The No-U-Turn Sampler as a Proposal Distribution in a Sequential Monte Carlo Sampler without Accept/Reject

Lee Devlin, Matthew Carter, Paul Horridge, Peter L. Green, and Simon Maskell

***Abstract*—Markov Chain Monte Carlo (MCMC) is a method for drawing samples from non-standard probability distributions. Hamiltonian Monte Carlo (HMC) is a popular variant of MCMC that uses gradient information to explore the target distribution. The Sequential Monte Carlo (SMC) sampler is an alternative sampling method which, unlike MCMC, can readily utilise parallel computing architectures. It is typical within SMC literature to target a tempered distribution using a proposal with an accept/reject mechanism. In this letter, we show how the proposal used in the No-U-Turn Sampler (NUTS), an advanced variant of HMC, can be incorporated into an SMC sampler without an accept/reject mechanism. Empirical results show that this can remove the need for tempering and gives rise to accurate estimates being generated in fewer iterations which motivates this technique being deployed on parallel hardware.**

***Index Terms*—Bayesian inference, Sequential Monte Carlo**

I. Introduction

Markov Chain Monte Carlo (MCMC) is a tool, often used in Bayesian inference to draw samples $\mathbf{x} \in \mathbb{R}^D$ from a D-dimensional probability distribution $\pi(\mathbf{x})$. In MCMC, samples are drawn by moving from a state $\mathbf{x}_{k-1}$ at iteration $k-1$ to a state $\mathbf{x}_k$ at iteration k, with some acceptance probability such that the Markov chain is ergodic (i.e., converges to a stationary distribution), and detailed balance is maintained. This is such that the stationary distribution of the Markov chain is equal to the target distribution. Gradient based methods such as Metropolis-Adjusted Langevin (MALA) [1] and Hamiltonian Monte Carlo (HMC) [2] are variants of MCMC which have grown in popularity due to their ability to efficiently explore continuous state spaces. HMC introduces a momentum vector $\mathbf{p} \in \mathbb{R}^D$ to facilitate the exploration of states via the numerical integration of Hamiltonian dynamics. The No-U-Turn Sampler (NUTS), first proposed in [3], auto-calibrates parameters of the HMC process by stopping a trajectory once the path begins to turn back on itself. As a result of its applicability and efficient operation across a range of specific distributions, NUTS is used by probabilistic programming languages such as Stan [4], PyMC3 [5], and NumPyro [6].

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible. Work funded by the Engineering and Physical Sciences Research Council, as part of the grant 'Big Hypotheses: a Fully Parallelised Bayesian Inference Solution' (EP/R018537/1)

Lee Devlin, Matthew Carter, Paul Horridge and Simon Maskell are with the Department of Electrical Engineering and Electronics, and Peter Green is with the Department of Mechanical Engineering, University of Liverpool, Liverpool L69 3GJ, U.K. (e-mail: Lee.Devlin@liv.ac.uk; M.J.Carter2@liv.ac.uk; Paul.Horridge@liv.ac.uk; PLGreen@liv.ac.uk; S.Maskell@liv.ac.uk).

Sequential Monte Carlo (SMC) samplers, first introduced in [7], provide a way of realising estimates with respect to $\pi(\mathbf{x})$ based on a population of N weighted hypotheses (often referred to as samples or particles) which evolve over k iterations and are moved by means of a forward proposal distribution. In the SMC sampler literature, a Gaussian random-walk kernel is typically used, but this is not a requirement. In this letter we show how the proposal in NUTS can be used instead. This approach differs from other methods which incorporate Hamiltonian proposals in SMC samplers, e.g. [8], by not using accept/reject within the SMC sampler's proposal, therefore allowing more freedom regarding the choice on the L-kernel, a key parameter in the context of SMC samplers. A similar approach is presented in [9] using Langevin dynamics, but our focus is different as we are interested in obtaining good estimates in fewer iterations. We also note that HMC has been incorporated into an importance sampling scheme [10] which is also different to what we present here.

The rest of this letter is structured as follows. In Section II we present how SMC samplers operate and in Section III we show how the proposal for NUTS can be used as the proposal distribution in an SMC sampler without the use of accept/reject. Section IV presents results in the context of two examples. Section V concludes the paper.

II. Sequential Monte Carlo Samplers

In this letter we consider an SMC sampler that at the k^{th} iteration does not target $\pi(\mathbf{x})$ directly, but rather does so over k iterations such that the joint distribution $\pi_{1\ldots k}(\mathbf{x}_{1\ldots k})$ of all previous states is the target:

$$\pi_{1\ldots k}(\mathbf{x}_{1\ldots k}) = \pi_k(\mathbf{x}_k) \prod_{k'=2}^{k} L\left(\mathbf{x}_{k'-1}|\mathbf{x}_{k'}\right), \tag{1}$$

where $L\left(\mathbf{x}_{k'-1}|\mathbf{x}_{k'}\right)$ is the L-kernel, sometimes called the 'backwards' kernel. We define the forwards proposal as:

$$q(\mathbf{x}_{1\ldots k}) = q(\mathbf{x}_1) \prod_{k'=2}^{k} q\left(\mathbf{x}_{k'}|\mathbf{x}_{k'-1}\right). \tag{2}$$

Using importance sampling, where we take the ratio of (1) and (2), we attribute an incremental weight to the i^{th} hypothesis at iteration k, w_k^i, which is updated from the previous iteration's weight w_{k-1}^i via:

$$w_k^i = w_{k-1}^i \frac{\pi_k(\mathbf{x}_k^i)}{\pi_{k-1}(\mathbf{x}_{k-1}^i)} \frac{L(\mathbf{x}_{k-1}^i|\mathbf{x}_k^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}, \tag{3}$$

where $\pi_k(\mathbf{x}_k^i)$ and $\pi_{k-1}(\mathbf{x}_{k-1}^i)$ are instances of the k^{th} target distributions at the i^{th} sample's new and previous state, respectively. Initial weights are evaluated as the ratio of the target of the samples at iteration 1, i.e. $\pi_{k=1}(x_{k=1}^i)$ divided by the proposal of the initial samples $q(x_{k=1}^i)$.

When degeneracy occurs, where a small subset of samples have relatively high importance weights, a new set of samples are selected from the current set with probability proportional to the normalised weight, $\widetilde{w}$, which are calculated by dividing each weight by the sum of all weights such they sum to 1. This process of generating a new set of samples is called resampling. This involves selecting elements, with replacement, from $[\mathbf{x}_k^1 \dots \mathbf{x}_k^N]$ with probability $[\widetilde{w}_k^1 \dots \widetilde{w}_k^N]$ into a new vector $\mathbf{x}_k^{\text{new}}$ which then overwrites the old samples, i.e. $\mathbf{x}_k \leftarrow \mathbf{x}_k^{\text{new}}$, before the weights of the new samples are all set to $1/N$. Resampling is typically set to occur when the effective number of samples falls below some threshold value. Values of interest, e.g. expectation with respect to the target distribution, can be realised using the normalised sample weights.

A. Tempering

It is typical in SMC to employ tempering [11] such that a monotonically increasing scalar value $\phi_k \in [0, 1]$ at iteration k is introduced where notationally we write the tempered target at iteration k as $\pi_k^{\phi_k}(\mathbf{x})$. In Bayesian inference problems where we wish to evaluate a set of parameters for a model given a set of data $\mathbf{y}$, the target is a posterior distribution and, when using tempering, we sample from a sequence of distributions: $\pi_k^{\phi_k}(\mathbf{x}) \propto p(\mathbf{x})p(\mathbf{y}|\mathbf{x})^{\phi_k}$, where ϕ_k is in the sequence $(0 = \phi_0 < \phi_1 < \ldots < \phi_k \leq 1)$.

B. Choices of L-kernel

The L-kernel can take the form of any valid probability distribution. However, the optimal L-kernel, which minimises the variance of the sample estimates [7] is:

$$L^{\text{OPT}}(\mathbf{x}_{k-1}|\mathbf{x}_k) = \frac{q(\mathbf{x}_k|\mathbf{x}_{k-1})\eta_{k-1}(\mathbf{x}_{k-1})}{\int q(\mathbf{x}_k|\mathbf{x})\eta_{k-1}(\mathbf{x})d\mathbf{x}} \tag{4}$$

where η_k is the distribution of samples at iteration k and the denominator will be $\int q(\mathbf{x}_k|\mathbf{x})\eta_{k-1}(\mathbf{x})d\mathbf{x} = \eta_k(\mathbf{x}_k)$. Equation (4) is often intractable.

One approach to using the optimal L-kernel is to carefully select the proposal distribution such that the optimal L-kernel is analytically tractable [12]. However, this is problem specific. Our motivation in this paper is to define a general-purpose proposal distribution and L-kernel within an SMC sampler that is agnostic to the target distribution of interest.

C. Accept/Reject in Sequential Monte Carlo Samplers

Within the SMC literature, it is a common to use a Metropolis-Hastings (MH) proposal $q_{\text{MH}}(\mathbf{x}_k|\mathbf{x}_{k-1})$, which includes accept/reject and has a stationary distribution:

$$\pi_k^{\phi_k}(\mathbf{x}_k) = \int q_{\text{MH}}(\mathbf{x}_k|\mathbf{x}_{k-1})\pi_k^{\phi_k}(\mathbf{x}_{k-1})d\mathbf{x}_{k-1}. \tag{5}$$

If the acceptance rate is $A(\mathbf{x}_k|\mathbf{x}_{k-1})$, then:

$$q_{\text{MH}}(\mathbf{x}_k|\mathbf{x}_{k-1}) = \begin{cases} q_{\text{MH}=}(\mathbf{x}_k|\mathbf{x}_{k-1}) & \mathbf{x}_k = \mathbf{x}_{k-1} \\ q_{\text{MH}\neq}(\mathbf{x}_k|\mathbf{x}_{k-1}) & \mathbf{x}_k \neq \mathbf{x}_{k-1} \end{cases} \tag{6}$$

where

$$\begin{aligned} q_{\text{MH}=}(\mathbf{x}_k|\mathbf{x}_{k-1}) = &\int (1 - A(\mathbf{x}_k|\mathbf{x}_{k-1}))q(\mathbf{x}_k|\mathbf{x}_{k-1})d\mathbf{x}_k \\ &+ q(\mathbf{x}_k = \mathbf{x}_{k-1}|\mathbf{x}_{k-1}) \end{aligned} \tag{7}$$

which we note includes an intractable integral and where

$$q_{\text{MH}\neq}(\mathbf{x}_k|\mathbf{x}_{k-1}) = A(\mathbf{x}_k|\mathbf{x}_{k-1})q(\mathbf{x}_k|\mathbf{x}_{k-1}). \tag{8}$$

We can capitalise on (5) to choose the L-kernel:

$$\begin{aligned} L(\mathbf{x}_{k-1}|\mathbf{x}_k) =& \frac{q_{\text{MH}}(\mathbf{x}_k|\mathbf{x}_{k-1})\pi_k^{\phi_k}(\mathbf{x}_{k-1})}{\int q_{\text{MH}}(\mathbf{x}_k|\mathbf{x})\pi_k^{\phi_k}(\mathbf{x})d\mathbf{x}} && (9) \\ =& \frac{q_{\text{MH}}(\mathbf{x}_k|\mathbf{x}_{k-1})\pi_k^{\phi_k}(\mathbf{x}_{k-1})}{\pi_k^{\phi_k}(\mathbf{x}_k)}. && (10) \end{aligned}$$

Given the similarity between (4) and (10) and that as $k \to \infty$, $\eta_k(\mathbf{x}) \to \pi_k(\mathbf{x})$, this L-kernel is considered to be 'asymptotically optimal'. The weight update for this combination of proposal and asymptotically optimal L-kernel is:

$$\begin{aligned} w_k^i &= w_{k-1}^i \frac{\pi_k^{\phi_k}(\mathbf{x}_k^i)}{\pi_{k-1}^{\phi_{k-1}}(\mathbf{x}_{k-1}^i)} \frac{L(\mathbf{x}_{k-1}^i|\mathbf{x}_k^i)}{q_{\text{MH}}(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)} && (11) \\ &= w_{k-1}^i \frac{\pi_k^{\phi_k}(\mathbf{x}_{k-1}^i)}{\pi_{k-1}^{\phi_{k-1}}(\mathbf{x}_{k-1}^i)} && (12) \end{aligned}$$

which, thanks to the choice of L-kernel, involves no intractable integrals.

III. Sequential Monte Carlo samplers without accept/reject

In this section we consider a novel formulation of HMC/NUTS within an SMC sampler without accept/reject. When used in MCMC, NUTS generates a sequence of samples from the target-space in both position $\mathbf{x}$ and momentum $\mathbf{p}$ from a proposal of the form $q(\mathbf{x}_k, \mathbf{p}_k|\mathbf{x}_{k-1}, \mathbf{p}_{k-1})$. In an SMC sampler, to calculate (3), we wish to consider a proposal of the form $q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)$. We can address this disparity by considering the numerical integration of the Hamiltonian dynamics to be a non-linear function which deterministically transforms the old position using a randomly sampled momentum.

HMC and NUTS use a numerical method, Leapfrog, to simulate Hamiltonian dynamics and explore the target space. Leapfrog has several useful qualities. Firstly it is symplectic, i.e. it preserves the geometric structure of the phase space $\{\mathbf{x}, \mathbf{p}\}$, and therefore generates states with high acceptance probability for sufficiently small step-sizes. Secondly, it is both reversible and time symmetric such that the resulting proposal satisfies the detailed balance condition. The Leapfrog method over one step of step-size h is as follows:

$$\mathbf{p}_{k-\frac{1}{2}} = \mathbf{p}_{k-1} - \frac{h}{2} \left.\frac{\partial U}{\partial \mathbf{x}}\right|_{\mathbf{x}_{k-1}} \tag{13}$$

$$\mathbf{x}_k = \mathbf{x}_{k-1} + h\boldsymbol{M}^{-1}\mathbf{p}_{k-\frac{1}{2}} \tag{14}$$

$$\mathbf{p}_k = \mathbf{p}_{k-\frac{1}{2}} - \frac{h}{2} \left.\frac{\partial U}{\partial \mathbf{x}}\right|_{\mathbf{x}_k} \tag{15}$$

where U is a potential energy function and equal to $U(\mathbf{x}) = -\log(\pi(\mathbf{x}))$, and $\boldsymbol{M} \in \mathbb{R}^{D\times D}$ is a diagonal mass matrix that can be tuned.

A. Non-linear Transform of the Proposal Distribution

We wish to evaluate the probability that a random variable $\mathbf{X}_{k-1}$ transforms to a random variable $\mathbf{X}_k$ using a Hamiltonian based proposal, and vice-versa. We write this as $q(\mathbf{X}_k = \mathbf{x}_k|\mathbf{X}_{k-1} = \mathbf{x}_{k-1})$ for the forwards kernel and $L(\mathbf{X}_{k-1} = \mathbf{x}_{k-1}|\mathbf{X}_k = \mathbf{x}_k)$ for the L-kernel. First we derive an expression for the forwards kernel. We consider Leapfrog to be a single function $f_{\mathrm{LF}}(.)$ which transforms a state $\mathbf{x}_{k-1}$ (and momentum $\mathbf{p}_{k-1}$) to $\mathbf{x}_k$, i.e. $\mathbf{x}_k = f_{\mathrm{LF}}(\mathbf{x}_{k-1}, \mathbf{p}_{k-1})$. We can rewrite the forward kernel as:

$$\begin{aligned} &q(\mathbf{X}_k = \mathbf{x}_k|\mathbf{X}_{k-1} = \mathbf{x}_{k-1}) = \\ &q(\mathbf{X}_k = f_{\mathrm{LF}}(\mathbf{x}_{k-1}, \mathbf{p}_{k-1})|\mathbf{X}_{k-1} = \mathbf{x}_{k-1}). \end{aligned} \tag{16}$$

We use a transform of variables to write (16) as:

$$\begin{aligned} &q(\mathbf{X}_k = \mathbf{x}_k|\mathbf{X}_{k-1} = \mathbf{x}_{k-1}) = \\ &q(\mathbf{P}_{k-1} = \mathbf{p}_{k-1}|\mathbf{X}_{k-1} = \mathbf{x}_{k-1}) \left|\frac{\partial f_{\mathrm{LF}}(\mathbf{x}_{k-1}, \mathbf{p}_{k-1})}{\partial \mathbf{p}_{k-1}}\right|^{-1}. \end{aligned} \tag{17}$$

The initial momentum is typically sampled from a normal distribution $\mathbf{p}_k \sim \mathcal{N}(0, \boldsymbol{M})$. It follows that:

$$\begin{aligned} &q(\mathbf{X}_k = \mathbf{x}_k|\mathbf{X}_{k-1} = \mathbf{x}_{k-1}) = \\ &\mathcal{N}(\mathbf{p}_{k-1}; 0, \boldsymbol{M}) \left|\frac{df_{\mathrm{LF}}(\mathbf{x}_{k-1}, \mathbf{p}_{k-1})}{d\mathbf{p}_{k-1}}\right|^{-1}. \end{aligned} \tag{18}$$

Turning our attention to the L-kernel, we utilise the fact that Leapfrog is a reversible integration method, i.e. if we start at a state $\{\mathbf{x}_{k-1}, \mathbf{p}_{k-1}\}$ and then transform this to $\{\mathbf{x}_k, \mathbf{p}_k\}$ then by reversing the momentum $\mathbf{x}_{k-1} = f_{\mathrm{LF}}(\mathbf{x}_k, -\mathbf{p}_k)$. Following the steps in (16) and (17), except this time starting at $\mathbf{x}_k$ and with a momentum $-\mathbf{p}_k$, we find that:

$$\begin{aligned} &L(\mathbf{X}_{k-1} = f_{\mathrm{LF}}(\mathbf{x}_k, -\mathbf{p}_k)|\mathbf{X}_k = \mathbf{x}_k) = \\ &L(\mathbf{P}_k = -\mathbf{p}_k|\mathbf{X}_k = \mathbf{x}_k) \left|\frac{df_{\mathrm{LF}}(\mathbf{x}_k, -\mathbf{p}_k)}{d\mathbf{p}_k}\right|^{-1}. \end{aligned} \tag{19}$$

For each sample in an SMC iteration we need to calculate the ratio of (19) and (18) to assign an incremental weight (3). Writing the updated state in terms of the initial state and momentum we find that (for a diagonal mass matrix):

$$\left|\frac{\partial f_{\mathrm{LF}}(\mathbf{x}_{k-1}, \mathbf{p}_{k-1})}{\partial \mathbf{p}_{k-1}}\right| = h^D \prod_{i=1}^{D} \boldsymbol{M}_{ii}^{-1}. \tag{20}$$

As Leapfrog is a reversible method, if the momentum is reversed and the step-size is equal to that used in the forwards case we similarly find the reverse determinant is equal to (20) such that the determinants will cancel when calculating (3) regardless of the total number of Leapfrog steps taken.

B. Proposals Without the Asymptotic Approximate L-kernel

While it is common to use the asymptotically optimal L-kernel (12), this is not a requirement. One alternative approach is to assume the reverse of the forward proposal, i.e. $L(\mathbf{x}_{k-1}^i|\mathbf{x}_k^i) = q(\mathbf{x}_{k-1}^i|\mathbf{x}_k^i)$. In this instance we may define:

$$L(\mathbf{P}_k = -\mathbf{p}_k|\mathbf{X}_k = \mathbf{x}_k) = \mathcal{N}(-\mathbf{p}_k; \mathbf{0}, \boldsymbol{M}). \tag{21}$$

This is a sub-optimal strategy but can result in a high effective sample size since the underlying HMC process used to generate the samples is able to generate samples that closely approximate samples drawn from $\pi(\mathbf{x})$.

From (4) it can be seen that $L^{\mathrm{OPT}}(\mathbf{x}_{k-1}|\mathbf{x}_k) \propto q(\mathbf{x}_{k-1}, \mathbf{x}_k)$. An alternative approach is to approximate the optimal L-kernel analytically [13] where we model the joint density by fitting a Gaussian mixture model. For the Hamiltonian case we use the same transformation as the previous section and $L^{\mathrm{OPT}}(\mathbf{x}_{k-1}|\mathbf{x}_k) \propto q(-\mathbf{p}_{k-1}, \mathbf{x}_k)$, where again the determinant term that will cancel with the numerator in (3). The joint density is then approximated by:

$$\begin{aligned} &q(\mathbf{P}_k = -\mathbf{p}_k, \mathbf{X}_k = \mathbf{x}_k) \approx \\ &\mathcal{N}\left(\begin{bmatrix}-\mathbf{p}_k \\ \mathbf{x}_k\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_{-\mathbf{p}_k} \\ \boldsymbol{\mu}_{\mathbf{x}_k}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{-\mathbf{p}_k,-\mathbf{p}_k} & \boldsymbol{\Sigma}_{-\mathbf{p}_k,\mathbf{x}_k} \\ \boldsymbol{\Sigma}_{\mathbf{x}_k,-\mathbf{p}_k} & \boldsymbol{\Sigma}_{\mathbf{x}_k,\mathbf{x}_k}\end{bmatrix}\right), \end{aligned} \tag{22}$$

where $\boldsymbol{\mu} \in \mathbb{R}^D$ are mean vectors, and $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times D}$ are block covariance matrices. We then use the properties of Gaussians to define:

$$L^{\mathrm{OPT}}(\mathbf{P}_k = -\mathbf{p}_k|\mathbf{X}_k = \mathbf{x}_k) \approx \mathcal{N}(-\mathbf{p}_k; \boldsymbol{\mu}_{-\mathbf{p}_k|\mathbf{x}_k}, \boldsymbol{\Sigma}_{-\mathbf{p}_k|\mathbf{x}_k}), \tag{23}$$

where:

$$\boldsymbol{\mu}_{-\mathbf{p}_k|\mathbf{x}_k} = \boldsymbol{\mu}_{-\mathbf{p}_k} + \boldsymbol{\Sigma}_{-\mathbf{p}_k,\mathbf{x}_k}\boldsymbol{\Sigma}_{\mathbf{x}_k,\mathbf{x}_k}^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_{\mathbf{x}_k}) \tag{24}$$

and

$$\boldsymbol{\Sigma}_{-\mathbf{p}_k|\mathbf{x}_k} = \boldsymbol{\Sigma}_{-\mathbf{p}_k,-\mathbf{p}_k} - \boldsymbol{\Sigma}_{-\mathbf{p}_k,\mathbf{x}_k}\boldsymbol{\Sigma}_{\mathbf{x}_k,\mathbf{x}_k}^{-1}\boldsymbol{\Sigma}_{\mathbf{x}_k,-\mathbf{p}_k}. \tag{25}$$

Algorithm 1 shows how (23) is used in an SMC sampler using NUTS for N samples over a total of K iterations. A slice sampler (Algorithm 3 in [3]) can be used to generate new samples for the NUTS step. For the sub-optimal L-kernel (21), steps 8 and 9 are replaced by (21). For the resampling step, several parallelized methods may be employed (e.g. see [15] and [16]) to speed-up the computation time.

IV. Results

We now demonstrate our approach using both (21) and (23) compared with using an accept/reject mechanism with tempering and the weight update strategy given in (12). We note that NUTS with a slice sampler (as described in [3]) does not have an explicit accept/reject step. We have therefore added one using the accept/reject mechanism for Hamiltonian proposals from [14]. For the temperature schedule we use Alg. 2 of [8]. We note this utilizes a bisection method that will effect the run-time. We therefore elect to compare methodologies as a function of iteration. These problems were selected to be cases where in the first instance, the joint density is well-described by a Gaussian, and in the second instance, it is not. In all examples the mass matrix is set equal to the identity matrix: Adapting the hyperparameters of the HMC process, as done in [8], is left as the subject of future work.

This article has been accepted for publication in IEEE Signal Processing Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LSP.2024.3386494

Algorithm 1 SMC using NUTS as a proposal distribution with a Gaussian approximation to the optimal L-kernel for K iterations and N samples.

for i=1…N **do**
  Sample $\mathbf{x}_1^i$ from $q(\mathbf{x}^i)$
  Set initial weights to $w_1^i = \frac{\pi(\mathbf{x}_1^i)}{q(\mathbf{x}_1^i)}$
for $k = 2$ to K **do**
  for i=1…N **do**
    Sample an initial momentum vector $\mathbf{p}^i \sim \mathcal{N}(\mathbf{0}, \mathbf{M})$
    $(\mathbf{x}_k^i, \mathbf{p}_k^i) = \text{NUTS}(\mathbf{x}_{k-1}^i, \mathbf{p}^i)$
  Calculate parameters of (22)
  Calculate (23) using (24) and (25).
  for i=1…N **do**
    Update sample weights w_k^i using (3)
  for i=1…N **do**
    Calculate normalised weights: $\widetilde{w}_k^i = \frac{w_k^i}{\Sigma_{j=1}^{j=N} w_k^j}$
  Calculate effective number of samples: $N_{\text{Eff}} = \frac{1}{\Sigma_{j=1}^{j=N} \widetilde{w}_k^{j2}}$
  if $N_{\text{Eff}} < N/2$ **then**
    Resample $[\mathbf{x}_k^1 \dots \mathbf{x}_k^N]$ with probability $[\widetilde{w}_k^1 \dots \widetilde{w}_k^N]$
    Reset all weights to $\frac{1}{N}$

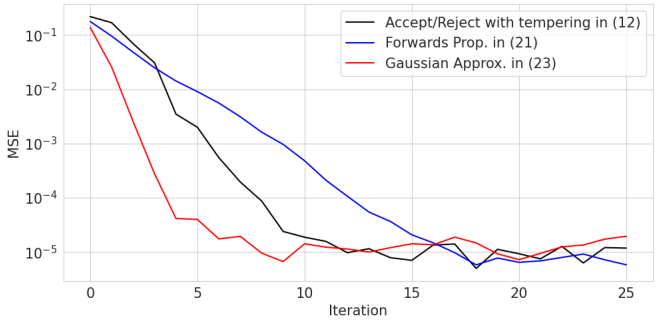

Fig. 1. Averaged mean square error of expectation values on logarithmic scale over 25 runs and 25 iterations for the ARMA model using an L-kernel parameterised by a proposal using accept/reject with tempering, a forwards proposal, and a Gaussian approximation to the optimal L-Kernel.

A. Auto-Regressive Moving Average Model

In this example, we compare performance in an auto-regressive moving average (ARMA) model. For a given set of T observations, $Y \in \mathbb{R}^T$ we infer parameters of the model:

$$\nu_t = \mu + \beta Y_{t-1} + \theta \epsilon_{t-1} \tag{26}$$

where ν_t is the predicted value at iteration t, β and μ are coefficients, and ϵ is a zero-mean noise term $\epsilon \sim \mathcal{N}(0, \sigma^2)$ parameterised by the standard-deviation, σ. The aim is to estimate μ, β, θ, and σ, which have the following prior distributions: $\mu \sim \mathcal{N}(0, 10)$, $\beta \sim \mathcal{N}(0, 2)$, $\theta \sim \mathcal{N}(0, 2)$, and $\sigma \sim \text{Cauchy}(0, 2.5)$, respectively. In Fig. 1 we plot the averaged (over 25 runs) mean-squared-error (MSE) of the parameters for 25 iterations with 200 samples. The error is calculated with respect to values calculated after running Stan for 250,000 iterations on the same model and then averaged over the runs. Using the Gaussian approximation to the optimal L-kernel, the sampler converges to the Stan estimates in fewer iterations than either of the two other approaches.

TABLE I
AVERAGE MEAN-SQUARED-ERROR USING DIFFERENT CONFIGURATIONS OF AN SMC SAMPLER WITH NUTS AS A PROPOSAL.

N	Case	Iterations 10	25	50
512	Acc./Rej. + tempering (12)	0.0343	0.0070	0.0046
	Forwards Prop. (21)	0.0109	0.0060	0.0043
	Gauss Approx. (23)	0.0273	0.0300	0.0837
1024	Acc./Rej. + tempering (12)	0.0284	0.0011	0.0007
	Forwards Prop. (21)	0.0061	0.0030	0.0016
	Gauss Approx. (23)	0.0053	0.0081	0.0773
2048	Acc./Rej. + tempering (12)	0.0358	0.0007	0.0007
	Forwards Prop. (21)	0.0016	0.0045	0.0022
	Gauss Approx. (23)	0.0041	0.0065	0.0046

B. Penalised Regression with Count Data

In this example we estimate the parameters of a penalised regression model with count data [17]. This problem makes use of Lasso regression [18] whereby a penalty constraint $\gamma \sum_{j=1}^{D} |\boldsymbol{\beta}_j|$ is placed on the size of the regression coefficients $\boldsymbol{\beta} \in \mathbb{R}^D$. We follow [17] (with associated details from [19]) by using the exponential power distribution bridge framework for our regularizing prior:

$$f(\boldsymbol{\beta}; \gamma, z) = \prod_{j=1}^{D} \frac{z}{2\gamma\Gamma(1/z)} \exp\left(-\left|\frac{\boldsymbol{\beta}_j}{\gamma}\right|^z\right), \tag{27}$$

where $z \in (0, 2)$. Our aim is to estimate the coefficients used to generate the count data. The likelihood is a Poisson distribution $\mathbf{y}_i \sim p(\mathbf{y}_i|\boldsymbol{\mu}_i)$ for the i^{th} observation, where:

$$\boldsymbol{\mu}_i = \exp\left(\boldsymbol{\beta}_0 + \sum_{j=1}^{D} \boldsymbol{\beta}_j \Phi_i^j(\mathbf{x}_{i,j})\right). \tag{28}$$

We generate 100 observations with a 12-Dimensional $\boldsymbol{\beta}$ vector where $\boldsymbol{\beta}_0 = 1$, $\boldsymbol{\beta}_2 = 1.5$, $\boldsymbol{\beta}_4 = -2$, $\boldsymbol{\beta}_6 = 1$, $\boldsymbol{\beta}_7 = -2$, $\boldsymbol{\beta}_9 = 1.2$ and all other values set to zero. The basis function Φ is a Gaussian kernel $\Phi = \exp\left(-\frac{(\mathbf{x}_i - \mathbf{c}_j)^2}{2\mathbf{r}_j^2}\right)$ with 11 equispaced centres $\mathbf{c}_j \in [-1, 4]$, all $\mathbf{r}_j$ values are set to 0.5, and $z = 0.5$.

Table I shows the MSE averaged over 5 runs for different numbers of samples and iterations, where the ground-truth has been found by running Stan for 250,000 iterations. For this example both the configurations without accept/reject converge quicker in the initial iterations, but the baseline method results in a lower error asymptotically.

V. Conclusions

We have shown how the proposal from NUTS can be used in an SMC sampler without accept/reject. By using this approach accurate estimates can be found more rapidly. As SMC samplers are parallelisable, this motivates employing this technique on parallel architectures where samples within an iteration are ran in parallel. We note a Gaussian approximation to the optimal L-kernel is not always ideal. Our future work therefore involves extending our approach to approximate the optimal L-kernel as a Gaussian mixture. We have made an implementation of our algorithm available, see [20].

REFERENCES

[1] G. O. Roberts and O. Stramer, "Langevin diffusions and Metropolis-Hastings algorithms," Methodology and Computing in Applied Probability 4, pp. 337—357, 2002.

[2] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," Physics Letters B, Volume 195, Issue 2, 1987.

[3] M. D. Hoffman and A. Gelman, "The No-U-Turn Sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo", Journal of Machine Learning Research, vol. 15, num. 47, pp. 1593–1623, 2014.

[4] Stan https://mc-stan.org/

[5] PyMC3 https://docs.pymc.io/

[6] NumPyro http://num.pyro.ai/

[7] P. Del Moral, A. Doucet, and A. Jasra, "Sequential Monte Carlo samplers," Journal of the Royal Statistical Society: Series B (Statistical Methodology), 68(3):411–436, 2006.

[8] A. Buchholz, N. Chopin and P. E. Jacob, "Adaptive Tuning of Hamiltonian Monte Carlo Within Sequential Monte Carlo", Bayesian Analysis, vol. 16, no. 3, 2021.

[9] J. Heng, A. Bishop, G. Deligiannidis, and A. Doucet, "Controlled Sequential Monte Carlo", The Annals of Statistics, VOL. 48, No.5, 2020.

[10] A. Mousavi, R. Monsefi, and V. Elvira, "Hamiltonian Adaptive Importance Sampling", IEEE Signal Processing Letters, VOL. 28, 2021.

[11] R. M. Neal, "Annealed importance samplin" Statistics and Computing, vol. 11, no. 2, pp. 125–139, 2001

[12] G. Peters, "Topics in Sequential Monte Carlo Samplers", MSc. Thesis, 2005.

[13] P. L. Green, L. J. Devlin, R. E. Moore, R. J. Jackson, J. Li, and S. Maskell "Increasing the efficiency of Sequential Monte Carlo samplers through the use of near-optimal L-kernels", Mechanical Systems and Signal Processing, Vol. 162, 2022.

[14] R. M. Neal, "MCMC using Hamiltonian dynamics", Handbook of Markov Chain Monte Carlo, Chapman & Hall, 2011.

[15] A. Varsi, J. Taylor, L. Kekempanos, E. Pyzer Knapp and S. Maskell,"A Fast Parallel Particle Filter for Shared Memory Systems," in IEEE Signal Processing Letters, vol. 27, pp. 1570-1574, 2020.

[16] A. Varsi, S. Maskell, and P. Spirakis "An O(log2 N) Fully-Balanced Resampling Algorithm for Particle Filters on Distributed Memory Architectures" in Algorithms, vol. 14, 2021.

[17] T. L. T. Nguyen, F. Septier, G. Peters, and Y. Delignon, "Efficient Sequential Monte-Carlo samplers for Bayesian inference", *IEEE Trans. on Signal Processing*, vol. 64, no. 5, pp. 1305–1319, March1, 2016.

[18] R. Tibshirani, "Regression shrinkage and selection via the Lasso", Journal of the Royal Statisitcal Society: Series B, Vol. 58, num. 1, 1996.

[19] T. L. T. Nguyen. "Sequential Monte Carlo Sampler for Bayesian Inference in Complex Systems", Ph.D Thesis, 2014.

[20] https://github.com/UoL-SignalProcessingGroup/SMC-NUTS