A General-Purpose Fixed-Lag No-U-Turn Sampler for Nonlinear Non-Gaussian State Space Models

Alessandro Varsi, Lee Devlin, Paul Horridge and Simon Maskell

***Abstract*—Particle Filters (PFs) are commonly used Sequential Monte Carlo (SMC) algorithms to process a never-ending stream of measurements relating to a nonlinear non-Gaussian state space model. Fixed-Lag SMC (FL-SMC) is an extension to the PF that allows for re-processing of historic data. FL-SMC is widely flexible, such that it can solve problems that are challenging for standard PFs. However, FL-SMC also inherits the challenges (in terms of maximizing accuracy and efficiency) that can limit PFs' efficacy when using a poor choice of the proposal distribution: this can be especially evident in strongly nonlinear scenarios. One alternative is to employ Sequential Markov Chain Monte Carlo (S-MCMC) methods, for which the literature offers a wider selection of efficient proposal distributions. However, S-MCMC does not inherently have the broad applicability of FL-SMC. In this paper, we present the Fixed-Lag No-U-Turn Sampler, an SMC framework that combines FL-SMC and No-U-Turn Sampler (NUTS), a gradient-based MCMC method. We show that, when compared with several variants of PFs, including one that employs Particle Flow, several variants of FL-SMC, and S-MCMC, our proposed approach provides significant accuracy and efficiency improvements, at the price of a moderate run-time overhead.**

***Index Terms*—Fixed Lag Sequential Monte Carlo, Particle Filters, Sequential Markov Chain Monte Carlo, No-U-Turn Sampler.**

I. Introduction

A. Motivation

Particle Filters (PFs) are well-known Sequential Monte Carlo (SMC) methods that perform state estimation for State Space Models (SSMs) of dynamic systems in response to an incoming stream of noisy measurements. More precisely, they are commonly used in the context of nonlinear non-Gaussian models, and therefore, find application in a vast range of domains, such as machine learning [1], weather forecasting [2], medical research [3], risk assessment [4] and target tracking [5]. The idea is to recursively represent the posterior by sampling N weighted hypotheses of the state (i.e. particles) from an arbitrary proposal distribution, and to use the resampling algorithm when the variance of the particle weights becomes too low [6]. The performance of PFs is typically assessed in terms of the accuracy of their state estimation, and their efficiency, as usually quantified using the variance of the particle weights.

This work was supported in part by the U.K. EPSRC Doctoral Training Award, in part by the U.K. EPSRC Big Hypothesis Award, in part by Dstl and in part by Schlumberger.

Alessandro Varsi, Lee Devlin, Paul Horridge and Simon Maskell are with the Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool L69 3GJ, U.K. (e-mail: A.Varsi@liverpool.ac.uk; ljdevlin@liverpool.ac.uk, P.Horridge@liverpool.ac.uk, S.Maskell@liverpool.ac.uk).

We can improve the performance by using several different approaches. Relatively straightforward ideas include changing the resampling algorithm [7], [8], using tempering [9], querying more sensors whenever possible [10], or simply increasing the number of particles [11]. All of these techniques are generically applicable, but the key to increasing the performance is to ensure that the proposal puts the particles in good places, i.e., in such a way they resemble the posterior.

Therefore, we argue that in order to have a high-performance PF that is effective across a diverse range of potential applications, we need to have an effective general-purpose proposal distribution.

B. Problem Definition and Related Work

The original bootstrap filter [12] and the Sequential Importance Resampling (SIR) PF [6] use the dynamics as a proposal. Although this convenient approach is widespread, it is not the optimal proposal, since it does not incorporate information from the measurements in how the particles are proposed. Indeed, the optimal proposal is dependent on the measurements and is defined as one that maximizes the efficiency, i.e., minimizes the variance of the weights [13], [14]. In theory, it is always possible to define an optimal proposal, but in practice such a proposal is often impossible to compute in closed form, as well as being model-specific.

There have been several attempts to approximate the optimal proposal. In [15], [16] a hierarchical SMC method in which every particle is itself a PF is presented. This approach improves the accuracy but greatly increases the run-time. Also, the choice of the proposal for the inner PFs is crucial for performance. Another approach is to use a Gaussian approximation of the optimal proposal by using variants of the Kalman Filter (KF), such as the Extended KF (EKF) or the Unscented KF (UKF) [17], to incorporate information from the measurements. While these approaches are still commonly used [18]–[20], real-world models might be strongly nonlinear [21], [22], which makes it challenging to define effective KF-based proposals. One strategy in such settings is to define a problem-specific transformation of variables such that the proposal is well approximated as Gaussian in the transformed space [17]. Another approximation to the optimal proposal, called PF with Invertible Particle Flow (PFIPF), is presented in [23]. For each particle, PFIPF introduces a Particle Flow [24] step in between the prediction and update steps of a KF-based proposal for nonlinear SSMs (e.g., EKF or UKF), and has been shown to handle the curse of dimensionality in highly-dimensional models more efficiently than SIR PF or the approaches in [18]–[20]. We note that, in common with the approach that we propose, Particle flow uses gradient information to handle high-dimensional models. More precisely, Particle flow uses these

gradients when considering the transport of the probability mass between the prior and posterior distributions in such a way that the particles are moved to approximate the posterior. We adopt a different approach to capitalizing on the availability of gradient information.

Other proposal distributions have been considered in the context of Markov Chain Monte Carlo (MCMC) methods, albeit typically applied to static systems (i.e., having a posterior distribution with constant state and data). Two methods stand out in terms of accuracy and acceptance rate, especially with highly-dimensional posterior distributions: Hamiltonian Monte Carlo (HMC) and the No-U-Turn-Sampler (NUTS) [25]–[28]. These are gradient-based proposals, which means they use the gradient of the posterior (i.e., including the likelihood of the measurement) to propose new samples. There have been attempts to apply MCMC in the context of dynamic systems (i.e., having a time-varying state or data). For example, at each iteration, Sequential MCMC (S-MCMC) generates a chain of N (unweighted) samples using MCMC and then proceeds to the next iteration by conditioning the MCMC proposal on just one sample of the state at the previous iteration. Since this sample is drawn from the (previous) filtering distribution, S-MCMC is unable to refine its single sample of the previous state in light of newly received data. None-the-less, we note that there have been some approaches to using gradient-based proposals in the context of S-MCMC [29]–[31].

Another approach to making the proposal more effective and widely applicable in a dynamic context is the one used in Fixed-Lag SMC (FL-SMC) methods, which are also described in the literature as Block Sampling PFs. Similarly to Fixed-Lag smoothing, FL-SMC refines the l previous states of each particle (where l is an arbitrary time step lag), in order to improve the quality of the new set of particles. This approach was presented in [32], [33], and recently further improved in [34]. FL-SMC finds application in several domains [35]–[37], can be effectively employed within hierarchical SMC methods [38], and can be configured (by extending the lag to the full measurement history) to work on static systems [39], to which MCMC methods are typically applied.

Broadly speaking, using a lag enables the proposal in FL-SMC to revisit historic portions of the trajectory. This means that FL-SMC shares PFs' ability to handle problems where sampling errors related to the current particles will have minimal impact on the future particles, but, crucially, FL-SMC can also work well when such sampling errors have a significant impact on future particles. When it comes to the convergence proofs for PFs [40], this notion of forgetting errors is referred to as *ergodicity*: systems that are highly ergodic forget errors quickly. Therefore, in this paper we distinguish between two extreme regimes in a continuum of problems:

***Definition** 1:* We define an SSM to exhibit *short-term memory* (STM) if a substantial state estimation error at time step $t-1$ is unlikely to compromise the state estimation at time step t, i.e., it is highly ergodic.

***Definition** 2:* We define an SSM to exhibit *long-term memory* (LTM) if a substantial state estimation error at time step $t-l$, for $l>0$, is likely to compromise the state estimation of the next l time steps, i.e., the problem is not ergodic.

More intuitively, STM models are those for which performance can be maximized by using a proposal that only uses the most recent measurement, $\mathbf{Y}_t$, while LTM models are those for which it is important for the proposal to be dependent on the last l measurements, $\mathbf{Y}_{t-l:t}$.

Using the dynamics as the proposal is widespread in the context of FL-SMC, since doing so is convenient from an implementation perspective. None-the-less, in [32], it is shown that the proposal can be straightforwardly switched to an EKF smoother, and, therefore, to any KF-based smoother. While this improves performance, the use of a KF-based proposal in the context of FL-SMC encounters the same challenges as using such proposals in the context of a PF.

Therefore, we argue that the key to constructing a general-purpose proposal distribution for SSMs is to combine the wide applicability of FL-SMC with the high efficiency of gradient-based MCMC proposals. However, this idea is not straightforwardly achievable for two primary reasons. Firstly, the posterior considered in dynamic systems is time-varying. Secondly, MCMC proposals can be slow to converge unless a (potentially also slow) burn-in phase is used.

C. Our Results

The contribution of this paper is to extend the preliminary work in [41] to propose Fixed-Lag NUTS (FL-NUTS), a general-purpose sequential Bayesian filter that combines FL-SMC and NUTS. More precisely, in this work, we present the following contributions relative to the preliminary work in [41]:

- We have revised the theory behind the weight update in Equation (22), to remove an unnecessary (and undesirable) assumption present in [41]. Approximating that this assumption holds would limit the range of scenarios where FL-NUTS has utility: this is discussed in Section IV-A;
- We have improved the proposal by adding an optimization step, which we describe in Section IV-B. This optimization step is new and necessary to achieve both accurate and efficient performance, as the empirical results in Section V emphasize;
- We have significantly extended the baselines for comparison relative to those considered in [41];
- We have a larger set of more developed scenarios (relative to [41]) which are used to generate the results we present.

Indeed, the resulting filter is compared with several variants of FL-SMC, PFIPF, and S-MCMC on several exemplary non-linear non-Gaussian SSMs configured to have an increasingly challenging posterior in terms of nonlinearity, memory latency, and state dimensionality. At the price of a slower run-time (up to 12 times more intensive), FL-NUTS improves the efficiency by up to 20 times and the accuracy by up to three orders of magnitude.

In doing so, the rest of the paper is organized as follows: in Section II, we describe FL-SMC. In Section III, we give brief information about gradient-based MCMC methods and S-MCMC. In Section IV, we present our approach in detail. Section V provides the numerical results. In Section VI, we draw our conclusions and give suggestions for possible future work.

II. Fixed Lag Sequential Monte Carlo Methods

In this section, we describe the FL-SMC method. To provide context, we first explain the SIR PF, and then progress to describe FL-SMC, as the latter can be intuitively viewed as an extension of the first. Further details can be found in [6], [32], [33].

A. Sequential Importance Resampling

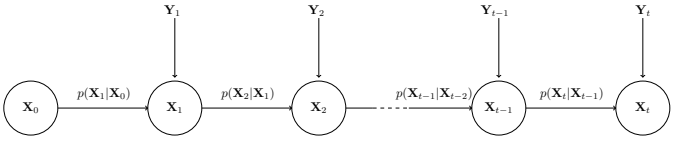


Fig. 1: SIR PF: state flow

At each time step $t = 0, 1, \ldots, T-1$, SIR employs the Importance Sampling (IS) principle to make Bayesian inferences of the true state, $\mathbf{X}_t \in \mathbb{R}^M$, of a dynamic system. The key idea consists of generating $\mathbf{x}_t \in \mathbb{R}^{N\times M}$, a population of N samples (or particles) of the true state, by randomly drawing each particle, $\mathbf{x}_t^i \in \mathbb{R}^M$, from an arbitrary proposal distribution, $q(\mathbf{X}_t|\mathbf{X}_{t-1}, \mathbf{Y}_t)$. This step is commonly referred to as prediction. Then, $\mathbf{x}_t^i$ is weighted by an unnormalized weight, $\mathbf{w}_t^i \in \mathbb{R}$, which ensures that the samples, $\mathbf{x}_t$, can be used to approximate estimates associated with $p(\mathbf{X}_t|\mathbf{Y}_{1:t})$. This step is usually called update. To update the weights at each iteration, SIR processes a datum, $\mathbf{Y}_t$, which also has to be a known function of $\mathbf{X}_t$.

At the time step $t = 0$, no data have been collected yet, so the particles can be drawn from an initial distribution, $p_0(\mathbf{X}_t)$, and weighted equally, i.e. $\mathbf{w}_t^i = 1/N \ \forall i$. For any time step $t = 1, 2, \ldots, T-1$, new measurements are collected, and each particle is drawn from the proposal distribution as follows:

$$\mathbf{x}_t^i \sim q(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, \mathbf{Y}_t), \tag{1}$$

and weighted as follows:

$$\mathbf{w}_t^i = \mathbf{w}_{t-1}^i \frac{\pi(\mathbf{x}_t^i, \mathbf{Y}_t|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, \mathbf{Y}_t)}, \tag{2}$$

where the incremental posterior, $\pi(\mathbf{X}_t, \mathbf{Y}_t|\mathbf{X}_{t-1}) = p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{Y}_t|\mathbf{X}_t)$, is known from the model. The optimal proposal is then that which minimizes the variance of the incremental weights and is defined in [13] as:

$$\begin{aligned} q^{\text{Opt}}\left(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, \mathbf{Y}_t\right) &= p\left(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, \mathbf{Y}_t\right) \\ &= \frac{p\left(\mathbf{x}_t^i, \mathbf{Y}_t|\mathbf{x}_{t-1}^i\right)}{\int p\left(\mathbf{X}_t, \mathbf{Y}_t|\mathbf{x}_{t-1}^i\right) d\mathbf{X}_t} = \frac{\pi\left(\mathbf{x}_t^i, \mathbf{Y}_t|\mathbf{x}_{t-1}^i\right)}{p\left(\mathbf{Y}_t|\mathbf{x}_{t-1}^i\right)}, \end{aligned} \tag{3}$$

which is such that the weight update with the optimal proposal is

$$\mathbf{w}_t^i = \mathbf{w}_{t-1}^i \frac{\pi(\mathbf{x}_t^i, \mathbf{Y}_t|\mathbf{x}_{t-1}^i)}{\frac{\pi\left(\mathbf{x}_t^i, \mathbf{Y}_t|\mathbf{x}_{t-1}^i\right)}{p\left(\mathbf{Y}_t|\mathbf{x}_{t-1}^i\right)}} = \mathbf{w}_{t-1}^i p\left(\mathbf{Y}_t|\mathbf{x}_{t-1}^i\right), \tag{4}$$

where the term $p\left(\mathbf{Y}_t|\mathbf{x}_{t-1}^i\right)$ is the likelihood conditioned to the previous state of each particle and is computed as follows:

$$\begin{aligned} p\left(\mathbf{Y}_t|\mathbf{x}_{t-1}^i\right) &= \int p\left(\mathbf{X}_t, \mathbf{Y}_t|\mathbf{x}_{t-1}^i\right) d\mathbf{X}_t \\ &= \int p(\mathbf{Y}_t|\mathbf{X}_t)p(\mathbf{X}_t|\mathbf{x}_{t-1}^i)d\mathbf{X}_t. \end{aligned}$$

Two problems arise when attempting to use the optimal proposal. The integral to compute $p\left(\mathbf{Y}_t|\mathbf{x}_{t-1}^i\right)$ for each i often does not have closed form solution, and, while it is theoretically possible to sample from $\pi\left(\mathbf{X}_t, \mathbf{Y}_t|\mathbf{X}_{t-1}\right)$, it is rarely achievable in practice, as well as being model specific [13], [14]. Hence, a typical approach used by SIR PFs is to propose samples from the dynamic model (as illustrated in Figure 1), $p(\mathbf{X}_t|\mathbf{X}_{t-1})$, such that (2) becomes $\mathbf{w}_t^i = \mathbf{w}_{t-1}^i p(\mathbf{Y}_t|\mathbf{x}_t^i)$.

To ensure the samples can be used to generate estimates with respect to $p\left(\mathbf{X}_t|\mathbf{Y}_{1:t}\right)$ and not just $p\left(\mathbf{X}_t, \mathbf{Y}_t|\mathbf{Y}_{1:t-1}\right)$, the weights are then normalized (to sum to unity) as follows:

$$\tilde{\mathbf{w}}_t^i = \frac{\mathbf{w}_t^i}{\sum_{j=0}^{N-1} \mathbf{w}_t^j}. \tag{5}$$

An estimate of the current state can then be computed as the weighted sum of the particles:

$$\mu_{\mathbf{X}_t} = E_{p(\mathbf{X}_t|\mathbf{Y}_{1:t})}[\mathbf{X}_t] = \sum_{i=0}^{N-1} \tilde{\mathbf{w}}_t^i \mathbf{x}_t^i. \tag{6}$$

The previous operations are however subjected to a numerical error, called particle degeneration, which makes all weights but one decrease towards 0, and hence causes (6) to diverge from (the true value of) $\mathbf{X}_t$. To correct for this error, the typical approach is to perform resampling, an algorithm that removes the particles with low weights and replaces them with copies of the particle(s) with high weight(s). In SIR, resampling is performed if the effective sample size (ESS)

$$N_{eff} = \frac{1}{\sum_{i=0}^{N-1} (\tilde{\mathbf{w}}_t^i)^2} \tag{7}$$

drops below an arbitrary threshold, typically set to $N/2$. Many resampling variants can be found in the literature [7]. Here, we only consider systematic resampling for all SMC methods, since the novelty of this work only focuses on IS. Systematic resampling first computes the cumulative density function of $N\tilde{\mathbf{w}}_t$, $\mathbf{cdf} \in \mathbb{R}^N$, as follows:

$$\mathbf{cdf}^i = N \sum_{j=0}^{i-1} \tilde{\mathbf{w}}^j, \quad \forall i = 0, 1, 2, ..., N-1. \tag{8}$$

Then each particle, $\mathbf{x}_t^i$, is copied as many times as:

$$\mathbf{ncopies}^i = \lceil \mathbf{cdf}^i + \tilde{\mathbf{w}}_t^i - u \rceil - \lceil \mathbf{cdf}^i - u \rceil, \tag{9}$$

where $\lceil \cdot \rceil$ is the ceiling function and $u \sim \texttt{Uniform}[0, 1)$. After that, each weight is reset to $1/N$, and the next time step starts over from (1).

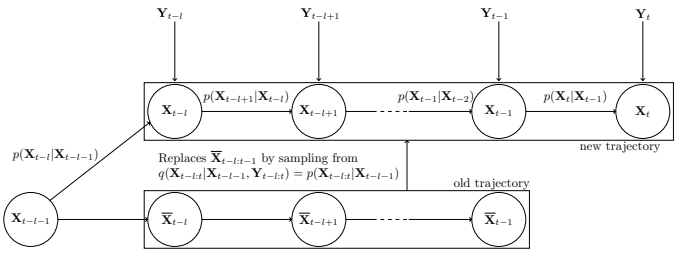


Fig. 2: FL-SMC: state flow

B. Fixed-Lag Sequential Monte Carlo

In the FL-SMC method, each particle is not a sample of $\mathbf{X}_t$ as in the SIR PF, but represents a sample of a trajectory of $l+1$ states, $\mathbf{X}_{t-l:t}$, where $l \in \mathbb{Z}^+$ is a lag. Therefore, each particle (also called particle trajectory or simply trajectory, in this method), is defined by the following notation: $\mathbf{x}^i_{t-l:t} \in \mathbb{R}^{N\times(l+1)\times M}$. The key idea to propose each $\mathbf{x}^i_{t-l:t}$ at every time step, t, is to rewind by l time steps and re-run the proposal given $\mathbf{x}^i_{t-l-1}$ and the last $l+1$ sets of data, $\mathbf{Y}_{t-l:t}$. This way, the intermediate states of each particle at time steps $\tau = t-l, t-l+1, \ldots, t-1$ are replaced and potentially improved; however, before running the proposal, these intermediate states need to be stored into a temporary vector, which we call old trajectory, $\overline{\mathbf{x}}^i_{t-l:t-1} \forall i$, as they are required to update the importance weights. Figure 2 illustrates these concepts. More precisely, at each time step, IS generates and weights each new particle trajectory $\mathbf{x}^i_{t-l:t}$ as follows:

$$\mathbf{x}^i_{t-l:t} \sim q(\mathbf{x}^i_{t-l:t}|\mathbf{x}^i_{t-l-1}, \mathbf{Y}_{t-l:t}), \tag{10}$$

such that

$$\mathbf{w}^i_t = \mathbf{w}^i_{t-1} \frac{\pi(\mathbf{x}^i_{t-l:t}, \mathbf{Y}_{t-l:t}|\mathbf{x}^i_{t-l-1}) L(\overline{\mathbf{x}}^i_{t-l:t-1}|\mathbf{x}^i_{t-l:t-1})}{\pi(\overline{\mathbf{x}}^i_{t-l:t-1}, \mathbf{Y}_{t-l:t}|\mathbf{x}^i_{t-l-1}) q(\mathbf{x}^i_{t-l:t}|\mathbf{x}^i_{t-l-1}, \mathbf{Y}_{t-l:t})}, \tag{11}$$

where the $L(\cdot)$ term is called backward kernel and is a design choice [32], [33]. This kernel expresses a probability of a backwards-going distribution and can intuitively be considered as a quantification of proximity (between the old and new trajectories) in the space of sampled trajectories. More specifically, it is a function of the old states at times $t-l, \ldots, t-1$ given the newly sampled values for those same states. The incremental posterior is computed as follows:

$$\pi\left(\mathbf{X}_{t-l:t}, \mathbf{Y}_{t-l:t}|\mathbf{X}_{t-l-1}\right) = \prod_{\tau=t-l}^{t} p\left(\mathbf{X}_\tau|\mathbf{X}_{\tau-1}\right) p\left(\mathbf{Y}_\tau|\mathbf{X}_\tau\right). \tag{12}$$

After (10) and (11), FL-SMC performs (5), (6), (7), and resampling as SIR. We also note that if $l = 0$ FL-SMC and SIR become equivalent.

As in SIR, it is theoretically possible to define an optimal proposal for FL-SMC, which again is the one that minimizes the variance of (11):

$$q^{\text{Opt}}\left(\mathbf{x}^i_{t-l:t}|\mathbf{x}^i_{t-l-1}, \mathbf{Y}_{t-l:t}\right) \propto \pi\left(\mathbf{x}^i_{t-l:t}, \mathbf{Y}_{t-l:t}|\mathbf{x}^i_{t-l-1}\right). \tag{13}$$

In such a scenario, [33] proves that the optimal choice for the $L(\cdot)$ kernel would simplify (11) to $\mathbf{w}^i_t = \pi(\mathbf{x}^i_{t-l})/\pi(\mathbf{x}^i_{t-l-1})$. It is often impossible to sample from (13). Hence, the typical proposal for the new trajectory is again the dynamics, i.e., $p(\mathbf{X}_{t:t-l}|\mathbf{X}_{t-l-1})$, and the backward kernel is chosen to match the proposal, i.e., $L(\cdot) = q\left(\overline{\mathbf{x}}^i_{t-l:t-1}|\mathbf{x}^i_{t-l-1}\right)$. However, it is straightforward to replace the dynamics with any Kalman-based proposal, such as the EKF [32]. In this case, (11) does not need to change, and we refer to the resulting method as FL-EKF. For completeness, we also consider a UKF proposal for FL-SMC, although this method is not technically considered in the literature on FL-SMC, albeit being a typical proposal for SIR. We refer to this method as FL-UKF[1].

We note that increasing l does not always translate to improved state estimation or ESS, and its optimal value varies depending on the specifics of the model, as also pointed out in [32].

III. Markov Chain Monte Carlo Methods

In this section, we first describe two gradient-based MCMC methods, HMC and NUTS, and then give a brief introduction to the S-MCMC framework. The reader is referred to [25] and [43] for further details.

A. Hamiltonian Monte Carlo

The goal of any MCMC algorithm is to generate a Markov chain of samples from a static posterior distribution, $\pi(\theta|\mathbf{Y})$. This distribution is a function of a generic sample, θ, and is defined given some data, $\mathbf{Y}$: we will explain the relationship between θ and $\mathbf{x}$ in Section IV. HMC draws upon ideas from statistical physics. It achieves the goal of sampling from a distribution of interest by hypothesizing a dynamic system that has a specific property. This property is that if we were able to simulate exactly the evolution of the system over time, the distribution of the states of the system (marginalized over time) would exactly match the distribution of interest. More formally, HMC achieves this by considering a Hamiltonian system, with potential and kinetic energy. The potential energy is a function of the posterior distribution. Each sample, θ, is viewed as an object with some kinetic energy defined by a momentum vector, $\mathbf{V}$. The object then moves around the posterior converting kinetic to potential energy and vice versa. HMC operates by considering the canonical distribution $p(\theta, \mathbf{V}) = \frac{1}{Z}\exp(-H(\theta, \mathbf{V})/T_e)$, where Z is a normalizing constant and T_e is a temperature value (we subsequently assume $T_e = 1$). $H(\theta, \mathbf{V})$ is called the Hamiltonian. It represents the total energy and is defined as the sum of the potential energy, $U(\theta)$, and the kinetic energy, $K(\mathbf{V})$: $H(\theta, \mathbf{V}) = U(\theta) + K(\mathbf{V})$. HMC sets the potential energy to be proportional to the log of the posterior distribution, as follows: $U(\theta) = -\log(\pi(\theta|\mathbf{Y}))$. The kinetic energy is set to be a function of the momentum given to the sample, $\mathbf{V}$, as follows: $K(\mathbf{V}) = \frac{1}{2}\mathbf{V}^\mathsf{T}\mathbf{m}^{-1}\mathbf{V}$ where $\mathbf{m} \in \mathbb{R}^{M\times M}$ is a mass matrix.

Every MCMC algorithm generates a chain of samples where each new sample, θ^*, is proposed from the previous one, $\overline{\theta}$.

[1]For brevity, we assume the reader is familiar with EKFs and UKFs, but further information can be found in [42].

In the specific case of HMC, the dynamics of the sample moving across the posterior are then modeled using Hamilton's equations:

$$\frac{d\mathbf{V}}{dt} = -\frac{\partial U(\theta)}{\partial \theta} \tag{14a}$$

$$\frac{d\theta}{dt} = \mathbf{m}^{-1}\mathbf{V}. \tag{14b}$$

However, the system of equations (14) typically has no closed-form solution and requires a numerical integrator, $\mathbf{f}(\theta, \mathbf{V})$. This numerical integrator is initialized to the previous sample, $\overline{\theta}$, and to $\overline{\mathbf{V}}$, a momentum vector randomly drawn from $\mathcal{N}(\mathbf{0}, \mathbf{m})$. After n integration steps of step-size Δh, $\overline{\mathbf{V}}$ becomes $\mathbf{V}^*$, and the previous position, $\overline{\theta}$, becomes the final position, θ^*, meaning that $\mathbf{f}(\overline{\theta}, \overline{\mathbf{V}}) = (\theta^*, \mathbf{V}^*)$. The final position is the proposed sample which is either accepted or rejected with an acceptance probability

$$\beta = \min(1, \exp(-H(\theta^*, \mathbf{V}^*) + H(\overline{\theta}, \overline{\mathbf{V}}))) \tag{15}$$

according to the same Rejection Sampling decision mechanism as used in Metropolis-Hastings, whose description is omitted for brevity. The process is repeated $\hat{N}$ times to generate a chain of $\hat{N}$ samples, as with any MCMC method. For performance reasons, a subset of N_b samples (typically $N_b = \hat{N}/2$) is discarded, a process which is called burn-in. The performance is then assessed over $N = \hat{N} - N_b$ samples. For MCMC methods, the ESS is approximated as follows:

$$N_{eff} = \frac{N}{1 + 2\sum_{i=0}^{N-1} c_i}, \tag{16}$$

where c_i is the autocorrelation at an offset of i. Since the samples are unweighted, the estimate in (6) becomes a mean:

$$\mu_\theta = \frac{1}{N}\sum_{i=0}^{N-1} \theta_i. \tag{17}$$

Leapfrog is the most common numerical integrator to solve (14) because it satisfies two important properties. Firstly, it is time-reversible. This means that given an initial position and momentum and applying leapfrog, then by running the integration in reverse we will arrive back at the initial state, i.e., $\mathbf{f}(\theta^*, -\mathbf{V}^*) = (\overline{\theta}, \overline{\mathbf{V}})$. Proposals that use leapfrog make the distribution being generated invariant, and therefore detailed balance is satisfied, which is required for the MCMC process to be valid. Secondly, it is a symplectic method. This means that leapfrog is able to preserve the geometric structure of the dynamics as the integration steps progress. As a result, energy is conserved, and therefore (15) remains close to unity with a small deviation, owing to small errors in the integration. As a result, samples are proposed with a high acceptance rate, i.e., quite efficiently. These properties, as we will demonstrate, will also become useful for the novelty of this work. Each integration step solves the following equations:

$$\theta_k = \theta_{k-1} + \Delta h \mathbf{m}^{-1}\mathbf{V}_{k-1} - \frac{1}{2}\Delta h^2 \mathbf{m}^{-1}\frac{\partial U(\theta_{k-1})}{\partial \theta} \tag{18a}$$

$$\mathbf{V}_k = \mathbf{V}_{k-1} - \frac{1}{2}\Delta h\left(\frac{\partial U(\theta_{k-1})}{\partial \theta} + \frac{\partial U(\theta_k)}{\partial \theta}\right) \tag{18b}$$

$\forall k = 1, 2, \ldots, n$, where the gradient terms are evaluations of the derivative of the negative of the log posterior at points on the distribution.

We note that the ideal value for the mass matrix is model-dependent. A typical choice is to set $\mathbf{m}$ to the identity matrix $\mathbf{I}$ in order to simplify (14) and (18). An optimized mass matrix will allow for efficient exploration through the posterior space. However, by setting a generic mass matrix, we place more emphasis on finding a suitable step-size to maintain high acceptance rates. There is currently active work on finding a heuristic method that would pre-tune $\mathbf{m}$ to any model [44]. However, proposing a novel approach that also makes use of such heuristics is beyond the scope of this manuscript. Therefore, from this point on, we consider $\mathbf{m} = \mathbf{I}$ for all filtering methods that use gradient-based MCMC proposals.

B. No-U-Turn Sampler

HMC requires two parameters to be tuned to sample from a distribution effectively: namely, the step-size, Δh, and the number of steps, n. NUTS auto-calibrates the number of leapfrog steps by extending a path until it turns back on itself, after which a state along the path is selected. When used in MCMC, NUTS is used to generate a single Markov chain. In this work, we utilize NUTS as a way of proposing new samples within the filter. An outline of how the process operates is described briefly below.

NUTS explores the posterior by taking leapfrog steps forwards ($+\Delta h$) and backwards ($-\Delta h$) in time from the initial state. The joint states at either end of the path, $(\theta^-, \mathbf{V}^-)$ and $(\theta^+, \mathbf{V}^+)$, are used to detect whether the path has U-turned. A U-turn is present if either of the two following conditions is met:

$$\left(\theta^+ - \theta^-\right) \cdot \mathbf{V}^- < 0 \tag{19a}$$

$$\left(\theta^+ - \theta^-\right) \cdot \mathbf{V}^+ < 0 \tag{19b}$$

NUTS decides whether to explore forwards or backwards through a Bernoulli trial with equal probability. Initially, a single step is taken, before a new direction is selected; the process then repeats but at each stage the number of steps doubles until a U-turn is detected. At its core, NUTS is a tree-building algorithm that, at a tree height of j, attempts to take 2^j steps. Upon the detection of a U-turn, the sub-tree being constructed is discarded, and a state is sampled from the completed tree. The process of discarding incomplete sub-trees guarantees that full trees are generated. This means that any leaf node in the tree is able to transition to any other without violating the U-turn condition. This ensures that the methodology is reversible and therefore detailed balance is maintained, which is required for MCMC.

Sampling a new state from the tree was initially performed with slice-sampling [25] which negates the fact that leapfrog only approximates the dynamics of the system. A state is then uniformly selected from those within the slice. However, multinomial sampling can also be used, and is currently utilized in the Stan [45] probabilistic programming language. For computational efficiency, the step-size should not be so small that large computationally expensive trees are grown.

To stop this from happening, a maximum tree depth is set to halt tree-growth if they grow too large.

The output of NUTS is a new position and momentum vector pair $(\theta^*, \mathbf{V}^*)$. In MCMC, the final momentum vector is discarded but in this work, as will be shown, this is used to compute importance weights.

C. Sequential Markov Chain Monte Carlo

Although MCMC methods are notorious in their application to static systems, it is possible to use them in dynamic contexts as well. In [29], an S-MCMC method called Sequential HMC (S-HMC) is presented. At every time step, t, this approach uses HMC to build a chain of N unweighted samples of the posterior to estimate $\mathbf{X}_t$. At the next time step, one of the samples from the previous step is picked randomly (by using a uniform random selection), and used to represent the previous state, $\mathbf{X}_{t-1}$. In the results section, we slightly augment the work in [29], and use Sequential NUTS (S-NUTS) as one of the baselines to compare our novel approach with. S-NUTS simply replaces HMC with NUTS in S-HMC. This is done to guarantee a fair comparison with our approach, which also uses NUTS. As explained in Section III-B and in [25], [43], HMC provides the same results as NUTS when it is possible to find the ideal number of leapfrog steps at compile time. However, NUTS simply removes the need to calibrate and hence is therefore generally more reliable.

IV. Fixed-Lag No-U-Turn Sampler

In this section, we describe the mathematical derivation of our approach. The general idea is to efficiently exploit the accuracy of gradient-based MCMC proposals, such as HMC and NUTS, in the prediction step of FL-SMC methods.

One could embed such MCMC algorithms (including the accept-reject step) directly into the $q(\cdot)$ terms of (10) and (11) and choose an L-kernel that enables (11) to be calculated. This approach has been successfully used when applying SMC samplers to static systems [46] (i.e., the counterpart of PFs for static posteriors). However, this approach assumes that the MCMC can easily generate samples of the pdf. This is less likely to be the case with dynamic systems (that are our focus in this paper), since the posteriors may move in space and/or change shape over time.

The approach we describe in the following sections splits up the prediction into three substeps and leads to greater accuracy and ESS albeit at the price of a (reasonable) run-time overhead.

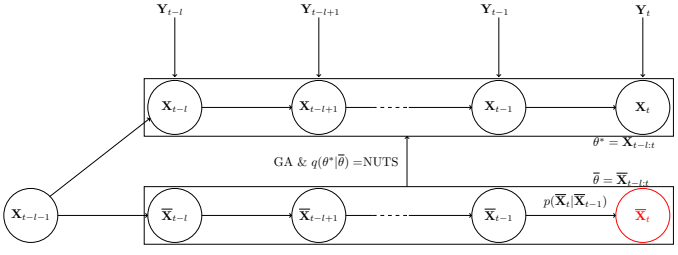


Fig. 3: FL-NUTS: state flow

A. Importance Weights

Both HMC and NUTS use the leapfrog numerical integrator in Equation (18). In this case, the proposal distribution to draw θ^* given $\bar{\theta}$ is:

$$q(\theta^*|\bar{\theta}) = \frac{q_{\mathbf{V}}(\overline{\mathbf{V}})}{||\mathbf{J}(\bar{\theta}, \overline{\mathbf{V}})||}, \tag{20}$$

where $q_{\mathbf{V}}(\cdot)$ is the proposal of the momentum vector, and $\mathbf{J}$ is the Jacobian of the leapfrog numerical integrator. By the same logic, the backward kernel can be chosen to be:

$$L(\bar{\theta}|\theta^*) = \frac{q_{\mathbf{V}}(\mathbf{V}^{back})}{||\mathbf{J}(\theta^*, \mathbf{V}^{back})||}, \tag{21}$$

where $\mathbf{V}^{back}$ is the momentum for the backward integration.

One might think that equations (20) and (21) could be applied directly to (11) if $\theta^* = \mathbf{X}_{t-l:t}$ and $\bar{\theta} = \overline{\mathbf{X}}_{t-l:t-1}$. Three problems may arise with this approach. The first is that $\mathbf{V}^{back}$ is, in general, unknown given $\mathbf{V}$. The second is that calculating the determinants of the Jacobians in (20) and (21) might be computationally prohibitive. The third is that $\theta^\star$ and $\bar{\theta}$ have different dimensionalities, making it impossible for the numerical integrator to get from one to the other.

Theorem 1 and Corollary 1 in Appendix A prove that, because the leapfrog numerical integrator in Equation (18) is reversible, then $\mathbf{V}^{back} = -\mathbf{V}^*$, the denominators in (20) and (21) may cancel when computing the ratio of $L(\cdot)$ and $q(\cdot)$. However, this can only be true if $\bar{\theta}$ has the same dimensionality as $\theta^\star$. In our setting, they do not. To overcome this issue, we draw a new "ghost" state, $\overline{\mathbf{X}}_t$, from a convenient distribution that we choose to be the dynamics, i.e., $\overline{\mathbf{X}}_t \sim p(\mathbf{X}_t|\mathbf{X}_{t-1})$ and then use HMC or NUTS. By doing this, $\bar{\theta}$ and $\theta^\star$ now have the same dimensionality, (28) holds and (11) becomes:

$$\begin{aligned}\mathbf{w}_t^i &= \mathbf{w}_{t-1}^i p(\mathbf{Y}_t^i|\overline{\mathbf{x}}_t^i)\frac{\pi(\mathbf{x}_{t-l:t}^i, \mathbf{Y}_{t-l:t}|\mathbf{x}_{t-l-1}^i)q_{\mathbf{V}}(-\mathbf{v}_{t-l:t}^i)}{\pi(\overline{\mathbf{x}}_{t-l:t}^i, \mathbf{Y}_{t-l:t}|\mathbf{x}_{t-l-1}^i)q_{\mathbf{V}}(\overline{\mathbf{v}}_{t-l:t}^i)} \\ &= \mathbf{w}_{t-1}^i \frac{\pi(\mathbf{x}_{t-l:t}^i, \mathbf{Y}_{t-l:t}|\mathbf{x}_{t-l-1}^i)q_{\mathbf{V}}(-\mathbf{v}_{t-l:t}^i)}{\pi(\overline{\mathbf{x}}_{t-l:t-1}^i, \mathbf{Y}_{t-l:t}|\mathbf{x}_{t-l-1}^i)p(\overline{\mathbf{x}}_t^i|\overline{\mathbf{x}}_{t-1}^i)q_{\mathbf{V}}(\overline{\mathbf{v}}_{t-l:t}^i)},\end{aligned} \tag{22}$$

where $\overline{\mathbf{v}}_{t-l:t}^i$ is the block (for the time steps $\tau = t-l, \ldots, t$) of momentum vectors generated from $q_{\mathbf{V}}(\cdot)$ and used to initialize leapfrog in NUTS (as also explained in Section III). At the end of the leapfrog integration steps, $\overline{\mathbf{v}}_{t-l:t}^i$ will evolve into a new block of momentum vectors, $\mathbf{v}_{t-l:t}^i$. Equation (22) is a computationally efficient way to update the importance weights in FL-SMC method when HMC or NUTS moves are used at each time step. The extra term $p(\overline{\mathbf{x}}_t^i|\overline{\mathbf{x}}_{t-1}^i)$ in the denominator takes into account that, after sampling $\overline{\mathbf{x}}_t^i$, the weight becomes $\mathbf{w}_{t-1}^i p(\mathbf{Y}_t|\overline{\mathbf{x}}_t^i)$ and that $\pi(\overline{\mathbf{x}}_t^i, \mathbf{Y}_t|\mathbf{x}_{t-l-1}^i)/p(\mathbf{Y}_t|\overline{\mathbf{x}}_t^i) = p(\overline{\mathbf{x}}_t^i|\overline{\mathbf{x}}_{t-1}^i)$. We point out that the work in [41] uses a similar approach (up to this point of the paper), in which the weight update considers the $p(\overline{\mathbf{x}}_t^i|\overline{\mathbf{x}}_{t-1}^i)$ as constant for all particles; this approximation makes the weight update faster than (22), but is only valid in specific cases, which we do not anticipate to have widespread relevance or utility. Note that (22) prioritizes those new trajectories that have improved the most with respect to their old counterpart,

up to a factor $q_{\mathbf{V}}(-\mathbf{v}^i_{t-l:t})/q_{\mathbf{V}}(\overline{\mathbf{v}}^i_{t-l:t})$. The reader may like to consider this product of ratios being analogous to that in (15).

B. Convergence Acceleration

As we will show empirically in the numerical results in Section V, simply using a NUTS move per particle within IS, and updating each particle weight as in (22), may provide competitive state estimates, on one hand, but also very poor ESS, on the other hand. This indicates that a large portion of the particle population is not actually contributing to the estimate of $\mathbf{X}_t$. This convergence problem is mainly due to two reasons. First, MCMC methods, including gradient-based methods such as NUTS, are notoriously slow at converging. Second, any SMC method that employs IS generates the particles independently, rather than as multiple outputs from the same chain (as done in S-MCMC methods).

This convergence problem is also complicated by having to sample from a dynamic posterior distribution: at each t the target moves, and we are given a new measurement, meaning that the particles that represented the pdf of $\mathbf{X}_{t-1}$ (at time $t-1$) may be a poor approximation to the current pdf of $\mathbf{X}_t$.

To accelerate the convergence of the filter, we advocate an additional step before performing the NUTS move to propose a new trajectory, $\mathbf{x}^i_{t-l:t}$. More specifically, we consider an optimization algorithm, such as Adagrad, a gradient ascent (GA) with an adaptive learning rate, η, to find a local maximum of the posterior distribution, before sampling through NUTS. Appendix B offers a brief description of this approach, but further details can be found in many textbooks, such as [47].

In summary, the approach we propose splits up the prediction in IS (which traditionally only performs sampling) into three phases in this order: first, we extend the old trajectory as (22) requires; then we optimize, and finally, we sample with NUTS. The three phases are linked together since we initialize NUTS to the output of Adagrad which is given the old trajectory in input.

After the prediction, the weights are updated as in (22). Algorithm 1 illustrates a pseudocode summarizing the ideas described in this section, which are also graphically represented in Figure 3.

As anticipated in the introduction, we refer to a FL-SMC method that employs the IS step described by Algorithm 1 as Fixed-Lag No U-Turn Sampler, or simply FL-NUTS.

C. Remarks

Remark 1: We note that this use of GA before NUTS would be likely to cause issues related to detailed balance if we were using MCMC. However, we emphasize that we are operating in an IS context.

Remark 2: We also note that Adagrad is used in Algorithm 1 for its simplicity and for providing much faster convergence than other first order approaches with fixed step-size: we note that it is one of the most commonly used optimization algorithms [48], [49]. However, Adagrad could be straightforwardly replaced with a more recent and advanced optimization algorithm [50]–[52]. Comparisons between different variants of Algorithm 1 using alternative approaches for the optimization step is a good starting point for future development, although beyond the scope of this paper.

Algorithm 1 Importance Sampling for FL-NUTS

Input: $\mathbf{x}_{t-l-1:t-1}$, $\mathbf{Y}_{t-l:t}$, $\mathbf{w}_{t-l-1}$, N, l, Δh, η, γ, k_{max}
Output: $\mathbf{x}_{t-l:t}$, $\mathbf{w}_t$

for $i \leftarrow 0$; $i < N$; $i \leftarrow i+1$ **do** Save old trajectories
  $\overline{\mathbf{x}}^i_{t-l:t-1} \leftarrow \mathbf{x}^i_{t-l:t-1}$
end for
for $i \leftarrow 0$; $i < N$; $i \leftarrow i+1$ **do** Extend old trajectories
  $\overline{\mathbf{x}}^i_t \sim p(\overline{\mathbf{x}}^i_t | \overline{\mathbf{x}}^i_{t-1})$
end for
for $i \leftarrow 0$; $i < N$; $i \leftarrow i+1$ **do** Optimization
  $\mathbf{x}^i_{t-l:t} \leftarrow \overline{\mathbf{x}}^i_{t-l:t}$
  $\mathbf{x}^i_{t-l:t} \leftarrow \texttt{Adagrad}(\mathbf{x}^i_{t-l:t}, \eta, \gamma, k_{max}, \pi(\mathbf{x}^i_{t-l:t}, \mathbf{Y}_\tau | \mathbf{x}^i_{t-l-1}))$
end for
for $i \leftarrow 0$; $i < N$; $i \leftarrow i+1$ **do** Sampling
  $\overline{\mathbf{v}}^i_{t-l:t} \sim q_{\mathbf{V}}(\overline{\mathbf{v}}^i_{t-l:t})$
  $\mathbf{x}^i_{t-l:t}, \mathbf{v}^i_{t-l:t} \sim \texttt{NUTS}(\mathbf{x}^i_{t-l:t}, \Delta h, \pi(\mathbf{x}^i_{t-l:t}, \mathbf{Y}_\tau | \mathbf{x}^i_{t-l-1}), \overline{\mathbf{v}}^i_{t-l:t})$
end for
for $i \leftarrow 0$; $i < N$; $i \leftarrow i+1$ **do** Update
  $\mathbf{w}^i_t \leftarrow \mathbf{w}^i_{t-1} \frac{\pi(\mathbf{x}^i_{t-l:t}, \mathbf{Y}_{t-l:t} | \mathbf{x}^i_{t-l-1}) q_{\mathbf{V}}(-\mathbf{v}^i_{t-l:t})}{\pi(\overline{\mathbf{x}}^i_{t-l:t-1}, \mathbf{Y}_{t-l:t} | \mathbf{x}^i_{t-l-1}) p(\overline{\mathbf{x}}^i_t | \overline{\mathbf{x}}^i_{t-1}) q_{\mathbf{V}}(\overline{\mathbf{v}}^i_{t-l:t})}$
end for

Remark 3: While we will show in the next section that FL-NUTS is quite efficient, we perceive further gains from optimizing the choice of $L(\cdot)$ in (29). Indeed, there is work to optimize SMC methods (in other contexts) by defining computationally efficient approximations to the optimal backward kernel [33], [53]. Finding a better backward kernel than (29) for FL-NUTS specifically is beyond the scope of this paper, albeit an interesting avenue for future work.

Remark 4: Our approach of using optimization in the context of each particle means that each particle will be parameterized by its local mode. Note that if the distribution is multimodal (e.g. as can be encountered in multi-sensor range-only measurement models), different particles will be parameterized by different modes. Indeed, if N is sufficiently high, different subsets of particles will be drawn toward each of the different modes that are present. Applications of FL-NUTS on these scenarios are beyond the scope of this paper, but certainly are interesting future development, especially if FL-NUTS is embedded within methods that are notoriously used for multi-target tracking, such as the SMC-PHD filter [54].

V. Numerical Results

In this section, we test and compare FL-NUTS with several alternative methods on four nonlinear non-Gaussian problems. As anticipated in Sections I, II-B and III-C, the alternative approaches we consider are: FL-SMC, FL-EKF, FL-UKF, S-NUTS, and the variant of the PFIPF method in [23] which utilizes the Particle Flow step in between the UKF prediction and update. A description of PFIPF is omitted here for brevity, but the reader is referred to [23]. However, we also consider

FL-NUTS without the optimization step, i.e., the method we present up to Section IV-A. We aim to underline the importance of including an optimizer to achieve both accurate estimation and high ESS. From this point on, we refer to the method without an optimizer as FL-NUTS w/o Opt. As we note in Section IV-A, since we compare with FL-NUTS w/o Opt, we do not see the utility in also comparing with an implementation that exactly replicates that in [41], given the implicit assumptions and approximations that [41] makes with respect to FL-NUTS w/o Opt.

For each case study, we provide numerical results of the run-time, accuracy, and efficiency. On a single Monte Carlo (MC) run, these metrics are measured as follows:

- The run-time is measured in seconds taken for T time steps.
- The accuracy is expressed in terms of Mean Squared Error (MSE) of the state estimates, and (in some examples) the MSE of the estimates of some nonlinear functions of $\mathbf{X}_t$ (with a view to quantifying more accurately how well the posterior is approximated).
- The efficiency is expressed as the average percentage ESS for T time steps. We note that the ESS for S-NUTS (calculated using (16)) is not directly comparable to that calculated for the alternative approaches (using (7)), primarily because S-NUTS is measuring the efficiency with which we sample from the incremental posterior, not the efficiency with which we sample from the filtered posterior. We do include both but avoid direct comparisons between the outputs of the two equations. We note that we also discard 50% of the total samples from S-NUTS prior to calculating ESS.

The reported numerical results are then averages computed over 100 MC runs. A total of 200 particles is used for all filtering methods in each MC run.

We consider four examples designed to test several progressively challenging scenarios in terms of nonlinearity, memory latency, and state dimensionality as follows:

- An STM model with a (nearly) linear, and (nearly) Gaussian posterior;
- An STM model with a nonlinear, non-Gaussian posterior;
- An LTM model with a nonlinear, non-Gaussian posterior;
- An STM model with a nonlinear, non-Gaussian, and highly-dimensional posterior.

The implementation of NUTS is that freely available as part of CmdStan, a command line, open source Stan interface with a back end written in C++ [45]. CmdStan also provides a fast symbolic automatic differentiation, which we have used for Algorithm 2. Hence, all the source code we use is written in C++. All experiments have been run on a machine mounting a '2 Xeon Gold 6138' CPU.

A. STM Model with Nearly Linear and Gaussian Posterior

In the first example, we consider a 2D random-walk model, with measurements of the range, ρ, and the bearing, ϕ, coming from a sensor in position $\mathbf{s} = \mathbf{0}$. This example was also considered in other relevant work [17]. The Cartesian coordinates of the state and the range are expressed in meters, while the bearing is expressed in radians. The SSM is as follows:

$$\mathbf{X}_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \mathbf{X}_{t-1} + \nu_t, \tag{23a}$$

$$\mathbf{Y}_t = \begin{bmatrix} \rho \\ \phi \end{bmatrix} = \begin{bmatrix} \|\mathbf{X}_t - \mathbf{s}\| \\ \arctan\left(\frac{\mathbf{X}_{t,1} - \mathbf{s}_1}{\mathbf{X}_{t,0} - \mathbf{s}_0}\right) \end{bmatrix} + \omega_t, \tag{23b}$$

where $\mathbf{X}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$, $\nu_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$, $\omega_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, and

$$\mathbf{P} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 0.02 \end{bmatrix},$$

and the total number of time steps is $T = 100$.

According to Definition 1, (23) is specified, in this configuration, such that it exhibits short-term memory. As a result, we anticipate that setting $l = 0$ is sufficient for accurate filtering. We also anticipate that the posterior distribution is well-approximated as Gaussian with a relatively small amount of measurement noise applied to the bearings with respect to the process noise.

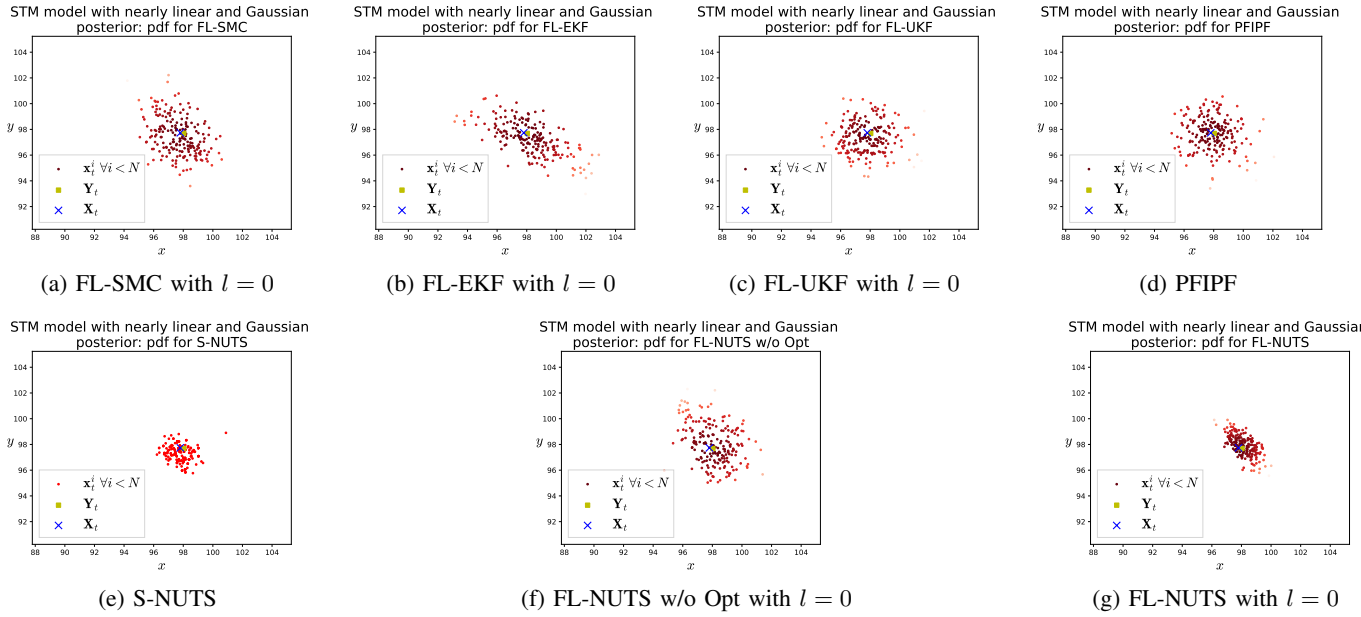
TABLE I: STM model with nearly linear and Gaussian posterior - results for 200 particles, and $T = 100$. The * is to remind the reader that N_{eff} in S-NUTS is computed as in (16) and 50% of the samples are burned-in.

Filter	l	**Time [s]**	$\mu_{\mathbf{X}_t}$	μ_ρ	$\frac{N_{eff}}{N}$
FL-SMC	0	0.67	2.69	1.86	0.442
FL-EKF	0	1.23	1.96	0.97	0.515
FL-UKF	0	1.76	2.01	3.08	0.590
PFIPF	–	2.05	2.07	1.34	0.621
S-NUTS	–	2.08	2.55	0.72	0.665*
FL-NUTS w/o Opt	0	2.15	2.52	1.76	0.438
FL-NUTS	0	3.03	2.41	0.62	0.701

For this experiment, we set the leapfrog step-size $\Delta h = 0.008$ for the three methods that involve NUTS, and for FL-NUTS we have set the initial learning rate $\eta = 0.1$, and the final convergence threshold $\gamma = 0.01$. These values were found empirically, and we acknowledge that future work would sensibly investigate automated settings of these parameters.

Table I provides the numerical results for this scenario. As we can see, all filters manage to follow the target with commensurate accuracy. A slight accuracy advantage is observed in the FL-EKF, FL-UKF, and PFIPF methods relative to the others (including FL-NUTS). This is expected since the nonlinearity is not particularly pronounced and the posterior is well approximated as Gaussian (see Figure 7a). In other words, the Kalman-based proposals use linearization of the model that is applicable across the bulk of the probability mass. All methods seem to represent the posterior well, as exemplified in Figure 4. Note that S-NUTS and FL-NUTS capture better the limited extent to which the posterior is non-Gaussian, which translates to a better MSE of the estimated range.

This advantage is also seen in terms of N_{eff}: FL-NUTS is between 1.1 and 1.8 times more efficient than all other PF or FL methods, including FL-NUTS w/o Opt, as we anticipated in Section IV. However, in such a simple model, this efficiency



(a) FL-SMC with $l = 0$ (b) FL-EKF with $l = 0$ (c) FL-UKF with $l = 0$ (d) PFIPF

(e) S-NUTS (f) FL-NUTS w/o Opt with $l = 0$ (g) FL-NUTS with $l = 0$

Fig. 4: STM model with nearly linear and Gaussian posterior - pdfs for an arbitrary time step t. For all filtering methods, except S-NUTS, the particles, $\mathbf{x}_t$, are colored in different shades of red to represent the particles' weights, $\mathbf{w}_t$, where darker shades of red represent higher weights. The particles in S-NUTS all have the same color, as the samples in S-NUTS are unweighted.

boost does not compensate for the extra run-time, as FL-NUTS is between 1.4 and 4.5 times slower than the other filters. We also note that the efficiency for S-NUTS is also commensurate here, albeit with different definitions for ESS.

B. STM Model with Nonlinear Non-Gaussian Posterior

In this second experiment, we are interested in comparing the same filtering methods as in Section V-A, when the posterior distribution is nonlinear and non-Gaussian. To consider such a setting, we modify the model described in (23) by significantly increasing the uncertainty on the bearing measurement while making the range observation considerably more informative. More precisely, we modify $\mathbf{R}$ to be:

$$\mathbf{R} = \begin{bmatrix} 0.001 & 0 \\ 0 & 1 \end{bmatrix}.$$

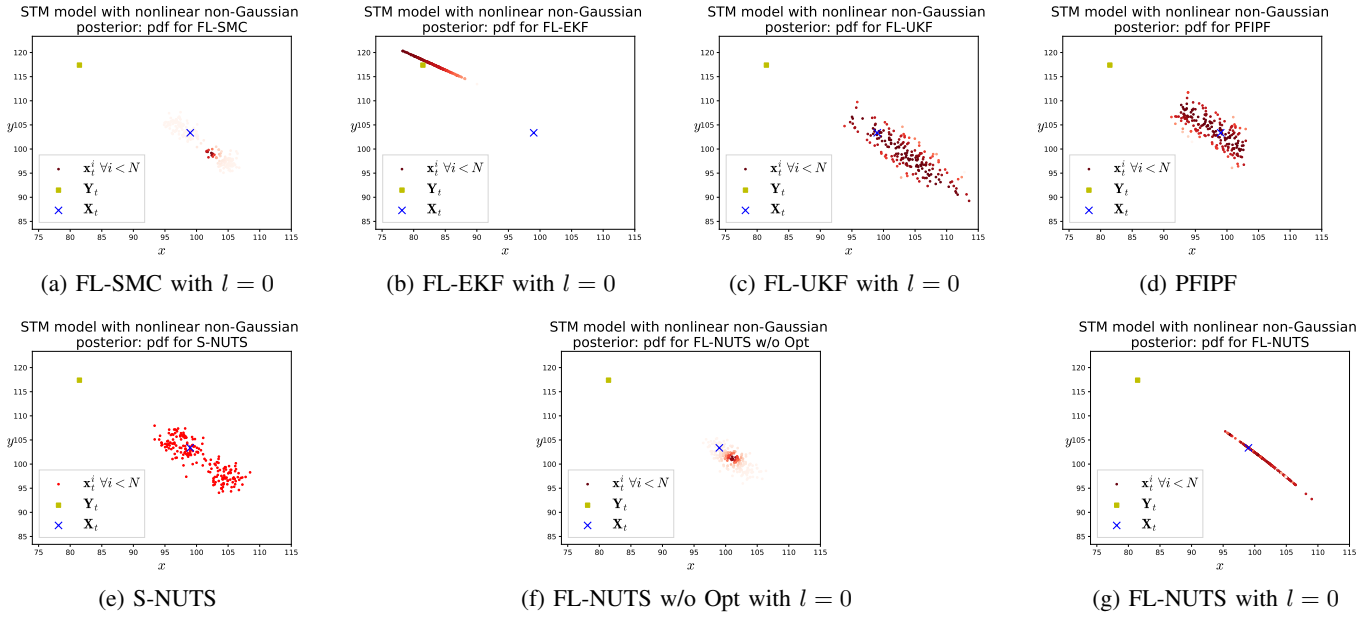
This is a routine scenario when we deal with data coming from radars, and the resulting posterior is the common "banana-shaped" pdf [17] (see Figure 7b), and notoriously challenging to sample from. The model still has short-term memory, and hence we use $l = 0$ for all SMC methods. Because of the challenges that tackling this model involves, we reset the convergence threshold in FL-NUTS to a more demanding value, precisely $\gamma = 0.001$.

Table II provides the numerical results for this second scenario. Here, FL-NUTS provides the best state and range MSEs. More precisely, in comparison with PFIPF, FL-UKF, S-NUTS, FL-SMC, and FL-NUTS w/o Opt, FL-NUTS improves the MSE for $\mathbf{X}_t$ by up to a factor of 4.8, and its MSE for ρ is at least three orders of magnitude better. In this case, FL-EKF happens to have a competitive MSE for ρ, but the MSE on the state estimation is about three orders of magnitude larger than the one for FL-NUTS, due to the poor linearization on this highly nonlinear model. These accuracy improvements for FL-NUTS are the result of a higher fidelity approximation of the posterior (see Figure 5).

TABLE II: STM model with nonlinear non-Gaussian posterior - results for 200 particles, and $T = 100$. The * is to remind the reader that N_{eff} in S-NUTS is computed as in (16) and 50% of the samples are burned-in.

Filter	l	**Time [s]**	$\mu_{\mathbf{X}_t}$	μ_ρ	$\frac{N_{eff}}{N}$
FL-SMC	0	0.78	23.14	1.21	0.039
FL-EKF	0	1.46	3210.26	0.01	0.361
FL-UKF	0	1.97	12.01	3.16	0.202
PFIPF	–	2.12	8.19	1.94	0.358
S-NUTS	–	2.21	13.76	1.79	0.041*
FL-NUTS w/o Opt	0	2.24	7.37	1.19	0.034
FL-NUTS	0	3.54	4.83	10^{-3}	0.754

Once again, this also translates to a better average ESS. The reported N_{eff} for FL-NUTS is up to 25 times larger than that for the other filtering methods considered, with FL-SMC, and FL-NUTS w/o Opt being the least competitive in terms of ESS. Indeed, FL-SMC struggles because the informative range measurements are not used in the proposal. As explained in Section IV-A, FL-NUTS w/o Opt experiences convergence issues, when having to deal with such a challenging model. As expected, FL-EKF's use of linearization makes it both



(a) FL-SMC with $l = 0$ (b) FL-EKF with $l = 0$ (c) FL-UKF with $l = 0$ (d) PFIPF

(e) S-NUTS (f) FL-NUTS w/o Opt with $l = 0$ (g) FL-NUTS with $l = 0$

Fig. 5: STM model with nonlinear non-Gaussian posterior - pdfs for an arbitrary time step t. For all filtering methods, except S-NUTS, the particles, $\mathbf{x}_t$, are colored in different shades of red to represent the particles' weights, $\mathbf{w}_t$, where darker shades of red represent higher weights. The particles in S-NUTS all have the same color, as the samples in S-NUTS are unweighted.

quite confident and also quite inaccurate. In this scenario, the accuracy and efficiency improvements provide plenty of compensation for the run-time overhead, as FL-NUTS is again no more than approximately 4.5 times slower than the other filters. We point out that the increases in N_{eff} are greater than $\sqrt{4.5}$: according to the law of large numbers, the variance of an MC estimate scales as $\frac{1}{\sqrt{N}}$ if the N samples are independent (and running for 4.5 times longer would mean we had roughly 4.5 times as many samples). We also note that akin to FL-NUTS w/o Opt, S-NUTS is also suffering from burn-in, which results in inefficient sampling from the incremental posterior.

C. LTM Model with Nonlinear Non-Gaussian Posterior

In this third experiment, we want to tackle a model with long-term memory that also requires sampling from a nonlinear non-Gaussian posterior. To achieve that, we start again from (23) but we change the prior covariance matrix to:

$$\mathbf{P} = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix},$$

while using the same matrix $\mathbf{Q}$ as in the previous experiments. At each time step, we assume to be given range and bearing measurements coming from a radar in position $\mathbf{s}_0 = [0, 0]$, as in the previous section. However, in this experiment, we also query a second radar in position $\mathbf{s}_1 = [100, 0]$ every $\Delta = 4$ time steps. For both radars, the measurement noise covariance $\mathbf{R}$ is the same as in Section V-B.

This model presents more layers of complexity than the model in Section V-B which makes it more challenging to accurately track the target. First, the prior covariance is 100 times larger than in the previous scenarios, meaning that the particles at $t = 0$ may be quite far from the true state; second, we query only one radar whose bearing measurement has high variance for the first $\Delta - 1$ steps. At time step $t = \Delta$ we query the second radar, which reduces the uncertainty on the true state, but also abruptly reshapes the posterior from a banana distribution with highly selective range to a quite heavily tailed distribution that is somewhat Gaussian (see Figures 7b and 7c). According to Definition 2, (23) with this configuration becomes an LTM model with a lag of $l = \Delta - 1$ time steps. In this example, we measure the ability of the filtering methods to localize the target after the initialization, and hence, all MC runs consider $T = 4$. We point out that it has been necessary to increase significantly the initial learning rate of FL-NUTS to $\eta = 25$, in order to take into account the much larger prior covariance, but we have kept the other parameters unchanged.

Table III shows the numerical results for this third scenario. On a qualitative level, we can say that all filters experience more difficulties in following the target relative to the previous experiments, which is expected given the extra challenges that this model presents. However, we can also say that FL-NUTS still reports good accuracy and efficiency, and offers substantial improvements relative to the other methods.

Given the LTM nature of this model, it is unsurprising to observe that all FL methods have improved performance when $l = 3$ with respect to the same methods when $l = 0$. In comparison with FL-SMC with $l = 3$, FL-UKF with $l = 3$, and FL-NUTS w/o Opt with $l = 3$, FL-NUTS with $l = 3$ reports a better MSE for the state by up to a factor of 2.5 and at least a two times smaller MSE in range. FL-NUTS is again significantly more accurate than FL-EKF (e.g., by

a factor of 20 in MSE of the state), but also S-NUTS (by almost a factor of 25 in MSE of the state). Indeed, FL-EKF is again struggling due to the linearization performed across the measurement(s) likelihood. Meanwhile, S-NUTS is unable to follow the target for two reasons. First, it does not possess the ability to correct previous state estimates. Second, S-NUTS conditions the incremental posterior at time step t to a single sample of $\mathbf{X}_{t-1}$. This means that poor past estimates (e.g. due to the large prior covariance) have a significant effect on later estimates. Just like S-NUTS, PFIPF is not equipped with the ability to reprocess historic data, but, in contrast to S-NUTS, PFIPF is an SMC method. Therefore, it is unsurprising that the performance of PFIPF is comparable to that of any FL method with $l = 0$. More precisely, PFIPF performs better than FL-UKF with $l = 0$, due to the improvements provided by the extra Particle Flow step, but not as well as most FL methods with $l = 3$. We conjecture that a variant of PFIPF with lag could be developed in the future, but we are unaware of any such method already existing. FL-NUTS' accuracy improvements are again a direct consequence of its improved ability to represent the posterior distribution: indeed, Figure 6g illustrates how the particles in FL-NUTS respond more promptly to the changes in shape of $\pi(\mathbf{X}_t, \mathbf{Y}_t|\mathbf{X}_{t-1})$ across consecutive time steps, and estimate more accurately the position of $\mathbf{X}_t$ when the second measurement arrives at $t = 4$.

In terms of ESS, FL-NUTS is again vastly more efficient than the other filtering methods, which are struggling in part for similar reasons to those relevant to the previous experiment and in part due to the bigger $\mathbf{P}$: e.g. the ESS for FL-SMC plunges to about 1.0, despite being again the fastest method. Therefore, as in the previous scenario, we can again conclude that the extra run-time that FL-NUTS requires (here being up to six times slower) is well spent in achieving better accuracy and efficiency. We note that S-NUTS offers good ESS, but that since this sample size is pertinent to the sequential posterior for a fixed previous state, this is misleading. We perceive that S-NUTS' improvements in ESS relative to the results in section V-B are caused by the incremental posterior for a very improbable (a posteriori) previous state being relatively easy for NUTS to sample from.

D. STM Model with Highly-Dimensional Nonlinear Non-Gaussian Posterior

In this final example, we consider an STM highly-dimensional nonlinear non-Gaussian scenario. More precisely, this scenario is a multivariate generalized hyperbolic skewed-t distribution, which was also considered in relevant work on S-NUTS and PFIPF [23], [29], and is described by the following dynamic model:

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) = \frac{K_{\frac{d+M}{2}}\left(\sqrt{(d+\xi(\mathbf{X}_t))(\mathbf{G}^T\mathbf{Q}^{-1}\mathbf{G})}\right)}{\left(1+\frac{\xi(\mathbf{X}_t)}{d}\right)^{\frac{d+M}{2}}} \times \times \frac{e^{(\mathbf{X}_t-\mathbf{A}\mathbf{X}_{t-1})^\intercal\mathbf{Q}^{-1}\mathbf{G}}}{\sqrt{(d+\xi(\mathbf{X}_t))(\mathbf{G}^T\mathbf{Q}^{-1}\mathbf{G})}^{-\frac{d+M}{2}}}, \quad (24)$$

TABLE III: LTM model with nonlinear non-Gaussian posterior - results for 200 particles, and $T = 4$. The * is to remind the reader that N_{eff} in S-NUTS is computed as in (16) and 50% of the samples are burned-in.

Filter	l	Time [s]	$\mu_{\mathbf{X}_t}$	μ_ρ	$\frac{N_{eff}}{N}$
FL-SMC	0	0.01	52.47	2.87	0.005
FL-SMC	3	0.04	31.73	2.96	0.016
FL-EKF	0	0.03	191.03	26.37	0.131
FL-EKF	3	0.11	194.78	25.94	0.116
FL-UKF	0	0.03	56.24	6.45	0.107
FL-UKF	3	0.12	32.51	6.73	0.215
PFIPF	–	0.03	50.48	5.83	0.128
S-NUTS	–	0.03	249.08	11.68	0.443*
FL-NUTS w/o Opt	0	0.04	51.93	2.95	0.005
FL-NUTS w/o Opt	3	0.15	29.46	3.06	0.015
FL-NUTS	0	0.06	28.72	1.62	0.174
FL-NUTS	3	0.24	11.36	1.59	0.562

and the following measurement model:

$$p(\mathbf{Y}_t|\mathbf{X}_t) = \prod_{j=0}^{M-1} \mathcal{P}(\mathbf{Y}_t^j; n_1 e^{n_2 \mathbf{X}_t^j}), \quad (25)$$

where $K_{\frac{d+M}{2}}(\cdot)$ is the modified Bessel function of the second kind of order $\frac{d+M}{2}$, the function $\xi(\mathbf{X}_t) = (\mathbf{X}_t - \mathbf{A}\mathbf{X}_{t-1})^\intercal\mathbf{Q}^{-1}(\mathbf{X}_t - \mathbf{A}\mathbf{X}_{t-1})$, and the state transition matrix, $\mathbf{A} = 0.9\mathbf{I}$, where $\mathbf{I}$ is an identity matrix of M dimensions. The values d, and the matrix $\mathbf{G}$ are arbitrary and determine the shape of the distribution. We have M sensors uniformly placed on a two-dimensional grid $\{1, 2, \ldots, \sqrt{M}\} \times \{1, 2, \ldots, \sqrt{M}\}$. The process noise covariance is given by the following equation:

$$\mathbf{Q} = \frac{d}{d-2}\hat{\mathbf{Q}} + \frac{d^2}{(2d-8)(\frac{d}{2}-1)^2}\mathbf{G}\mathbf{G}^\intercal, \quad (26)$$

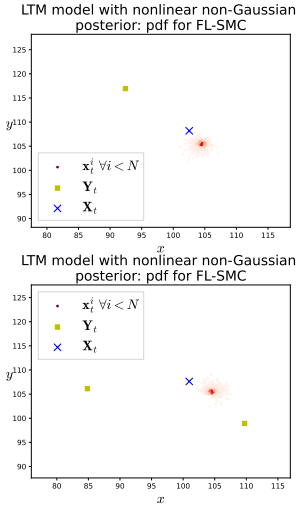
where each i, j term in $\hat{\mathbf{Q}}$ is computed as follows:

$$\hat{\mathbf{Q}}_{i,j} = 3e^{-\frac{||\mathbf{s}_i-\mathbf{s}_j||_2^2}{20}} + 0.01\delta_{i,j}, \quad \forall i, \forall j, \quad (27)$$

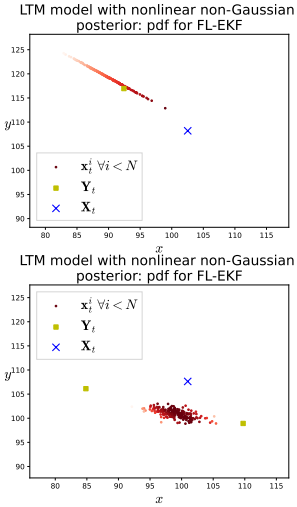
where $\mathbf{s}_i$, and $\mathbf{s}_j$ are the i-th and j-th sensors, respectively. The term $\delta_{i,j}$ is the Kronecker symbol, which is equal to $\delta_{i,j} = 1$ if $i = j$, and equal to $\delta_{i,j} = 0$ if $i \neq j$.

$\mathcal{P}(\cdot;\cdot)$ is a Poisson distribution, where the hyperparameters $n_1 = 1$, $n_2 = 1/3$ as in [23], [29]. The state dimensionality, M, is arbitrary, and in [23], [29] several values of M up to $M = 400$ were considered. For brevity, in this work, we directly report the results for $M = 400$. All filters have been initialized in the true state 0 for all state dimensions, which is the scenario that has been considered in [23], [29]. Also, we have used $T = 10$ time steps as in [23], [29]. The leapfrog step-size has been empirically set to $\Delta = 0.01$ for all methods using NUTS, and for FL-NUTS we have utilized $\eta = 1.0$, and $\gamma = 0.01$.

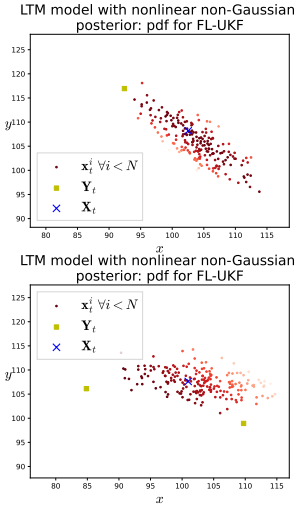
The numerical results for this final scenario are reported in Table IV. Here, we can see that only the methods that sample from the full incremental posterior (i.e., both dynamics and



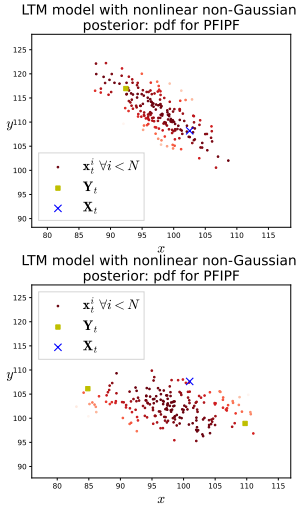
(a) FL-SMC with $l = 3$ at $t=3$ (top), and $t=4$ (bot.)



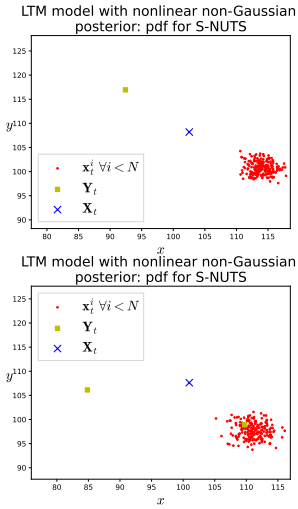
(b) FL-EKF with $l = 3$ at $t=3$ (top), and $t=4$ (bot.)



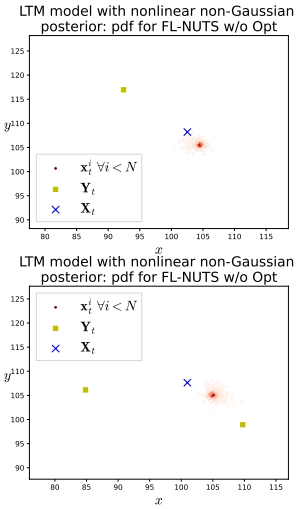
(c) FL-UKF with $l = 3$ at $t=3$ (top), and $t=4$ (bot.)



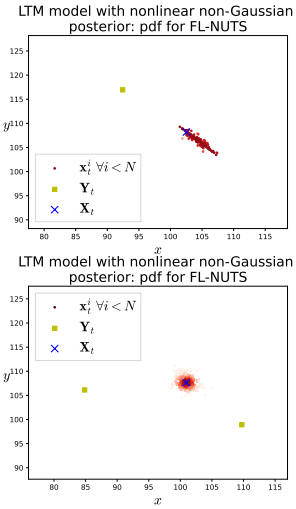
(d) PFIPF at $t=3$ (top), and $t=4$ (bot.)



(e) S-NUTS at $t=3$ (top), and $t=4$ (bot.)



(f) FL-NUTS w/o Opt with $l = 3$ at $t=3$ (top), and $t=4$ (bot.)

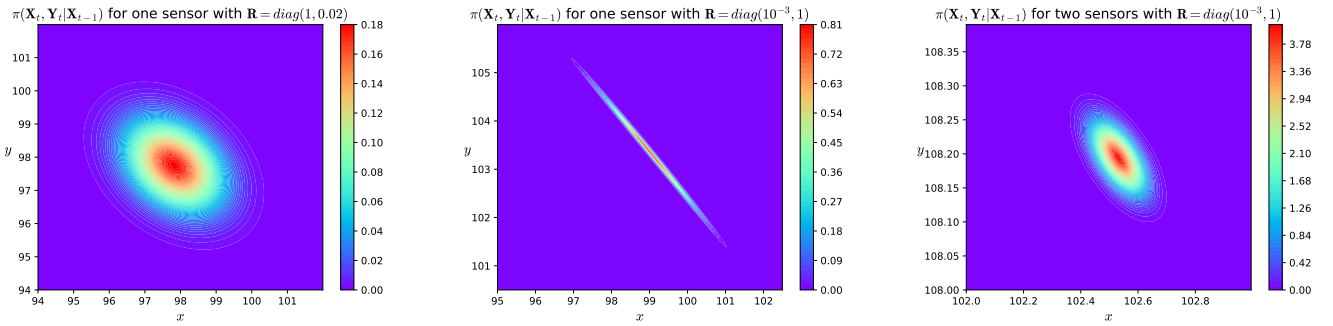


(g) FL-NUTS with $l = 3$ at $t=3$ (top), and $t=4$ (bot.)

Fig. 6: LTM model with nonlinear non-Gaussian posterior - pdfs at time steps $t = 3, 4$. For all filtering methods, except S-NUTS, the particles, $\mathbf{x}_t$, are colored in different shades of red to represent the particles' weights, $\mathbf{w}_t$, where darker shades of red represent higher weights. The particles in S-NUTS all have the same color, as the samples in S-NUTS are unweighted.

likelihood) can provide at least a moderately accurate estimate of the true state, $\mathbf{X}_t$. Indeed, FL-SMC propagates the particles by only using the dynamics, and, despite being once again the fastest method, has a significantly higher MSE than the other methods. FL-EKF and FL-UKF score a moderate average MSE with respect to the other considered methods. Indeed, the best average MSEs are scored by FL-NUTS, FL-NUTS w/o Opt, S-NUTS, and PFIPF, and are within the same order of magnitude, with FL-NUTS providing an average MSE between 1.4 and 2.0 times better than FL-NUTS w/o Opt, S-NUTS, and PFIPF.

From this point of view, it may look like in this scenario FL-NUTS does not provide significant improvement, considering the extra run-time (here being about 12 times slower than the run-time of FL-SMC). However, in such a highly-dimensional scenario, the curse of dimensionality increases the particle degeneracy, which typically results in a much lower ESS than in low-dimensional scenarios. Indeed, FL-SMC, FL-EKF, FL-UKF, FL-NUTS w/o Opt, and PFIPF score an efficiency between 0.5% and 1.7%. On the other hand, FL-NUTS achieves roughly 20% efficiency, which is indeed lower than in the previous examples, but still at least about 12 times higher than the other methods, meaning that extra run-time is again well spent. This is not unexpected, as the literature on MCMC has well documented the ability of gradient-based MCMC propos-



(a) $\mathbf{R} = diag(1, 0.02)$ & one sensor (b) $\mathbf{R} = diag(0.001, 1)$ & one sensor (c) $\mathbf{R} = diag(0.001, 1)$ & two sensors

Fig. 7: $\pi(\mathbf{X}_t, \mathbf{Y}_t|\mathbf{X}_{t-1})$ in model (23) for different $\mathbf{R}$ and number of sensors. $\mathbf{X}_{t-1}$ is the true value.

als (e.g., NUTS or HMC) to sample from highly-dimensional posteriors more efficiently than other MCMC proposals, once the burn-in period is finished [25], [43]. Furthermore, the use of the optimizer in FL-NUTS accelerates the convergence of the independent particles, especially with respect to FL-NUTS w/o Opt.

TABLE IV: STM Model with highly-dimensional nonlinear non-Gaussian posterior - results for 200 particles, and $T = 10$. The * is to remind the reader that N_{eff} in S-NUTS is computed as in (16) and 50% of the samples are burned-in.

Filter	l	**Time [s]**	$\mu_{\mathbf{X}_t}$	$\frac{N_{eff}}{N}$
FL-SMC	0	14.24	21.12	0.005
FL-EKF	0	79.56	8.2	0.011
FL-UKF	0	89.04	4.1	0.0095
PFIPF	–	100.68	1.12	0.017
S-NUTS	–	92.02	0.89	0.0521*
FL-NUTS w/o Opt	0	108.72	0.91	0.0075
FL-NUTS	0	176.86	0.54	0.2065

VI. Conclusion

In this paper, we have presented FL-NUTS, an SMC method that combines the ability of block sampling particle filters to handle blocks of measurements across multiple time steps, and the ability of gradient-based MCMC methods, such as NUTS, to explore efficiently highly challenging posterior distributions.

Despite being between two to twelve times slower (per particle) than other popular alternative methods due to the gradient computations, FL-NUTS compensates for this by greatly improving the accuracy and the effective sample size, especially in models that require sampling from strongly nonlinear and non-Gaussian posterior densities, such as when dealing with range-bearing measurements from radars or highly-dimensional states. More precisely, in such models, the estimated quantities are at least three times and (in some cases) up to three orders of magnitude more accurate; the effective sample size may increase by up to two orders of magnitude in some extreme cases, and typically by a factor of five.

Although the findings are encouraging, there still is wide room for improvements. First, the effective sample size can be increased by investigating a better backward kernel than the one we describe in Section IV-A, which was primarily designed to accelerate calculation of the weight update. In terms of run-time, the particle prediction could be accelerated by using a better optimizer than Adagrad, or an adaptive step size across different leapfrog steps during the sampling phase. We also remind the reader that the mass matrix we have used in NUTS is the identity, but future work should investigate methods to construct an adaptive mass matrix, which would likely accelerate the convergence. We also note that run-time can be straightforwardly reduced through parallel computing on clusters of big computers [55], [56]. Future work should also investigate the benefits of using FL-NUTS within other Monte Carlo methods that use PFs, such as nested SMC for filtering, particle MCMC or SMC^2 [57] for parameter estimation, or the SMC implementation of the PHD filter for multi-target tracking [54].

VII. Acknowledgements

The authors appreciate input from Dr. Alexey Narykov.

Appendix A
Reversible and Symplectic Numerical Integrators: Properties

***Theorem** 1:* Let $q(\theta^*|\bar{\theta})$ be the proposal distribution of a gradient-based MCMC method (e.g. NUTS) used by a FL-SMC method to propose a new single-state sample $\theta^* = \mathbf{X}_t$, given the old one $\bar{\theta} = \overline{\mathbf{X}}_t$ at a certain time step t. Let $L(\bar{\theta}|\theta^*)$ be the related backward kernel. If leapfrog is the chosen numerical integrator, it is possible to prove that:

$$\frac{L(\bar{\theta}|\theta^*)}{q(\theta^*|\bar{\theta})} = \frac{q_{\mathbf{V}}(-\mathbf{V}^*)}{q_{\mathbf{V}}(\overline{\mathbf{V}})}, \tag{28}$$

where $\mathbf{V}^*$ is the final momentum after n leapfrog steps, and $q_{\mathbf{V}}(\cdot)$ is the proposal for the momentum.

Proof of Theorem 1: Let $\mathbf{f}$ be the leapfrog integrator, such that $\mathbf{f}(\overline{\theta}, \overline{\mathbf{V}}) = (\theta^*, \mathbf{V}^*)$, and let $\overrightarrow{\mathbf{J}} = \mathbf{J}(\overline{\theta}, \overline{\mathbf{V}})$ be the Jacobian matrix associated to this integration. Let also $\overleftarrow{\mathbf{J}} = \mathbf{J}(\theta^*, \mathbf{V}^*)$ be the Jacobian matrix of the backward integration. Equations (20) and (21) describe generically the forward and backward kernels to be used in (28).

The first thing we can notice is that leapfrog is time reversible, which means that $\mathbf{f}(\theta^*, -\mathbf{V}^*) = (\overline{\theta}, \overline{\mathbf{V}})$. Hence, $\overleftarrow{\mathbf{J}} = \mathbf{J}(\theta^*, -\mathbf{V}^*)$ and (21) becomes

$$L(\overline{\theta}|\theta^*) = \frac{q_{\mathbf{V}}(-\mathbf{V}^*)}{||\mathbf{J}(\theta^*, -\mathbf{V}^*)||}. \tag{29}$$

To prove (28), we first need the expression for $\overrightarrow{\mathbf{J}}$:

$$\overrightarrow{\mathbf{J}} = \begin{bmatrix} \frac{\partial \theta^*}{\partial \overline{\theta}} & \frac{\partial \theta^*}{\partial \overline{\mathbf{V}}} \\ \frac{\partial \mathbf{V}^*}{\partial \theta} & \frac{\partial \mathbf{V}^*}{\partial \overline{\mathbf{V}}} \end{bmatrix}. \tag{30}$$

The determinant of this matrix has been shown to be equal to an absolute value of 1 and means that the geometric volume is preserved over the integration step [58]. This is a property of all symplectic integration methods, including leapfrog. For leapfrog, which is time reversible, it is also possible to show that the backward Jacobian matrix can be computed by transposing each block of the matrix $\overrightarrow{\mathbf{J}}$:

$$\overleftarrow{\mathbf{J}} = \begin{bmatrix} \left(\frac{\partial \theta^*}{\partial \overline{\theta}}\right)^{\mathsf{T}} & \left(\frac{\partial \theta^*}{\partial \overline{\mathbf{V}}}\right)^{\mathsf{T}} \\ \left(\frac{\partial \mathbf{V}^*}{\partial \overline{\theta}}\right)^{\mathsf{T}} & \left(\frac{\partial \mathbf{V}^*}{\partial \overline{\mathbf{V}}}\right)^{\mathsf{T}} \end{bmatrix}. \tag{31}$$

In other words, if $\mathbf{B}$ is an arbitrary block matrix, and we let α be the operation

$$\alpha(\mathbf{B}) = \alpha\left(\begin{bmatrix} \mathbf{B}^{1,1} & \mathbf{B}^{1,2} \\ \mathbf{B}^{2,1} & \mathbf{B}^{2,2} \end{bmatrix}\right) = \begin{bmatrix} \left(\mathbf{B}^{1,1}\right)^{\mathsf{T}} & \left(\mathbf{B}^{1,2}\right)^{\mathsf{T}} \\ \left(\mathbf{B}^{2,1}\right)^{\mathsf{T}} & \left(\mathbf{B}^{2,2}\right)^{\mathsf{T}} \end{bmatrix}, \tag{32}$$

we can then say that for a single leapfrog step:

$$\overleftarrow{\mathbf{J}} = \alpha(\overrightarrow{\mathbf{J}}). \tag{33}$$

Now, we need to consider the generic scenario, where the integration $\mathbf{f}(\overline{\theta}, \overline{\mathbf{V}}) = (\theta^*, \mathbf{V}^*)$ is the result of n leapfrog steps, such that we have a sequence of position-momentum pairs $(\overline{\theta}, \overline{\mathbf{V}}), (\theta^0, \mathbf{V}^0), (\theta^1, \mathbf{V}^1), ..., (\theta^{n-1}, \mathbf{V}^{n-1}) = (\theta^*, \mathbf{V}^*)$. In this case, the forward Jacobian for the full forward integration is given by the following matrix product:

$$\overrightarrow{\mathbf{J}} = \overrightarrow{\mathbf{J}^{n-1}} \cdot \overrightarrow{\mathbf{J}^{n-2}} \cdot \overrightarrow{\mathbf{J}^{n-2}} \cdots \overrightarrow{\mathbf{J}^{1}} \cdot \overrightarrow{\mathbf{J}^{0}}, \tag{34}$$

while $\overleftarrow{\mathbf{J}}$ is computed as follows:

$$\overleftarrow{\mathbf{J}} = \overleftarrow{\mathbf{J}^{0}} \cdot \overleftarrow{\mathbf{J}^{1}} \cdot \overleftarrow{\mathbf{J}^{2}} \cdots \overleftarrow{\mathbf{J}^{n-2}} \cdot \overleftarrow{\mathbf{J}^{n-1}}, \tag{35}$$

where by (33) $\overleftarrow{\mathbf{J}^k} = \alpha(\overrightarrow{\mathbf{J}^k})$.

It is relatively straightforward to prove that for a block matrix $\mathbf{B} = \mathbf{B}^0 \cdot \mathbf{B}^1 \cdot \mathbf{B}^2 \cdots \mathbf{B}^{n-2} \cdot \mathbf{B}^{n-1}$, we have:

$$\begin{aligned} \alpha(\mathbf{B}) &= \alpha(\mathbf{B}^0 \cdot \mathbf{B}^1 \cdot \cdots \cdot \mathbf{B}^{n-2} \cdot \mathbf{B}^{n-1}) \\ &= \alpha(\mathbf{B}^{n-1}) \cdot \alpha(\mathbf{B}^{n-2}) \cdot \cdots \cdot \alpha(\mathbf{B}^1) \cdot \alpha(\mathbf{B}^0). \end{aligned} \tag{36}$$

By applying (36) to (34), we obtain:

$$\begin{aligned} \alpha(\overrightarrow{\mathbf{J}}) &= \alpha(\overrightarrow{\mathbf{J}^0}) \cdot \alpha(\overrightarrow{\mathbf{J}^1}) \cdot \cdots \cdot \alpha(\overrightarrow{\mathbf{J}^{n-2}}) \cdot \alpha(\overrightarrow{\mathbf{J}^{n-1}}) \\ &= \overleftarrow{\mathbf{J}^0} \cdot \overleftarrow{\mathbf{J}^1} \cdots \overleftarrow{\mathbf{J}^{n-2}} \cdot \overleftarrow{\mathbf{J}^{n-1}} = \overleftarrow{\mathbf{J}}, \end{aligned} \tag{37}$$

which means (33) can be applied to the full integrator. In particular, (37) automatically proves that the determinants of $\overrightarrow{\mathbf{J}}$ and $\overleftarrow{\mathbf{J}}$ have the same absolute value:

$$||\mathbf{J}(\overline{\theta}, \overline{\mathbf{V}})|| = ||\mathbf{J}(\theta^*, -\mathbf{V}^*)||, \tag{38}$$

which also means that (29) divided by (20) equals (28). ■

Collary 1: Let $\theta^* = \mathbf{X}_{t-l:t}$ be a trajectory of $l > 0$ states in a FL-SMC method, generated from its old trajectory, $\overline{\theta} = \overline{\mathbf{X}}_{t-l:t}$, by a gradient-based MCMC method (e.g. NUTS) performing n leapfrog steps per each new state. Let $\overline{\mathbf{V}} = \overline{\mathbf{V}}_{t-l:t}$ and $\mathbf{V}^* = \mathbf{V}_{t-l:t}$ be the input and the final momentum vectors for the leapfrog steps. It is possible to prove that (28) is still valid.

Proof of Corollary 1: In this case, (34) and (35) have to be reformulated respectively as follows:

$$\overrightarrow{\mathbf{J}} = \prod_{\tau=t-l}^{t} \prod_{k=n-1}^{0} \overrightarrow{\mathbf{J}_\tau^k}, \tag{39}$$

$$\overleftarrow{\mathbf{J}} = \prod_{\tau=t-l}^{t} \prod_{k=0}^{n-1} \overleftarrow{\mathbf{J}_\tau^k}. \tag{40}$$

Since (33) is still valid, we now have $\overleftarrow{\mathbf{J}_\tau^k} = \alpha(\overrightarrow{\mathbf{J}_\tau^k})$ $\forall \tau$ and $\forall k$, and by applying (36), we reach the same conclusion as in (37) and (38) because the number of states in θ^* and $\overline{\theta}$ is equal. ■

APPENDIX B
GRADIENT ASCENT WITH ADAPTIVE LEARNING RATE

Gradient descent/ascent is a first-order optimization algorithm that looks for a local minimum/maximum of a differentiable, multi-variable objective function, $\pi(\theta)$. The goal is then to look for a $\theta = \tilde{\theta}$ such that $\nabla\pi(\tilde{\theta}) \approx 0$. Considering the scope of this paper, in this appendix, we only describe gradient ascent.

The overall idea is to perform a sequence of iterations, where at each step the previous local maximum, θ_{k-1}, is moved towards the direction of $\nabla\pi(\theta)$ by an arbitrary step size, η, which is also commonly known as the learning rate. More precisely, the current local maximum is computed as follows:

$$\theta_k = \theta_{k-1} + \eta\nabla\pi(\theta_{k-1}), \tag{41}$$

where the initial value, θ_0, could be picked randomly or deterministically. This routine goes on until $|\pi(\theta_k) - \pi(\theta_{k-1})|$ falls down below a pre-defined threshold, γ. It is also common to set another exit condition on the maximum number of iteration steps, k_{max}, in order to accelerate the run-time.

The choice of the learning rate is critical for a good compromise between accuracy and run-time. Broadly speaking,

decreasing η makes the convergence slower but more accurate, while increasing η tends to speed-up the convergence at the expense of the accuracy. A typical solution to this is to use an adaptive learning rate, where the learning rate is decreased or increased as θ_{k-1} approaches or moves away from $\tilde{\theta}$. Several alternative methods have been presented in the literature [47]. As explained in Section IV-B, here we only consider the Adagrad method. In this approach, the original learning rate is rescaled by a factor of $\sqrt{\epsilon + \mathbf{g}_k}$, where ϵ is an arbitrarily small offset (typically set to 10^{-8}) to avoid division by 0 errors, and $\mathbf{g}_k$ is a vector containing the sum of all gradients up to the current iteration step. More precisely, (41) becomes:

$$\mathbf{g}_k = \mathbf{g}_{k-1} + \nabla\pi(\theta_{k-1}) \tag{42a}$$

$$\theta_k = \theta_{k-1} + \frac{\eta}{\sqrt{\epsilon + \mathbf{g}_k}}\nabla\pi(\theta_{k-1}). \tag{42b}$$

This routine is described in the following pseudocode.

Algorithm 2 Adagrad

Input: θ_0, η, γ, k_{max}, $\pi(\theta)$
Output: $\tilde{\theta}$

$\mathbf{g}_0 \leftarrow \mathbf{0}$, $k \leftarrow 1$, $\epsilon \leftarrow 10^{-8}$
while $k \leq k_{max}$ **do**
  $\mathbf{g}_k \leftarrow \mathbf{g}_{k-1} + \nabla\pi(\theta_{k-1})$
  $\theta_k \leftarrow \theta_{k-1} + \frac{\eta}{\sqrt{\epsilon+\mathbf{g}_k}}\nabla\pi(\theta_{k-1})$
  if $|\pi(\theta_k) - \pi(\theta_{k-1})| < \gamma$ **then**
    break
  end if
  $k \leftarrow k + 1$
end while
$\tilde{\theta} \leftarrow \theta_k$

We note that gradient descent only requires changing the $+$ sign after θ_{k-1} in Equations (41) and (42b) to a $-$ sign.

REFERENCES

[1] X. Ma, P. Karkus, D. Hsu, and W. S. Lee, "Particle filter recurrent neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5101–5108, Apr. 2020.

[2] P. J. van Leeuwen, H. R. Künsch, L. Nerger, R. Potthast, and S. Reich, "Particle filters for high-dimensional geoscience applications: A review," *Quarterly Journal of the Royal Meteorological Society*, vol. 145, no. 723, pp. 2335–2365, 2019.

[3] J. M. Costa, H. Orlande, H. Campos Velho, S. Pinho, G. Dulikravich, R. Cotta, and S. Cunha Neto, "Estimation of tumor size evolution using particle filters," *Journal of computational biology : a journal of computational molecular cell biology*, vol. 22, 05 2015.

[4] Q. Li and S. Y. Liang, "Degradation trend prediction for rotating machinery using long-range dependence and particle filter approach," *Algorithms*, vol. 11, no. 7, 2018.

[5] M. Bruno and A. Pavlov, "Improved sequential monte carlo filtering for ballistic target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 3, pp. 1103–1108, 2005.

[6] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[7] J. D. Hol, T. B. Schon, and F. Gustafsson, "On Resampling Algorithms for Particle Filters," in *2006 IEEE Nonlinear Statistical Signal Processing Workshop*, pp. 79–82, Sept 2006.

[8] T. Li, M. Bolic, and P. M. Djuric, "Resampling methods for particle filtering: Classification, implementation, and strategies," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 70–86, 2015.

[9] E. Herbst and F. Schorfheide, "Tempered particle filtering," *Journal of Econometrics*, vol. 210, no. 1, pp. 26–44, 2019. Annals Issue in Honor of John Geweke "Complexity and Big Data in Economics and Finance: Recent Developments from a Bayesian Perspective".

[10] G. Ren, V. Maroulas, and I. Schizas, "Distributed spatio-temporal association and tracking of multiple targets using multiple sensors," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 4, pp. 2570–2589, 2015.

[11] F. Lopez, L. Zhang, A. Mok, and J. Beaman, "Particle filtering on gpu architectures for manufacturing applications," *Computers in Industry*, vol. 71, pp. 116–127, 2015.

[12] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, pp. 107–113(6), April 1993.

[13] C. Andrieu, A. Doucet, and E. Punskaya, *Sequential Monte Carlo Methods for Optimal Filtering*, pp. 79–95. New York, NY: Springer New York, 2001.

[14] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and computing*, vol. 10, pp. 197–208, 2000.

[15] C. Naesseth, F. Lindsten, and T. Schön, "Nested sequential monte carlo methods," in *International Conference on Machine Learning*, pp. 1292–1301, PMLR, 2015.

[16] C. A. Naesseth, F. Lindsten, and T. B. Schön, "High-dimensional filtering using nested sequential monte carlo," *IEEE Transactions on Signal Processing*, vol. 67, no. 16, pp. 4177–4188, 2019.

[17] S. Maskell, M. Briers, R. Wright, and P. Horridge, "Tracking using a radar and a problem specific proposal distribution in a particle filter," *IEE Proceedings - Radar, Sonar and Navigation*, vol. 152, pp. 315–322(7), October 2005.

[18] Y. Xu, K. Xu, J. Wan, Z. Xiong, and Y. Li, "Research on particle filter tracking method based on kalman filter," in *2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC)*, pp. 1564–1568, 2018.

[19] M. J. Ransom, L. Vladimirov, P. R. Horridge, J. F. Ralph, and S. Maskell, "Integrated expected likelihood particle filters," in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pp. 1–8, 2020.

[20] Y. Wang and Z. Chen, "A framework for state-of-charge and remaining discharge time prediction using unscented particle filter," *Applied Energy*, vol. 260, p. 114324, 2020.

[21] M. Yu, C. Liu, B. Li, and W.-H. Chen, "An enhanced particle filtering method for gmti radar tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 3, pp. 1408–1420, 2016.

[22] K. Romeo, P. Willett, and Y. Bar-Shalom, "Particle filter tracking for banana and contact lens problems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 2, pp. 1098–1110, 2015.

[23] Y. Li and M. Coates, "Particle filtering with invertible particle flow," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 4102–4116, 2017.

[24] F. Daum and J. Huang, "Particle flow for nonlinear filters with log-homotopy," in *Signal and Data Processing of Small Targets 2008* (O. E. Drummond, ed.), vol. 6969, p. 696918, International Society for Optics and Photonics, SPIE, 2008.

[25] M. D. Hoffman and A. Gelman, "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo," *J. Mach. Learn. Res.*, vol. 15, p. 1593–1623, Jan. 2014.

[26] G. Graditi, R. Ciavarella, M. Valenti, A. Bracale, and P. Caramia, "Advanced Forecasting Method to the Optimal Management of a DC Microgrid in Presence of Uncertain Generation," in *2015 International Conference on Renewable Energy Research and Applications (ICRERA)*, pp. 1586–1590, 2015.

[27] M. Nishio and A. Arakawa, "Performance of Hamiltonian Monte Carlo and No-U-Turn Sampler for Estimating Genetic Parameters and Breeding Values," *Genetics, Selection, Evolution : GSE*, vol. 51, 2019.

[28] T. Wang, Z. Liu, and N. Mrad, "A Probabilistic Framework for Remaining Useful Life Prediction of Bearings," *IEEE Transactions on Instrumentation and Measurement*, pp. 1–1, 2020.

[29] F. Septier and G. W. Peters, "Langevin and Hamiltonian Based Sequential mcmc for Efficient Bayesian Filtering in High-Dimensional Spaces," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 312–327, 2016.

[30] P. Pileggi, M. Bocquel, and M. Podt, "Integrated Processing for Extended Target Tracking Using Sequential Hamiltonian Monte Carlo," in *2017 20th International Conference on Information Fusion (Fusion)*, pp. 1–8, 2017.

[31] R. Daviet, "Inference with hamiltonian sequential monte carlo simulators," *SSRN Electronic Journal*, 01 2016.

[32] A. Doucet and S. Sénécal, "Fixed-Lag Sequential Monte Carlo," in *2004 12th European Signal Processing Conference*, pp. 861–864, 2004.

[33] A. Doucet, M. Briers, and S. Sénécal, "Efficient Block Sampling Strategies for Sequential Monte Carlo Methods," *Journal of Computational and Graphical Statistics*, vol. 15, no. 3, pp. 693–711, 2006.

[34] H. Ruzayqat, A. Er-raiy, A. Beskos, D. Crisan, A. Jasra, and N. Kantas, "A lagged particle filter for stable filtering of certain high-dimensional state-space models," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 10, no. 3, pp. 1130–1161, 2022.

[35] M. Daigle and K. Goebel, "Model-based prognostics with fixed-lag particle filters," in *Annual Conference of the PHM Society*, vol. 1, 2009.

[36] E. Rakhimberdiev, D. Winkler, E. Bridge, N. Seavy, D. Sheldon, T. Piersma, and A. Saveliev, "A hidden markov model for reconstructing animal paths from solar geolocation loggers using templates for light intensity," *Movement Ecology*, vol. 3, 10 2015.

[37] R. K. Tiwari, S. Bhaumik, P. Date, and T. Kirubarajan, "Particle Filter for Randomly Delayed Measurements With Unknown Latency Probability," *Sensors*, vol. 20, no. 19, 2020.

[38] A. M. Johansen, "On blocks, tempering and particle mcmc for systems identification," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 969–974, 2015. 17th IFAC Symposium on System Identification SYSID 2015.

[39] M. Scharth and R. Kohn, "Particle efficient importance sampling," *Journal of Econometrics*, vol. 190, no. 1, pp. 133–147, 2016.

[40] D. Crisan and A. Doucet, "A survey of convergence results on particle filtering methods for practitioners," *IEEE Transactions on signal processing*, vol. 50, no. 3, pp. 736–746, 2002.

[41] A. Varsi, *Streaming Multi-core Sample-based Bayesian Analysis*. PhD thesis, University of Liverpool, 2021.

[42] S. Särkkä, *Bayesian Filtering and Smoothing*. 10 2013.

[43] M. Betancourt, "A conceptual introduction to hamiltonian monte carlo," 2017.

[44] M. Hirt, M. Titsias, and P. Dellaportas, "Entropy-based adaptive hamiltonian monte carlo," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28482–28495, 2021.

[45] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell, "Stan: A probabilistic programming language," *Journal of Statistical Software*, vol. 76, no. 1, p. 1–32, 2017.

[46] A. Buchholz, N. Chopin, and P. E. Jacob, "Adaptive Tuning of Hamiltonian Monte Carlo Within Sequential Monte Carlo," *Bayesian Analysis*, vol. 16, no. 3, pp. 745 – 771, 2021.

[47] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for optimization*. Mit Press, 2019.

[48] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, A. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large scale distributed deep networks," *Advances in neural information processing systems*, 10 2012.

[49] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, Oct. 2014.

[50] K. Donghwan and J. Fessler, "Optimized first-order methods for smooth convex minimization," *Mathematical Programming*, vol. 159, 2016.

[51] Y. Drori, "The exact information-based complexity of smooth convex minimization," *Journal of Complexity*, vol. 39, pp. 1–16, 2017.

[52] L. Zhang, B. Carpenter, A. Gelman, and A. Vehtari, "Pathfinder: Parallel quasi-newton variational inference," *Journal of Machine Learning Research*, vol. 23, no. 306, pp. 1–49, 2022.

[53] P. Green, L. Devlin, R. Moore, R. Jackson, J. Li, and S. Maskell, "Increasing the efficiency of sequential monte carlo samplers through the use of approximately optimal l-kernels," *Mechanical Systems and Signal Processing*, vol. 162, p. 108028, 2022.

[54] N. Whiteley, S. Singh, and S. Godsill, "Auxiliary particle implementation of probability hypothesis density filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, no. 3, pp. 1437–1454, 2010.

[55] A. Varsi, J. Taylor, L. Kekempanos, E. P. Knapp, and S. Maskell, "A fast parallel particle filter for shared memory systems," *IEEE Signal Processing Letters*, vol. 27, pp. 1570–1574, 2020.

[56] A. Varsi, S. Maskell, and P. G. Spirakis, "An o (log2n) fully-balanced resampling algorithm for particle filters on distributed memory architectures," *Algorithms*, vol. 14, no. 12, p. 342, 2021.

[57] N. Chopin, P. E. Jacob, and O. Papaspiliopoulos, "SMC2: an efficient algorithm for sequential analysis of state space models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 75, no. 3, pp. 397–426, 2013.

[58] E. Hairer, C. Lubich, and G. Wanner, "Geometric numerical integration illustrated by the Störmer–Verlet method," *Acta Numerica*, vol. 12, p. 399–450, 2003.