode for two versions of the NUTS algorithm: Naive NUTS and Efficient NUTS. In both of them, after computing a point $(\theta', r')$ by means of the leapfrog integrator, one tests whether the condition

$$
\exp\left\{-\frac{1}{2}(\theta')^2 - \frac{1}{2}(r')^2\right\} \leq u \exp(-\Delta),
$$

holds where $u$ is the slice variable and $\Delta \gg 1$ is a user-chosen parameter. If the condition obtains, the integration is stopped in order to avoid wasting computational effort in regions of extremely low probability. In what follows and for simplicity in the exposition, we will sometimes assume that this test is suppressed from the algorithms (or equivalently that $\Delta$ is taken to be $\infty$).

Both Efficient NUTS and Naive NUTS include an outer for loop with an index $m = 1, \ldots, M$; each value of $m$ corresponds to a step in the Markov chain. Nested in this loop is a while loop that contains a block of code that successively builds binary trees of increasing height. As shown in Proposition 3 when the target is the standard Gaussian distribution and $\epsilon \geq 2$, for each value of $m$ both versions of NUTS will almost surely go only once through the while loop. It is then possible to rewrite the algorithms in a simplified way by omitting the inner loop.

We first consider the Efficient version of NUTS in [10]. The corresponding simplified algorithm, that we shall call Algorithm E, is given in Table 1. All integrations in Algorithm E are performed with $v = 1$; there is no point in choosing $v$ randomly, because the distribution of $r$ is symmetric and a forward leapfrog time-step with initial momentum $r$ achieves the same effect as a backward time-step with initial momentum $-r$. We also remark that $(\theta^m, r^m)$ is a deterministic function of $(\theta^{m-1}, r^{m-1})$: randomness only enters in Algorithm E through resampling the momentum and the slice
Figure 1: The top panels correspond to the leapfrog map \((\theta, r) \mapsto (\theta', r')\) when \(\epsilon = 2\). The circle \(C\) of radius 2 (a slice) is mapped into an ellipse \(E\) of the same area: \(m(E) = m(C)\). The ellipse is the disjoint union of two lobes \(L_1, L_2\) (that consists of points with \((\theta')^2 + (r')^2 > 4\), i.e. that violate the slide condition) and a central part \(D = E \setminus (L_1 \cup L_2)\) (where \((\theta')^2 + (r')^2 \leq 4\)). Each lobe \(L_j, j = 1, 2\), is the image by the leapfrog map of a set \(S_j\) that resembles a circular segment; \(D\) is then the image of \(R = C \setminus (S_1 \cup S_2)\). The sets \(R\) and \(D\) have the same area because leapfrog is area-preserving. The bottom panels correspond to the map \(F : (\theta, r) \mapsto (\theta^*, r^*)\) used in Algorithm E to find \((\theta^{m-1}, r^{m-1})\) as a (deterministic) function of \((\theta^{m-1}, r^{m-1})\). For points in \(R\) (slice condition satisfied), \(F\) coincides with the leapfrog mapping; for points in \(S_1 \cup S_2\), \((\theta^*, r^*) = (\theta, r)\). For the image \(F(C) = D \cup F(S_1) \cup F(S_2)\), we clearly have \(m(F(C)) < m(C)\). Here, \(m(D) + m(F(S_1)) + m(F(S_2)) = m(R) + m(S_1) + m(S_2) = m(C)\), but \(m(D) + m(F(S_1)) + m(F(S_2)) > m(F(C))\) because \(F(S_1)\) and \(F(S_2)\) have overlaps with \(D\); the area of the overlaps is the difference \(m(C) - m(F(C))\). The behaviour illustrated here is not specific to the circle of radius 2 and persists when the radius of the circle changes. Similarly, all values of \(\epsilon > 0\) lead to a qualitatively similar picture.
variable \( u \). To prove the following result it is sufficient to compare the flows of the algorithms involved:

**Proposition 4** Consider the application of Efficient NUTS to the univariate standard Gaussian distribution with \( \epsilon \geq 2 \) and \( \Delta = \infty \). Conditional on \( \theta^0 \), the distribution of the variables \( \theta^m, m = 1, \ldots, M \), coincides with the distribution of the variables \( \theta^m \) in Algorithm E.

In addition, conditional on \( \theta^0 \) and with probability 1, each realization \( \theta^1, \ldots, \theta^M \) generated by Efficient NUTS coincides with the realization \( \theta^1, \ldots, \theta^M \) generated by Algorithm E, provided that both algorithms use the same draws for the slice variable and, in addition, the draws of \( r^{m-1} \) in Algorithm E coincide with the products \( vr \), where \( v \) and \( r \) are respectively the NUTS draws for the momentum and the variable that indicates whether the integration proceeds forward or backward.

Clearly, the very simple Algorithm E has the structure of a slice/Gibbs sampler.

For each \( m \), the algorithm first resamples \( r^{m-1} \) from its marginal distribution \( N(0, 1) \) and then samples \( u \) from its distribution conditional on \( (\theta^{m-1}, r^{m-1}) \). Finally, the move \( (\theta^{m-1}, r^{m-1}) \mapsto (\theta', r') \) should ideally preserve the joint distribution of \( \theta \) and \( r \) conditional on \( u \). This conditional distribution is, of course, uniform in the circle (slice)

\[
C_u = \{ (\theta, r) : u \leq \exp(-\theta^2/2 - r^2/2) \}.
\]

In fact, if the numerical integration were exact, the (deterministic) move \( (\theta^{m-1}, r^{m-1}) \mapsto (\theta', r') \) would exactly preserve the uniform distribution in the slice because, first, the exact Hamiltonian flow preserves the value of the energy \( \theta^2/2 + r^2/2 \) and therefore maps \( C_u \) into itself and, second, the exact Hamiltonian flow preserves area. Unfortunately the leapfrog integrator does not conserve energy and therefore it is possible that \( (\theta', r') \notin C_u \) (i.e. the result of the integration violates the slice condition). Algorithm E remedies this drawback by stipulating that, if \( (\theta', r') \notin C_u \), then \( (\theta^m, r^m) = (\theta^{m-1}, r^{m-1}) \in C_u \). Therefore the deterministic map \( F \) used in the algorithm to refresh \( (\theta, r) \) maps \( C_u \) into itself. Unfortunately \( F \) does not preserve area for reasons explained in Figure [1]. The conclusion is that Algorithm E cannot be expected to preserve the target density \( \propto \exp(-\theta^2/2) \).

Before we proceed, we note that the standard recipe to overcome the difficulties stemming from the possibility of having \( (\theta', r') \notin C_u \) (i.e. the difficulties generated by energy errors in the integrator) would have been the introduction of a suitable accept/reject mechanism for the “proposal” \( (\theta', r') \), but Algorithm E (or for that matter Efficient NUTS) does not include such a mechanism.

We have applied the Efficient NUTS algorithm as described in [10], not Algorithm E!, to the simulation of the standard univariate Gaussian distribution with \( \epsilon = 4, \theta^0 = 0 \) and a number of samples \( M = 10,000,000 \); see Figure [2]. The algorithm incorporates the test \( [7] \) with \( \Delta = 1000 \) but it never encountered a point satisfying this low-probability condition. In agreement with Proposition [8] at each of the 10,000,000

---

1 It would of course be possible not to use the figure and instead to describe analytically all the regions involved and check that \( F \) contracts area, but there is no real point in doing so.

2 In fact, as seen in the figure, the trouble with this simulation was precisely the opposite: the algorithm kept away from the tails of the distribution.
Figure 2: Histogram of $M = 10,000,000$ samples obtained by applying the Efficient NUTS algorithm to the standard normal distribution with $\epsilon = 4$, $\Delta = 1000$, and $\theta^0 = 0$.

steps of the Markov chain, the algorithm detected a U-turn after performing a single leapfrog time-step and therefore never constructed a binary tree with more than two leaves. This implies that the analysis presented here for Algorithm N applies to the Efficient NUTS algorithm: one of the Gibbs substeps is not correct. In fact it is not necessary to perform any statistical test to see that the histogram looks quite different from that of a normal variable. In the samples obtained, the magnitude $|\theta|$ never came even close to exceed the value 3, when it is well known that a standard normal variable is $\geq 3$ with probability $\approx 0.3\%$.

Turning now to the Naive version of NUTS, Table 2 gives an Algorithm N that, for the standard normal target and $\epsilon \geq 2$, is equivalent to that version (with $\Delta = \infty$). “Equivalent” is employed here in the sense used in Proposition 4. In Naive NUTS, the next state $\theta^m$ of the Markov chain is chosen uniformly at random from all the leaves in the binary tree that satisfy the slide condition; this is to be compared with Efficient NUTS which biases the choice of $\theta^m$ trying to get away from $\theta^{m-1}$. Correspondingly, when $(\theta', r')$ satisfies the slide condition, Algorithm N chooses the next state as either $(\theta^m, r^m) = (\theta', r')$ or $(\theta^m, r^m) = (\theta^{m-1}, r^{m-1})$, with probability $1/2$ each, and Algorithm E sets $(\theta^m, r^m) = (\theta', r')$. Algorithm N refreshes $(\theta, r)$ with fixed $u$ by means of a random procedure, rather than deterministically as it is the case in Algorithm E.

When the tree has only two leaves, this means that the next state is either the current state or the state computed by a single leapfrog time-step (each with probability $1/2$), under the assumption that the latter satisfies the slice condition.
Given \( \theta^0, \epsilon > 0, M \):

Set \( v = 1 \).

for \( m = 1 \) to \( M \) do

- Resample \( r^{m-1} \sim \mathcal{N}(0, 1) \).
- Resample \( u \sim \text{Uniform}\left([0, \exp\left(-\frac{1}{2}(\theta^m)^2 - \frac{1}{2}(r^m)^2\right)]\right) \).
- Compute a leapfrog step \((\theta', r') \leftarrow (\theta^{m-1}, r^{m-1}, \epsilon, v)\).

if \( u \leq \exp\left(-\frac{1}{2}(\theta')^2 - \frac{1}{2}(r')^2\right) \) (\% slice condition holds) then

Set \((\theta^m, r^m) = (\theta', r')\) or \((\theta^m, r^m) = (\theta^{m-1}, r^{m-1})\) with probability \(1/2\) each.

else

\((\theta^m, r^m) = (\theta^{m-1}, r^{m-1})\).

end if

end for

Table 2: Algorithm N to generate samples \( \theta^m, m = 1, \ldots, M \), from the standard normal distribution. If \( \epsilon \geq 2 \), it is equivalent to Naive NUTS.

Referring to bottom panels in Figure 1, in the transition \((\theta, r) \mapsto (\theta^*, r^*)\) the set

\[ \{ (\theta^*, r^*) : (\theta^*)^2 + (r^*)^2 \leq 4 \} \setminus (D \cup \mathcal{F}(S_1) \cup \mathcal{F}(S_1)) \]

(the reentrant wedges in the right panel), that receives no probability mass in Algorithm E, in Algorithm N receives only 50% of the target probability mass. Algorithm N cannot be expected to preserve the target density \( \propto \exp(-\theta^2/2) \).

4 Discussion

When performing Hamiltonian Monte Carlo sampling, the widely popular NUTS algorithm frees the user from her most delicate task: the choice of the duration of the integration legs of the Hamiltonian dynamics. This is achieved by increasing at each step that duration until the algorithm detects the occurrence of a U-turn. We may say that NUTS makes matters much easier for the user at the price of complicating the inner workings of the algorithm, which in turn implies that NUTS is not overly transparent. We have proved that, in admittedly very special circumstances (univariate Gaussian target and step-size \( \epsilon \geq 2 \)), NUTS only goes once through its inner while loop and this makes the algorithm more amenable to analysis. In those circumstances, NUTS is really a slice sampler where unfortunately one of the sub-steps is not correct due to the numerical integration errors. This certainly implies that the arguments provided in [10] to show that NUTS is correct cannot be complete. In fact [10] only demands (Condition C.1) that the numerical simulation of the Hamiltonian dynamics be performed in such a way that volume is preserved and we have shown (see Figure 2) that there are instances of the use of NUTS where the target distribution is only preserved in an approximate way, even when the transformations involved preserved volume.

\[ \text{In addition [10] makes systematic use of the time-reversibility of the integrator; but this requirement is not explicitly incorporated to list of conditions C1-C4.} \]
Lest we miss the obvious: the limitation of the analysis in this paper to univariate Gaussian targets is not a problem. Certainly an algorithm cannot possibly be correct if is not so for all targets.

It is fair to ask if, still under the assumption of univariate Gaussian target, NUTS, that is not correct for $\epsilon \geq 2$, is correct when $\epsilon$ is chosen in a suitable interval $\epsilon \in (0, \epsilon_{\text{max}})$, $\epsilon_{\text{max}} \leq 2$. We do not think that to be the case:

1. First of all, no matter how small $\epsilon$, the algorithm may encounter states of the Markov chain where a U-turn takes place after a single time-step and, at those Markov transitions, the algorithm will be incorrect. In fact when one looks at the integrator after a single time-step nothing changes dramatically as $\epsilon$ crosses the value 2. A value of $\epsilon$ below 2 leads in the top panels in Figure 1 to the circle $C$ still being mapped into an ellipse $E$ with lobes $L_j$ where the slice condition is violated. The only difference is that, as $\epsilon$ decreases, the eccentricity of the ellipse becomes gradually less pronounced and the area of the lobes diminishes. So, regardless of the value of $\epsilon$, it is always true that, whenever a U-turn takes place after a single time-step, the update of the $(\theta, r)$ variables does not preserve their correct marginal distribution. In this connection we point out that when, for instance, $\epsilon = 1.9$, efficient NUTS performs an average of $\approx 1.14$ time-steps for each step of the Markov chain, which implies that at least in 85% Markov transitions the algorithm only takes a single time-step and coincides with Algorithm E.

2. Even if, with $\epsilon < 2$, the U-turn occurs after several time-steps, the algorithm will be in a situation similar to that in Figure 1. A circular slice $C$ is transformed by one step of the leapfrog map into a set $T_1$, by two steps into a set $T_2$, etc.; due to integration errors, all the sets $T_j$ may have lobes of points that violate the slice condition and the algorithm transfers the probability mass of the lobes back to subsets of the slice. In this transference, overlaps may occur resulting in the uniform distribution in the slice not being preserved.

Some readers will have noticed that the value $\epsilon = 2$ that establishes the threshold for NUTS to only build trees with two leaves is precisely the so-called stability limit [5] of the leapfrog algorithm. We do not attach any significance to this fact and believe it is a mere coincidence. Stability is only relevant [5] when many (or at least several) time-steps are taken; the difficulty identified in Figure 1 happens after one time-step. As pointed out above, when one looks at the integrator after a single time-step nothing changes dramatically as $\epsilon$ crosses the value 2.

To sum up, numerical integration errors invariably lead to position/momentum pairs that violate the slide condition. The way those pairs are dealt with in NUTS is not entirely satisfactory: excluding them from the list of candidates to be the next state of the chain impacts the conservation of volume that is necessary for the $(\theta, r)$-update to be correct. It is of course conceivable that one may devise an accept/reject rule that may be turn the algorithm into a correct one.

Acknowledgements. I was supported by project MTM2016-77660-P(AEI/FEDER, UE) MINECO (Spain). I am thankful to Celia Caballero for coding the algorithms I used.
References

[1] E. Akhmatskaya, M. Fernández-Pendás, T. Radivojević, and J. M. Sanz-Serna, Adaptive splitting integrators for enhancing sampling efficiency of modified Hamiltonian Monte Carlo methods in molecular simulation, *Langmuir* 33 (2017), 11530–11542.

[2] A. Beskos, N. Pillai, G. O. Roberts, J. M. Sanz-Serna, and A. M. Stuart, Optimal tuning of the hybrid Monte Carlo algorithm, *Bernoulli* 19 (2013), 1501–1534.

[3] S. Blanes, F. Casas, and J. M. Sanz-Serna, Numerical integrators for the hybrid Monte Carlo method, *SIAM Journal on Scientific Computing* 36 (2014), A1556–A1580.

[4] N. Bou-Rabee and J. M. Sanz-Serna, Randomized Hamiltonian Monte Carlo, *The Annals of Applied Probability* 27 (2017), 2159–2194.

[5] N. Bou-Rabee and J. M. Sanz-Serna, Geometric integrators and the Hamiltonian Monte Carlo method, *Acta Numerica* 27 (2018), 113–206.

[6] M. P. Calvo, D. Sanz-Alonso, and J. M. Sanz-Serna, HMC: avoiding rejections by not using leapfrog and some results on the acceptance rate, arXiv 1912.03253, submitted.

[7] C. M. Campos and J. M. Sanz-Serna, Palindromic 3-stage splitting integrators, a roadmap, *Journal of Computational Physics* 346 (2017), 340–355. 2017.

[8] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, Hybrid Monte Carlo, *Physics letters B* 195 (1987), 216–222.

[9] M. Fernández-Pendás, E. Akhmatskaya, and J. M. Sanz-Serna. Adaptive multi-stage integrators for optimal energy conservation in molecular simulations, *Journal of Computational Physics* 327 (2016), 434–449. 2016.

[10] M. D. Hoffman and A. Gelman, The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo, *Journal of Machine Learning Research* 15 (2014), 1593–1623.

[11] R. M. Neal, MCMC using Hamiltonian dynamics, *Handbook of Markov chain Monte Carlo* 2 (11,) 2011.

[12] T. Radivojević, M. Fernández-Pendás, J. M. Sanz-Serna, and E. Akhmatskaya, Multi-stage splitting integrators for sampling with modified Hamiltonian Monte Carlo methods, *Journal of Computational Physics* 373 (2018), 900–916.

[13] J. M. Sanz-Serna and M. P. Calvo. *Numerical Hamiltonian Problems*. Dover Publications, 2018.