BAYESIAN SEQUENTIAL LEARNING FOR HIDDEN SEMI-MARKOV MODELS

Patrick Aschermayr, Konstantinos Kalogeropoulos
Department of Statistics
LSE
p.aschermayr@lse.ac.uk; k.kalogeropoulos@lse.ac.uk

December 24, 2021

ABSTRACT

In this paper, we explore the sequential estimation setting of state space models, in particular the Hidden Semi-Markov Model (HSMM), a flexible extension of the well known Hidden Markov Model (HMM) that allows the underlying stochastic process to be a semi-Markov chain. HSMMs are typically less often used than their basic HMM counterpart due to the increased computational challenges to evaluate the likelihood function. Moreover, while both models are sequential in nature, parameter estimation is mainly conducted via batch estimation methods. A major motivation of this paper is thus to provide methods to estimate HSMMs (1) in a computationally feasible time and (2) in a sequential setting. We provide and verify an efficient computational scheme for Bayesian parameter estimation on HSMMs. Moreover, we demonstrate predictive model performance by estimating the parameters for a financial times series for HSMMs against other popular discrete State Space Model (SSM)s.

***K*eywords** Hidden semi-Markov Model, Hidden Markov Model, Markov Chain Monte Carlo, Particle Filter, Sequential Monte Carlo, Sequential Monte Carlo Squared, Sequential Parameter Estimation

1 Introduction

State Space Model (SSM) are a flexible class of models that are used in ecology, economics, finance, robotics and signal processing [Bulla and Bulla, 2006, Lindsten and Schön, 2013, Kastner, 2016, Chopin and Papaspiliopoulos, 2020, Corenflos et al., 2021], among others. SSMs can handle structural breaks, shifts, or time-varying parameters and still have an interpretable structure. Moreover, such models are generative, allow for multi step forecasting, imputing missing data, and account for non-equal time steps. However, analytical forms of the likelihood function are only available in special cases, and thus standard parameter optimization routines might be unfeasible. The major challenge in solving SSMs is thus the (in general) intractable likelihood function $p_\theta(e_{1:t}) = \int p_\theta(e_{1:t}, s_{1:t}) ds_{1:t}$, which integrates over the latent state trajectory. Throughout this paper, S_t is a latent/hidden variable at time point t while E_t denotes the observed data. Model parameter are denoted with $\theta \in \mathbb{R}^D,\ D \geq 1$.

A flexible discrete SSM that is of particular focus in this paper is known as Hidden Semi-Markov Model (HSMM). HSMMs have a flexible state duration distribution, well suited for processes that remain in any particular state for an extended period of time, and are a generalization from the well known basic Hidden Markov Model (HMM) introduced in Baum and Petrie [1966]. HSMMs have been used in ecology, epidemiology, finance [Bulla and Bulla, 2006, Pohle et al., 2021, Visani et al., 2021] and many other fields [Yu, 2016], but are typically used less often than their standard HMM counterparts because the likelihood function is much more costly to evaluate. In the HMM case, the likelihood is computable in $\mathcal{O}(K^2T)$, where K = number of latent states, T = number of data points, see Baum and Petrie [1966]. However, for the HSMM, this is a much more expensive operation of order $\mathcal{O}(K^2(d_{max} - d_{min})^2T)$, d_{min} = minimal state duration, d_{max} = maximal state duration, typically $(d_{max} - d_{min}) >> K$, described in more detail in section 2. We provide a particle filter implementation to obtain an unbiased estimate of the likelihood in the same computational complexity as for the basic HMM. This estimate is typically obtained much faster than exact evaluation.

Traditionally, Sequential Monte Carlo (SMC) samplers, such as Particle Filter (PF)s, have been used to estimate the underlying state sequence $p_\theta(s_{1:t} \mid e_{1:t})$ of SSMs, while standard Markov Chain Monte Carlo (MCMC) samplers are a popular Bayesian estimation method for all other continuous model parameter. Combining said methods, however, is becoming increasingly popular, see Daviet [2018] and Buchholz et al. [2020]. A method that jointly infers the latent state sequence and model parameter is known as Particle MCMC (PMCMC) [Andrieu et al., 2010]. A drawback of PMCMC is that, while this method is tailored towards sequential data, it is a batch estimation method. Once new data is obtained, parameter have to be estimated again. Given that SSMs are inherently of sequential order, this is a critical problem. A sequential estimation technique, in which previous parameter estimates can conveniently be reused, is known as Sequential Monte Carlo Squared (SMC2), is introduced in Chopin et al. [2013], an extension from their previous work in Chopin [2002]. Similar approaches include Fearnhead and Taylor [2010] and Crisan and Miguez [2017]. A review about all mentioned methods can be seen in chapter 3.

HSMMs are typically less often used than their basic HMM counterpart due to the increased computational challenges to evaluate the likelihood function. Moreover, while both models are sequential in nature, parameter estimation is mainly conducted via batch estimation methods. A major motivation of this paper is thus to provide methods to estimate HSMMs (1) in a computationally feasible time and (2) in a sequential setting. HSMMs are formally introduced in section 2. Section 3 describes all Bayesian inference techniques mentioned above in more detail, provides insight in tuning settings for said algorithm, and discusses several model selection criteria that come as byproduct from the estimation process. Chapter 4 validates all implementations of algorithms mentioned in the previous chapter by estimating model parameter using sample data from a known model. We also discuss the possibility of estimating the number of states using SMC2. Finally, section 5 discusses real world applications, and HSMMs and several discrete benchmark SSMs predictive performance is measured against each other on a financial data set.

The main contribution of this paper are as follows:

- First, we have a algorithmic contribution by proposing an efficient computational scheme to obtain an unbiased estimate for the likelihood function of an HSMM via a particle filter. This is particularly important in the sequential estimation context, where we can reuse information from previous iterations.Moreover, as can be seen in an example in section 3, it can also be advantageous to use particle based methods instead of standard exact filtering techniques for batch parameter estimation.
- Second, we have a methodological contribution by providing a flexible sequential estimation framework that is able to fit arbitrary forms of distributions for the observation as well as transition distributions of HSMMs. Both continuous and discrete SSMs can be fitted seamlessly.
- Third, we have a model contribution by combining a HSMM and eGARCH model to accurately mimic known stylized financial facts that are otherwise difficult to describe for standalone models. This model is described in detail in section 5, and has a superior forecasting performance against other popular discrete SSMs for both of our model evaluation criteria mention in section 3.

- Fourth, we have a computational contribution by open sourcing a modular estimation toolbox for other researcher to reproduce, adopt or further improve our research output. All models and algorithm routines in this paper are extensively validated on simulated and real world data. While there there are an increasing amount of excellent MCMC software solutions, such as Stan Development Team [2021], Ge et al. [2018], Salvatier et al. [2016], available, we fill a niche by focusing on sequential estimation and providing full support for SSMs with arbitrary model dynamics.

2 Hidden Semi-Markov Model

One of the most popular SSMs is formally known as a Hidden Markov Model, which can be defined as follows:

Definition 2.1. *Hidden Markov Model (HMM) A hidden Markov model is a bivariate stochastic process* $\{E_t, S_t\}_{t=1,2,...}$*, where* S_t *is an unobserved Markov chain and, conditional on* S_t*,* E_t *is an observed sequence of independent random variables. The model is fully specified by the transition distribution* f_θ*,* $S_t \sim f_\theta(s_t \mid s_{t-1})$*,* $t \geq 2$*, the corresponding initial distribution* π_θ*,* $S_1 \sim \pi_\theta(s_1)$*, and the observation distribution* g_θ*,* $E_t \sim g_\theta(e_t \mid s_t)$*,* $t \geq 1$*.*

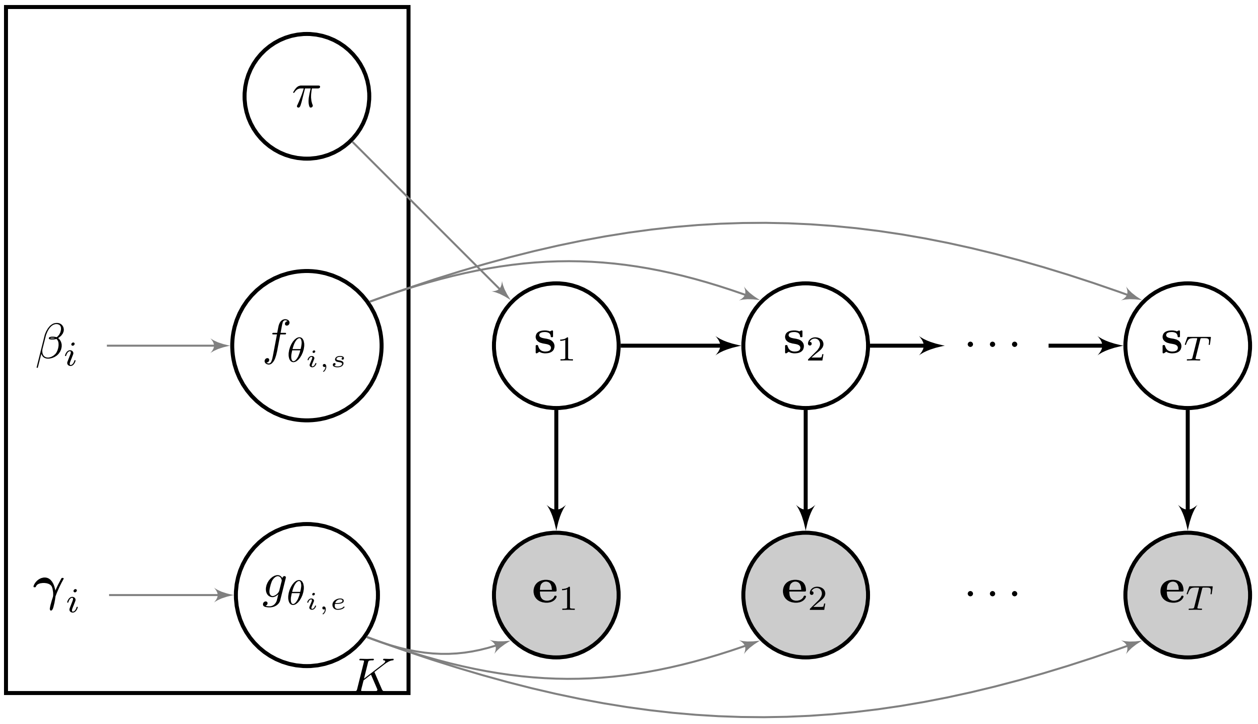

Figure 1: K-state Bayesian HMM, parameter θ and hyper-parameter $\{\beta, \gamma\}$. The shaded nodes e_t denote the observed data at time t, while the unshaded nodes indicate the latent state s_t. $\theta_{i,s}$ denotes the parameter at state i for latent state s, and β_i the corresponding hyper-parameter. f is the transition distribution s, g is the observation distribution for e. π denotes the initial distribution for s.

Naively computing the likelihood function involves summing up over all possible state sequences,

$$p(e_{1:T}) = \sum_{s_{1:T}} p(e_1 \mid s_1) p(s_1) \prod_{t=2}^{T} p(e_t \mid s_t) p(s_t \mid s_{t-1}). \tag{1}$$

Hence, various filtering techniques have been proposed that take into account the memory of the latent state variable to reduce the computational costs to $\mathcal{O}(K^2T)$, where K is the number of latent states and T the number of data points, see Baum and Petrie [1966]. Note that in the literature, filtering usually refers to targeting the marginals $p(s_t \mid e_{1:t})$. However, most filtering methods for state space models rely on an approximation of $p(s_{1:t} \mid e_{1:t})$ even when only the final time marginal is of interest, see, e.g., Kantas et al. [2015]. This is due to the difficulty associated with evaluating the integrals which appears in the marginals explicitly. However, once the joint distribution $p(s_{1:t} \mid e_{1:t})$ is obtained, the marginal $p(s_t \mid e_{1:t})$ can be retrieved by just dropping the previous terms. In the context of state space models, the goal is thus to target a sequence of increasing distributions

$$p(s_{1:t} \mid e_{1:t}) = \frac{p(s_{1:t}, e_{1:t})}{p(e_{1:t})} \tag{2}$$

The filtering recursion then becomes:

$$p(s_{1:t} \mid e_{1:t}) = \frac{P(s_1)\prod_{n=2}^{t} p(s_n \mid s_{n-1}) \prod_{n=1}^{t} p(e_n \mid s_n)}{\int p(s_{1:t}, e_{1:t}) ds_{1:t}} \tag{3}$$

Given above assumptions, we can proceed to write the filtering equations as recursions. The joint density can be written as

$$\begin{aligned} p(s_{1:t}, e_{1:t}) &= p(s_{1:t-1}, e_{1:t-1}, s_t, e_t) \\ &= p(s_{1:t-1}, e_{1:t-1}) p(s_t, e_t \mid s_{1:t-1}, e_{1:t-1}) \\ &= p(s_{1:t-1}, e_{1:t-1}) p(e_t \mid s_t, s_{1:t-1}, e_{1:t-1}) p(s_t \mid s_{1:t-1}, e_{1:t-1}) \\ &= p(s_{1:t-1}, e_{1:t-1}) p(e_t \mid s_t) p(s_t \mid s_{t-1}), \end{aligned} \tag{4}$$

hence

$$\begin{aligned} p(s_{1:t} \mid e_{1:t}) &= \frac{p(s_{1:t-1}, e_{1:t-1}) p(e_t \mid s_t) p(s_t \mid s_{t-1})}{p(e_t \mid e_{1:t-1}) p(e_{1:t-1})} \\ &= p(s_{1:t} \mid e_{1:t-1}) \frac{p(e_t \mid s_t) p(s_t \mid s_{t-1})}{p(e_t \mid e_{1:t-1})}, \end{aligned} \tag{5}$$

where we depict the denominator as

$$\begin{aligned} p(e_t \mid e_{1:t-1}) &= \int p(e_t, s_{t-1:t} \mid e_{1:t-1}) ds_{t-1:t} \\ &= \int p(e_t \mid e_{1:t-1}, s_{t-1:t}) p(s_{t-1:t} \mid e_{1:t-1}) ds_{t-1:t} \\ &= \int p(e_t \mid s_t) p(s_t \mid s_{t-1}) p(s_{t-1} \mid e_{1:t-1}) ds_{t-1:t}. \end{aligned} \tag{6}$$

The likelihood can then easily calculated from the filtering equations, $p(e_t \mid e_{1:t-1}) = p(e_1) \prod_{n=2}^{t} p(e_n \mid e_{1:n-1})$ as we have

$$\begin{aligned} p(e_{1:t}) &= p(e_t, e_{1:t-1}) \\ &= p(e_t \mid e_{1:t-1}) p(e_{1:t-1}) \end{aligned} \tag{7}$$

For discrete state space models, one can alternatively write equation 1 as sum over a joint probability,

$$\begin{aligned} p(e_{1:T}; \theta) &= \sum_k p(e_{1:T}, S_T = k; \theta) \\ &= \sum_k \alpha_T(k) \end{aligned} \tag{8}$$

The alpha term in equation 8 is usually called Forward probability, and can be iteratively calculated. At any time point t,

$$\begin{aligned}
\alpha_t(k) &= \sum_j p(e_{1:t}, S_t = k, S_{t-1} = j) \\
&= \sum_j p(e_t, e_{1:t-1}, S_t = k, S_{t-1} = j) \\
&= \sum_j p(e_t, S_t = k \mid e_{1:t-1}, S_{t-1} = j) p(e_{1:t-1}, S_{t-1} = j) \\
&= \sum_j p(e_t \mid S_t = k, e_{1:t-1}, S_{t-1} = j) p(S_t = k \mid e_{1:t-1}, S_{t-1} = j) p(e_{1:t-1}, S_{t-1} = j) \\
&= \sum_j p(e_t \mid S_t = k) p(S_t = k \mid S_{t-1} = j) p(e_{1:t-1}, S_{t-1} = j) \\
&= p(e_t \mid S_t = k) \sum_j p(S_t = k \mid S_{t-1} = j) \alpha_{t-1}(j),
\end{aligned} \tag{9}$$

where we start with $\alpha_1(k) = p(e_1, S_1 = k) = p(e_1 \mid S_1 = k)p(S_1 = k)$. An interesting fact is that this filtering process can be run online, or recursively, as new data comes in. Hence, the algorithm can be used to efficiently evaluate the likelihood of a basic HMM given current parameter θ. In case one needs to additionally sample from $p(s_{1:t} \mid e_{1:t})$, one typically uses 'smoothed' probabilities, $p(S_t = k \mid e_{1:T})$, instead of the filtered equations above. To do so, one can solve this via two steps. First, calculate

$$\begin{aligned}
p(S_t = k, e_{1:T}) &= p(S_t = k, e_{1:t}, e_{t+1:T}) \\
&= p(e_{t+1:T} \mid S_t = k, e_{1:t}) p(S_t = k, e_{1:t}) \\
&= p(e_{t+1:T} \mid S_t = k) p(S_t = k, e_{1:t}) \\
&= \beta_t(k) \alpha_t(k),
\end{aligned} \tag{10}$$

where $\beta_t(k) = p(e_{t+1:T} \mid S_t = k)$ starts at $\beta_T(k) = 1 \ \forall k$ as there is no observation at time $T+1$. The other terms of β can also be calculated iteratively

$$\begin{aligned}
\beta_t(k) &= \sum_j p(e_{t+1}, e_{t+2:T}, S_{t+1} = j \mid S_t = k) \\
&= \sum_j p(e_{t+2:T} \mid e_{t+1}, S_t = k, S_{t+1} = j) p(e_{t+1}, S_{t+1} = j \mid S_t = k) \\
&= \sum_j p(e_{t+2:T} \mid e_{t+1}, S_t = k, S_{t+1} = j) p(S_{t+1} = j \mid S_t = k) p(e_{t+1} \mid S_t = k, S_{t+1} = j) \\
&= \sum_j p(e_{t+2:T} \mid S_{t+1} = j) p(S_{t+1} = j \mid S_t = k) p(e_{t+1} \mid S_{t+1} = j).
\end{aligned} \tag{11}$$

The smoothed probabilities can then easily be calculated via

$$\begin{aligned}
p(S_t = k \mid e_{1:T}) &= \frac{p(S_t = k, e_{1:T})}{p(e_{1:T})} \\
&= \frac{\alpha_t(k)\beta_t(k)}{\sum_i \alpha_t(i)\beta_t(i)}.
\end{aligned} \tag{12}$$

Note that $\sum_i \alpha_t(i)\beta_t(i) = p(e_{1:T;\theta})$, the likelihood of the model, will be the same regardless of which time step t one chooses. A major criticism of basic HMMs is the inability of the model to stay for a prolonged time in any particular state. To give insight into this issue, define $p(S_{t+k} = j,\ S_{t+1:t+k-1} = i \mid s_t = i)$, the probability a state remains in any current state until it switches, which is known as state duration distribution. In the basic HMM case, this probability is implicitly geometric. Set $p(s_t = i \mid s_{t-1} = i) = \mathcal{T}_{ii}$, and assume there are only 2 states, then for a homogeneous Markov chain, using the chain rule and the Markov assumption, it holds:

$$\begin{aligned}
p(S_{t+3} = j, S_{t+2} = i, S_{t+1} = i \mid S_t = i) &= p(S_{t+3} = j \mid S_{t+2} = i) p(S_{t+2} = i, \mid S_{t+1} = i) p(S_{t+1} = i \mid S_t = i) \\
&= (1 - \mathcal{T}_{ii}) * \mathcal{T}_{ii}^2
\end{aligned} \tag{13}$$

In general, for $t + k$ steps, it holds that

$$\begin{aligned} p(S_{t+k} = j, \ldots, S_{t+1} = i \mid S_t = i) &= (1 - \mathcal{T}_{ii}) * \mathcal{T}_{ii}^{k-1} \\ &= Geometric_{\mathcal{T}_{ii}}, \end{aligned} \tag{14}$$

where the geometric distribution has to be interpreted as the length of state duration up to and including the transition to the other state. For processes that tend to stay in any particular state for a long time horizon, this may be a poor modelling choice. One could, alternatively, explicitly model the state duration distribution. A Hidden Semi-Markov Model, see [Murphy, 2002, Yu, 2010, 2016], is generalization of an HMM and equivalent to one if one chooses a geometric distribution as state duration distribution. A graph structure and a comparison the the standard HMM can be seen in the upper figures 1 and 2, and a pseudo algorithm to forward sample in algorithm 1. One particular form of HSMM is known as Explicit-duration Hidden Markov Model (EDHMM), a HSMM that explicitly defines the duration distribution. Transitions are allowed only at the end of each state, resulting in the following definition:

Definition 2.2. *Hidden semi-Markov Model (HSMM) A hidden semi-Markov model is a bivariate stochastic process* $\{E_t, Z_t\}_{t=1,2,\ldots}$, *where* $Z_t = \{S_t, D_t\}$ *is an unobserved semi-Markov chain and, conditional on* Z_t, E_t *is an observed sequence of independent random variables. The model is fully specified by the transition distribution* $f_\theta(s_t \mid s_{t-1}, d_{t-1})$ *of* S_t

$$S_t \sim \begin{cases} \delta(s_t, s_{t-1}) & d_{t-1} > 0 \\ f_\theta(s_t \mid s_{t-1}, d_{t-1}) & d_{t-1} = 0\ , \end{cases} \tag{15}$$

the duration distribution h_θ *of* D_t

$$D_t \sim \begin{cases} \delta(d_t, d_{t-1} - 1) & d_{t-1} > 0 \\ h_\theta(d_t \mid s_t, d_{t-1}) & d_{t-1} = 0\ , \end{cases} \tag{16}$$

the corresponding initial distribution π_θ *of* Z_t, *and the observation distribution* g_θ, $E_t \sim g_\theta(e_t \mid s_t)$,

$$E_t \sim g_\theta(e_t \mid s_t). \tag{17}$$

where $\delta(a, b)$ *is the Kronecker product and equals* 1 *if* $a = b$ *and* 0 *otherwise.*

Popular choices for the duration distribution are, for instance, the Poisson distribution or the Negative Binomial distribution for greater flexibility at the cost of an additional model parameter. The joint distribution of an EDHMM given the parameter can be stated as

$$p_\theta(s_{1:T}, d_{1:T}, e_{1:T}) = \pi_\theta(s_1)\pi_\theta(d_1)g_\theta(e_1 \mid s_1) \prod_{t=2}^{T} f_\theta(s_t \mid s_{t-1}, d_{t-1})h_\theta(d_t \mid s_t, d_{t-1})g_\theta(e_t \mid s_t) \tag{18}$$

One can analytically obtain the likelihood by integrating out both $s_{1:T}$ and $d_{1:T}$. Due to the additional latent variable this is a much more expensive operation than in the standard HMM case. In order to gain more insight on this, we can shrink the graphical model structure of a HMM and HSMM to a single time step, shown in the lower figures 2 and 2. In order to compute the likelihood of observation e_{t+1}, one has to sum out all possible state transitions, as can be seen in equation 19. This looks similar to a standard mixture model computation, with the transition matrix of the HMM replacing the mixture component weights.

$$\begin{aligned} p(e_{t+1} \mid s_t = k) &= \sum_{s_{t+1}} p(e_{t+1}, s_{t+1} \mid s_t = k) \\ &= \sum_{s_{t+1}} p(s_{t+1} \mid s_t = k) p(e_{t+1} \mid s_{t+1}) \end{aligned} \tag{19}$$

For the HSMM, however, the duration variable means an additional sum over a random variable that has at worst an infinite number of terms in the case of duration distributions with countably infinite support, shown in equation 20.

$$\begin{aligned} p(e_{t+1} \mid s_t = k, d_t = j) &= \sum_{s_{t+1}} \sum_{d_{t+1}} p(e_{t+1}, s_{t+1}, d_{t+1} \mid s_t = k, d_t = j) \\ &= \sum_{s_{t+1}} \sum_{d_{t+1}} p(s_{t+1} \mid s_t = k, d_t = j) p(d_{t+1} \mid s_{t+1}, d_t = j) p(e_{t+1} \mid s_{t+1}) \end{aligned} \tag{20}$$

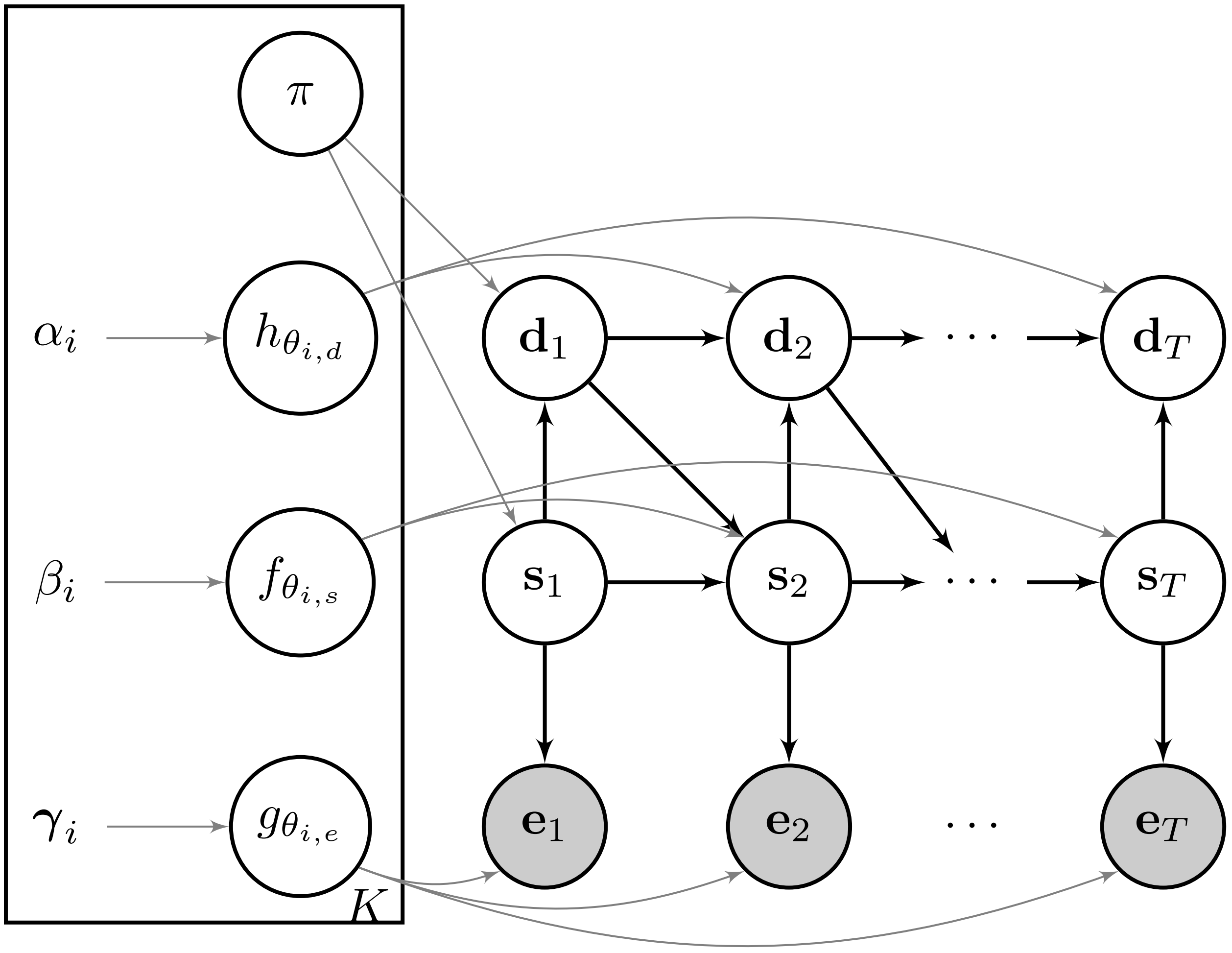

Figure 2: K-state Bayesian HSMM, parameter θ and hyper-parameter $\{\alpha, \beta, \gamma\}$. The shaded nodes e_t denote the observed data at time t, while the unshaded nodes indicate latent duration d_t and state s_t. $\theta_{i,d}$ denotes the parameter at state i for latent duration d, and α_i the corresponding hyper-parameter. h and f are the transition distributions for d and s, g is the observation distribution for e. π denotes the initial distribution for d and s.

Similar to the basic HMM case, one can iteratively define the α and β terms, see Rabiner [1989]. Set $Z_t = \{S_t, D_t\}$, then the forward filtering equations can be stated as

$$\alpha_t(z_t) = p(e_t \mid z_t) \sum_{z_{t-1}} p(z_t \mid z_{t-1}), \tag{21}$$

where $p(z_t \mid z_{t-1}) = p(s_t \mid s_{t-1}, d_{t-1})p(d_t \mid d_{t-1}, s_t)$. However, note that $\sum_{z_{t-1}} = \sum_{d_{t-1}} \sum_{s_{t-1}}$, which sums up all possible durations over all states, has at worst an infinite number of terms if the duration distributions have countably infinite support, and at best a large number of terms for long sequences, see Dewar et al. [2012]. The standard approach is to set up a minimum and maximum duration d_{min} and d_{max}, where the computational complexity of the forward-backward algorithm reduces to $O(T(K(d_{max} - d_{min})^2)$, compared to the original $O(TK^2)$ in the basic HMM (for each iteration in a MCMC step). If the truncation is too small, then inference will typically fail, if is too large then the problem might become computationally infeasable. Hence, $(d_{max} - d_{min})$ may increase the computational complexity to burdensome levels, or requires the modeler to set d_{max} too small. Other approaches include [Johnson and Willsky, 2013, Johnson, 2014], who can decrease computational complexity by censoring the initial or end time.

To sum up, choosing an appropriate maximum duration varies depending on the underlying data. In order to appropriately estimate a HSMM, we assume that at least a single (possibly much more) state transition has to occur, hence $d_{max} < T$, but $(d_{max} - d_{min}) >> K$. Naively extending the sequential likelihood computation from a single data point to T data points would cause to take into account all possible state transitions, but taking into ac-

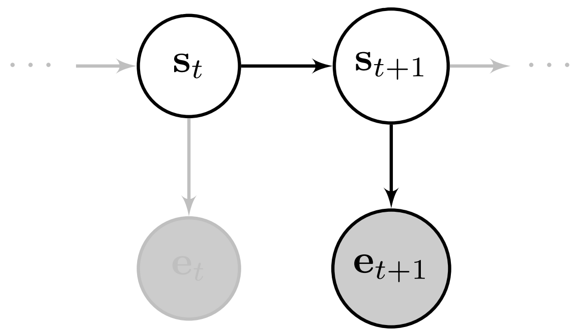
$$\begin{tikzcd} \cdots \arrow[r] & \mathbf{s}_t \arrow[r] \arrow[d] & \mathbf{s}_{t+1} \arrow[r] \arrow[d] & \cdots \\ & \mathbf{e}_t & \mathbf{e}_{t+1} & \end{tikzcd}$$

Figure 3: HMM graph: s denotes the latent state, e denotes the observed observation at each time point.

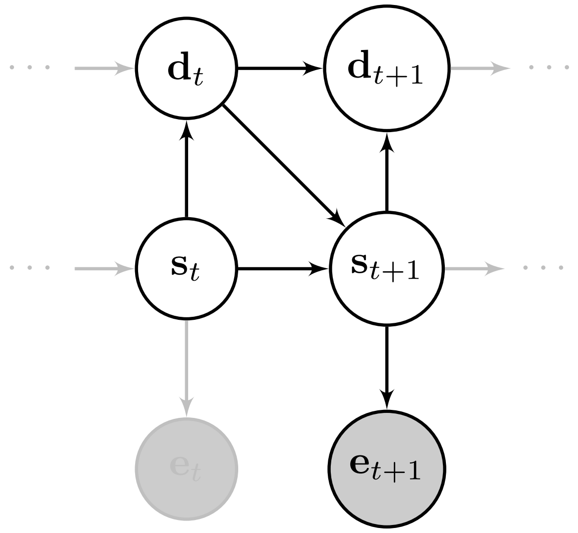
$$\begin{tikzcd} \cdots \arrow[r] & \mathbf{d}_t \arrow[r] \arrow[dr] & \mathbf{d}_{t+1} \arrow[r] & \cdots \\ \cdots \arrow[r] & \mathbf{s}_t \arrow[r] \arrow[u] \arrow[d] & \mathbf{s}_{t+1} \arrow[r] \arrow[u] \arrow[d] & \cdots \\ & \mathbf{e}_t & \mathbf{e}_{t+1} & \end{tikzcd}$$

Figure 4: HSMM graph: the upper random variable d denotes the remaining time spent in any particular hidden state s. e denotes the observed observation at each time point.

count the memory properties simplifies the computational costs for the basic HMM to $\mathcal{O}(K^2T)$ and for the HSMM to $\mathcal{O}(K^2(d_{max} - d_{min})^2T)$, see, e.g., Baum and Petrie [1966], Murphy [2002]. For example, the computational costs for a 5-state HMM for 1000 data points would be $\mathcal{O}(5^2 \times 1000)$, while for the HSMM, assuming $d_{max} = 500$ and $d_{min} = 0$, $\mathcal{O}(5^2 \times 500^2 \times 1000)$. An alternative is to approximate the likelihood function. For example, an unbiased particle filter estimate can be obtained in $\mathcal{O}(NT)$ operations, independent if the model has HMM or HSMM dynamics. It is a priori unclear if $\mathcal{O}(NT)$ is drastically smaller than $\mathcal{O}(K^2(d_{max} - d_{min})^2T)$ for small K. However, as can be seen in section 3 and 4, setting $N = \frac{T}{10}$ and $N = \frac{T}{2}$ is more than sufficient to obtain stable estimates for the likelihood. Moreover, the particle filter does not need to set d_{max} or d_{min}. Consequently, even for standard batch parameter estimation methods, the particle filter is better suited than most exact filtering techniques. Moreover, another major advantage that the particle filter has is that we can update the likelihood estimate online in case parameter are fixed.

3 Bayesian Inference on State Space Models

In a Bayesian framework, given the latent states $s_{1:T}$ and observed data $e_{1:T}$, the goal is to infer the full posterior distribution

$$p(s_{1:T}, \theta \mid e_{1:T}) = \frac{p_\theta(e_{1:T} \mid s_{1:T})\, p_\theta(s_{1:T})\, p(\theta)}{p(e_{1:T})}. \tag{22}$$

Likewise, one may target the marginal posterior distribution

$$p(\theta \mid e_{1:T}) = \frac{p_\theta(e_{1:T})\, p(\theta)}{p(e_{1:T})}. \tag{23}$$

However, this involves computing the marginal likelihood $p_\theta(e_{1:t}) = \int p_\theta(e_{1:T}, s_{1:T})\, ds_{1:T}$,which is typically intractable or costly to evaluate. If $S_{1:T}$ is continuous, we can theoretically target $p(s_{1:t}, \theta \mid y_{1:t}) \propto p(y_{1:t} \mid s_{1:t}, \theta) \times p(s_{1:t} \mid \theta) \times p(\theta)$ via MCMC. However, we would need to sample a state trajectory of $p(s_{1:T} \mid \theta)$ at the same time as we do inference, usually resulting in very poor outcomes of this strategy. A more general attempt to jointly target the full joint posterior $p(s_{1:T}, \theta \mid e_{1:T})$ can be shown as follows:

- 1. propose $\theta^\star \sim f(\theta^\star \mid \theta)$,
- 2. propose $S^\star_{1:T} \sim p_{\theta^\star}(s^\star_{1:T} \mid e_{1:T})$,

- 3. accept $(\theta^\star, s^\star_{1:T})$ with acceptance probability

$$\begin{aligned} a((s^\star_{1:T}, \theta^\star), (s_{1:T}, \theta)) &= \frac{p_{\theta^\star}(s^\star_{1:T})}{p_\theta(s_{1:T})} \frac{p_{\theta^\star}(e_{1:T} \mid s^\star_{1:T})}{p_\theta(e_{1:T} \mid s_{1:T})} \frac{p_\theta(s_{1:T} \mid e_{1:T})}{p_{\theta^\star}(s^\star_{1:T} \mid e_{1:T})} \frac{p(\theta^\star)}{P(\theta)} \frac{q(\theta \mid \theta^\star)}{q(\theta^\star \mid \theta)} \\ &= \frac{p_{\theta^\star}(e_{1:T})}{p_\theta(e_{1:T})} \frac{p(\theta^\star)}{P(\theta)} \frac{q(\theta \mid \theta^\star)}{q(\theta^\star \mid \theta)}. \end{aligned} \tag{24}$$

The first three terms in equation 24 can be seen as the prior, the likelihood and the posterior of $S_{1:T}$ for a given θ and data $e_{1:T}$. The whole equation can be simplified by using the Basic Marginal Likelihood Identity (BMI) of Chib [1995]. This framework allows to jointly sample θ and $S_{1:t}$. However, the in general intractable likelihood term is still contained in step 3, and with this all the problems mentioned before. Particle MCMC (PMCMC) [Andrieu et al., 2010] is an inference algorithm where the likelihood in step 3 is replaced with an unbiased estimate $\hat{p}_\theta(e_{1:T})$ from a PF, thereby circumventing the need to be able to evaluate the likelihood function of the State Space Model of choice. Alternative schemes use beam sampling, see Dewar et al. [2012], or define custom Gibbs sampler, see Johnson and Willsky [2013], in this step, but we will see major advantages of the particle filter usage in a sequential estimation context later on.

3.1 Particle Filtering

Particle Filters are usually used to solve filtering equations in the form of

$$\pi_t(x_{1:t}) = \frac{\tau_t(x_{1:t})}{Z_t}. \tag{25}$$

The goal is to sequentially sample a sequence of random variables, $X_t, t \in (1, ..., T)$ that come from a sequence of target probabilities $\pi_t(x_{1:t})$ with the same computational complexity at each time step. If we cannot sample from π_t, we can instead use the proposal distribution q_t, s.t. $\pi_t(x_{1:t}) > 0 \Rightarrow q_t(x_{1:t}) > 0$. In this case, we have to introduce an un-normalized weight function

$$w_t(x_{1:t}) = \frac{\tau_t(x_{1:t})}{q_t(x_{1:t})}, \tag{26}$$

s.t. we can rewrite the target as:

$$\pi_t(x_{1:t}) = \frac{w_t(x_{1:t}) q_t(x_{1:t})}{Z_t}. \tag{27}$$

This method is known as Importance Sampling (IS). Often, the variable of interest is the normalizing constant $Z_t = \int \tau_t(x_{1:t}) dx_{1:t} = \int w_t(x_{1:t}) q_t(x_{1:t}) dx_{1:t}$, which we can approximate via the un-normalized weight functions:

$$\hat{Z}_t = \frac{1}{t} \sum_{i=1}^{K} \frac{\tau_t(x_{1:t})}{q_t(x_{1:t})} = \frac{1}{t} \sum_{i=1}^{K} w_t(x^i_{1:t}). \tag{28}$$

Note that the integrator $q(x)$ here is replaced by the sum. How to decide which q to choose? One could try to minimize the the variance for the Monte Carlo estimator, which can be stated as:

$$Var(\hat{Z}_t) = \frac{Z_t^2}{N} \Big(\int \frac{\pi_t(x_{1:t})^2}{q_t(x_{1:t})} dx_{1:t} - 1 \Big), \tag{29}$$

see, e.g., Doucet and Johansen [2011]. If we set $q_t(x_{1:t}) = \pi_t(x_{1:t})$, then the integral in the brackets integrates to 1, which results in 0 variance. One can also show that one can equivalently minimize the variance of the corresponding importance weights, which also results in $q_t(x_{1:t}) = \pi_t(x_{1:t})$, which obviously cannot be done as it is the reason we apply IS in the first place, but provides some insight: choose a q as close to π as possible. This method does, unfortunately, run into problems as t becomes larger. A method to sequentially sample from such distributions is called Sequential Importance Sampling (SIS), a method to keep the computational cost fixed given additional time steps t. One can decompose the joint distribution $\tau_t(x_{1:t}) = \tau_{t-1}(x_{1:t-1}) \tau_t(x_t \mid x_{1:t-1})$. Similarly, you can select an importance distribution that can be decomposed as follows:

$$q_t(x_{1:t}) = q_1(x_1) \prod_{n=2}^{t} q_n(x_n \mid x_{1:n-1}), \tag{30}$$

In most state space models, the memory for $q_t(x_t \mid x_{1:t-1})$ is usually only up to $t-1$, so for each additional time step t you can just sample $q_t(x_t \mid x_{t-1})$ at fixed computational costs, and then apply the equations above to get the desired quantity. The associated un-normalized weights can then be computed recursively via:

$$\begin{aligned} w_t(x_{1:t}) &= \frac{\tau_t(x_{1:t})}{q_t(x_{1:t})} \\ &= \frac{\tau_{t-1}(x_{1:t-1})}{q_{t-1}(x_{1:t-1})} \frac{\tau_t(x_t \mid x_{1:t-1})}{q_t(x_t \mid x_{1:t-1})} \\ &= w_{t-1}(x_{1:t-1}) \alpha_t(x_{1:t}) \\ &= w_1(x_1) \prod_{k=1}^{t} \alpha_k(x_{1:k}), \end{aligned} \tag{31}$$

where the incremental importance weight $\alpha_t(x_{1:t})$ are given as

$$\alpha_t(x_{1:t}) = \frac{\tau_t(x_t \mid x_{1:t-1})}{q_t(x_t \mid x_{1:t-1})}. \tag{32}$$

The only freedom in this framework is choosing an appropriate q_t. As already mentioned, optimal - in terms of minimizing weight variance, would be the unfeasible: $q_t^{opt}(x_t \mid x_{1:t-1}) = \pi_t(x_t \mid x_{1:t-1})$. Note that, in sequential problems, this formulation would let the computational time complexity grow with t, but all the most common SSMs have limited memory, hence the computational costs remain fixed per additional time step.

For a given θ, SIS algorithms approximate $\pi_\theta(x_{1:t})$ by propagating forward a set of particles over time t according to the proposal q, with a fixed computational costs according to the memory properties of the corresponding state space model. Unfortunately, it can be shown that the variance of the corresponding weights grows with t, and refer to this problem as weight degeneracy. This is problematic as it increases the variance of the estimator of the normalizing constant Z. Hence, usually a resampling step for the particle trajectory is applied, which normalizes the corresponding particle weights. Algorithms that resample the particle trajectories at each iteration are referred as Sequential Importance Resampling (SIR). However, resampling at each iteration also creates the problem of sample degeneracy. Balancing weight and sample path degeneracy is an ongoing research topic, and the most common method to do so is to resample trajectories only if certain thresholds are reached, for example, if the effective sample size of the particles is less than an a priori number. For a discussion on several different resampling techniques, see Douc and Cappe [2005]. Adaptive resampling mitigates the exploding variance of the particle weights and keeps sample path degeneracy in check.

Such algorithms are referred to as Particle Filters. This algorithm proceeds sequentially by propagating forward a set of particles through the set of observations by a series of propagation, weighting and adaptive resampling steps with constant computational costs per time step. This procedure has a fixed computational complexity that is both linear in time T and in number of particles N, $\mathcal{O}(NT)$, and returns an estimate of the normalizing constant $\hat{Z}_t$ and a particle path of $\hat{\pi}(x_{1:t})$. While $\hat{Z}_t$ is only an unbiased estimate, the computational costs are fixed for all state space models used in this paper, even if the term may not be evaluated analytically.

In the SSM case, the joint distribution and the normalizing constant are of the form $\tau_t(x_{1:t}) = p_\theta(s_{1:t}, e_{1:t})$ and $Z = p_\theta(e_{1:t})$. An approximation for the marginal likelihood can be computed via the weights $\hat{Z} = \frac{1}{N}\sum_{n=1}^{N} w_t(s_{1:t}^n, e_{1:t})$, which can be decomposed in the following recursive form:

$$\begin{aligned} w_t(s_{1:t}, e_{1:t}) &= \frac{p_\theta(s_{1:t}, e_{1:t})}{q_t(s_{1:t} \mid e_{1:t})} \\ &= \frac{p_\theta(s_{1:t-1}, e_{1:t-1})}{q(s_{1:t-1} \mid e_{1:t-1})} \frac{p_\theta(e_t \mid s_{1:t}, e_{1:t-1})\, p_\theta(s_t \mid s_{1:t-1}, e_{1:t-1})}{q(s_t \mid s_{1:t-1}, e_{1:t})} \\ &= w_{t-1}(s_{1:t-1}, e_{1:t-1}) \frac{p_\theta(e_t \mid s_{1:t}, e_{1:t-1})\, p_\theta(s_t \mid s_{1:t-1}, e_{1:t-1})}{q(s_t \mid s_{1:t-1}, e_{1:t})} \\ &= w_1(s_1, e_1) \prod_{k=1}^{t} \alpha_k(s_{1:k}, e_{1:k}). \end{aligned} \tag{33}$$

The incremental weight α is defined as

$$\alpha_t(s_{1:t}, e_{1:t}) = \frac{p_\theta(e_t \mid s_{1:t}, e_{1:t-1})\, p_\theta(s_t \mid s_{1:t-1}, e_{1:t-1})}{q(s_t \mid s_{1:t-1}, e_{1:t})}. \tag{34}$$

We can also express the likelihood estimate via the incremental weights as $\hat{Z} = \frac{1}{N}\sum_{n=1}^{N}\prod_{k=1}^{T}\alpha_k(s_{1:k}^n, e_{1:k})$, which is usually the preferred method in particle filter implementation as this avoids memory allocations. $p_\theta(e_t \mid s_{1:t}, e_{1:t-1})$ and $p_\theta(s_t \mid s_{1:t-1}, e_{1:t-1})$ are model distributions, and - depending on the SSM - have usually limited memory. The only free distribution to choose is $q(s_t \mid s_{1:t-1}, e_{1:t})$, which should ideally look like $p_\theta(s_t \mid s_{1:t-1}, e_{1:t})$. A common and simple choice is known as Bootstrap Particle Filter (BPF), which takes $q(s_t \mid s_{1:t-1}, e_{1:t}) = p_\theta(s_t \mid s_{1:t-1}, e_{1:t-1})$, reducing the incremental weight to $\alpha_t(s_{1:t}, e_{1:t}) = p_\theta(e_t \mid s_{1:t}, e_{1:t-1})$. Another popular approach is the so called Auxiliary Particle Filter (APF) [Pitt and Shephard, 1999], which assumes $\alpha_t(s_{1:t}, e_{1:t})$ to be independent of s_t (in the Bootstrap Particle Filter, this does not hold!). In this case, one can interchange the sampling and re-sampling process, which yields a better approximation of the distribution as it provides a greater number of distinct particles to approximate the target distribution. Pitt and Shephard [1999] thus argue that resampling, if it is to be applied in a particular iteration, should be performed before, rather than after, any operation that doesn't influence the importance weights in order to minimise the loss of information. The author's originally suggested Auxiliary Particle Filter essentially does this by using an additional auxiliary variable (hence the name). However, improved versions without the additional auxiliary variable have been suggested since then, see Carpenter et al. [2000]. We can gain more insight by choosing (the unattainable) $p_\theta(s_t \mid s_{1:t-1}, e_{1:t})$ as proposal, then the incremental weight becomes:

$$\begin{aligned}
\alpha_t(s_{1:t}, e_{1:t}) &= \frac{p_\theta(e_t \mid s_{1:t}, e_{1:t-1})\, p_\theta(s_t \mid s_{1:t-1}, e_{1:t-1})}{p_\theta(s_t \mid s_{1:t-1}, e_{1:t})} \\
&= \frac{p_\theta(e_t \mid s_{1:t}, e_{1:t-1})\, p_\theta(s_t \mid s_{1:t-1}, e_{1:t-1})}{\frac{p(s_{1:t}, e_{1:t})}{p(s_{1:t-1}, e_{1:t})}} \\
&= \frac{p_\theta(e_t \mid s_{1:t}, e_{1:t-1})\, p_\theta(s_t \mid s_{1:t-1}, e_{1:t-1})}{\frac{p(e_t \mid s_{1:t}, e_{1:t-1})\, p(s_t \mid s_{1:t-1}, e_{1:t-1})\, p(s_{1:t-1}, e_{1:t-1})}{p(e_t \mid s_{1:t-1}, e_{1:t-1})\, p(s_{1:t-1}, e_{1:t-1})}} \\
&= p(e_t \mid s_{1:t-1}, e_{1:t-1})
\end{aligned} \tag{35}$$

The perfect proposal distribution is closely related to the Auxiliary Particle Filter. Because one can see here that the weight does not depend on s_t, so one could reverse the sampling and weighting process (that would note make a difference here obviously, but the key here is that one could then also weight s_t given e_{t+1} before sampling s_t, a goal the APF wants to accomplish by targeting a slightly different distribution). Note that, in most cases a proposal of the form $q(s_t \mid s_{1:t-1}, e_{1:t})$ is unattainable and approximations have to be applied. We will not discuss proposal candidates further here, and refer to [Kantas et al., 2015, Doucet and Johansen, 2011] for further review.

A pseudo algorithm implementation for a standard PF can be found in algorithm 2, where the auxiliary variable a_t^i refers to the ancestor path of a particular particle s^i at time t. Hence a particle trajectory can be recursively defined as $s_{1:t}^i = (s_{1:t-1}^{a_t^i}, s_t^i)$. Resampling the whole particle trajectory is equivalent to sampling a new ancestor path. Note that it is usually much faster to sample ancestors one at a time and then recursively recover the resampled particle path than to resample the whole particle trajectory at each iteration. A variant of this algorithm is known as Conditional Particle Filter (CPF), where one particle path, $s'_{1:T}$, is chosen a priori as reference trajectory. This implementation tracks a slightly different target distribution, $\hat{p}(s_{1:T} \mid s'_{1:T}, e_{1:T})$. In this case, the last particle at each iteration, s_t^N, is determined by the reference trajectory. In the basic CPF case, the last particle at each iteration, s_t^N is deterministically set to s'_t, which is equivalent to $a^N = N$. [Lindsten et al., 2014, 2015] introduce the ancestor sampling step, where instead of fixing $a^N = N$, the ancestors are drawn with probability

$$\begin{aligned}
P(a_t^N = i) &= \alpha_{t-1}^i(s_{1:t-1}^i, e_{1:t-1})\, \frac{p(s_{1:t-1}^i, s'_{t:T} \mid e_{1:T})}{p(s_{1:t-1}^i \mid e_{1:t-1})} \\
&= \alpha_{t-1}^i(s_{1:t-1}^i, e_{1:t-1})\, \frac{p(s_{1:t-1}^i, s'_{t:T}, e_{1:t-1}, e_{t:T})}{p(s_{1:t-1}^i, e_{1:t-1})} \\
&= \alpha_{t-1}^i(s_{1:t-1}^i, e_{1:t-1})\, \frac{p(s'_{t:T}, e_{t:T} \mid s_{1:t-1}^i, e_{1:t-1}) p(s_{1:t-1}^i e_{1:t-1})}{p(s_{1:t-1}^i, e_{1:t-1})} \\
&= \alpha_{t-1}^i(s_{1:t-1}^i, e_{1:t-1})\, p(s'_{t:T}, e_{t:T} \mid s_{1:t-1}^i, e_{1:t-1})
\end{aligned} \tag{36}$$

Here, $\alpha_{t-1}^i(s_{1:t-1}^i, e_{1:t-1})$ can be interpreted as the prior probability of s_{t-1}^i, and $p(s'_{t:T}, e_{t:T} \mid s_{1:t-1}^i, e_{1:t-1})$ as the likelihood of observing s'_t given s_{t-1}^i. Note that, depending on the SSM, $p(s'_{t:T}, e_{t:T} \mid s_{1:t-1}^i, e_{1:t-1})$ will have a much reduced memory, both backward and forward in time. With this step, one mimics backward smoothing in a forward only manner, without the additional computational costs - which makes it possible to run this sampler with less number of particles N for the same variance of the likelihood estimate. A pseudo algorithm for this Conditional Particle Filter (CPF) with ancestor sampling can be found in algorithm 3.

Once a particle filter has been run, the particles can be reused to forecast both observed and latent data to first sample a new state $S_{T+1} \sim p_\theta(\cdot \mid s_{1:T}, e_{1:T})$ and then a new data point given this state $E_{T+1} \sim p_\theta(\cdot \mid s_{1:T+1}, e_{1:T})$. S_{T+1} can be sampled by forward propagating algorithm 2 or 3 from T to $T+1$, reusing particles from 1 to T. If θ is fixed, one can easily sample from $p_\theta(s_{T+1:T+i}, e_{T+1:T+i} \mid e_{1:T}, e_{1:T})$, for $i \geq 1$, by

- (1) running a particle filter that targets $p(s_{1:T} \mid e_{1:T}, \theta)$.
- (2) **for** $n \leftarrow T+1$ **to** $T+i$ **do**
  - (i) draw $S_n \sim p_\theta(\cdot \mid s_{1:n-1}, e_{1:n-1})$,
  - (ii) draw $E_n \sim p_\theta(\cdot \mid s_{1:n}, e_{1:n-1})$.

In (2), the Particle Filter can be updated online to propagate particles up to $T+i$, resulting in a very fast procedure to sample from predictive distributions. However, this mechanism will be noisy if the times series becomes too long, and model parameter may change the further the time series is increasing in time t.

3.1.1 Tuning a Particle Filter

In our setup, we use a standard bootstrap particle filter with transition distribution as in chapter 2. The particle resampling method is chosen to be systematic, and the resampling threshold for the ESS calculation set to 75%. The only free tuning parameter in this case is the number of particles N. The higher N, the lower the variance for the log likelihood estimate, but the higher the computational costs. Often, people set N equal to the number of data points received, but in practice significantly less particle may be used. As a sanity check for the HSMM, we compute a likelihood estimate for a range of parameter values for μ, σ, k, r, p of the HSMM described above for 1000 data points, and the log likelihood estimate and the corresponding analytical value can be seen in figure 5. As there is little difference in the variance between 50% and 200% particles in terms of total data points, we will use the former figure as N for our analysis. Note that this is different for each individual SSM, for instance we found that as little as 10% is sufficient for a standard HMM, but the additional latent variable in the HSMM causes a higher variance for the log likelihood estimate of the particle filter for the latter model. Another interesting observation of this illustration is that the variance of the likelihood estimate is not constant across the whole parameter range for some parameter, so a more optimized tuning technique to choose the number of particles might lead to better results for both the PMCMC and SMC2 algorithm.

3.2 Particle Markov Chain Monte Carlo

At the beginning of this chapter, joint estimation techniques for the latent trajectory $S_{1:T}$ and model parameter vector θ are discussed. The former case has already been covered, what about θ? The most common Bayesian inference technique is Markov Chain Monte Carlo. We assume basic familiarly with this concept, and refer to Craiu and Rosenthal [2014] for a more detailed review about standard MCMC techniques. A pseudo algorithm for a basic Metropolis step can be found in algorithm 4. The major difficulty in algorithm 4 is finding a good proposal distribution f, which often results in slow mixing. A MCMC kernel that automatically tunes its proposal distribution at the cost of additional tuning parameter is known as Hamiltonian Monte Carlo (HMC), see [Neal, 2012] for an introduction and [Betancourt, 2018] for a review on this topic. To draw from some posterior distribution of interest, an auxiliary momentum variable ρ is introduced to draw from the joint density $p(\rho, \theta)$. Usually, it is assumed ρ does not depend on θ, $p(\rho, \theta) = p(\theta)p(\rho)$. The so called Hamiltonian is defined as

$$\begin{aligned} H(\rho, \theta) &= -log\, p(\rho, \theta) \\ &= -logp(\rho \mid \theta) - log\, p(\theta) \\ &= T(\rho \mid \theta) + V(\theta), \end{aligned} \tag{37}$$

where $T(\rho \mid \theta)$ is called "kinetic energy", and $V(\theta)$ "potential energy". A common choice for the kinetic energy is the Gaussian distribution centered around 0, $p(\rho \mid \theta) = N(\rho \mid 0, M)$, where M is a covariance matrix commonly referred to as mass matrix. This defines the Euclidean kinetic energy $T(\rho \mid \theta) = \frac{1}{2}\rho^T M^{-1}\rho + log\, |M| + constant$, but other choices are possible. $V(\theta)$ will be the target distribution of interest, i.e., the log posterior distribution,

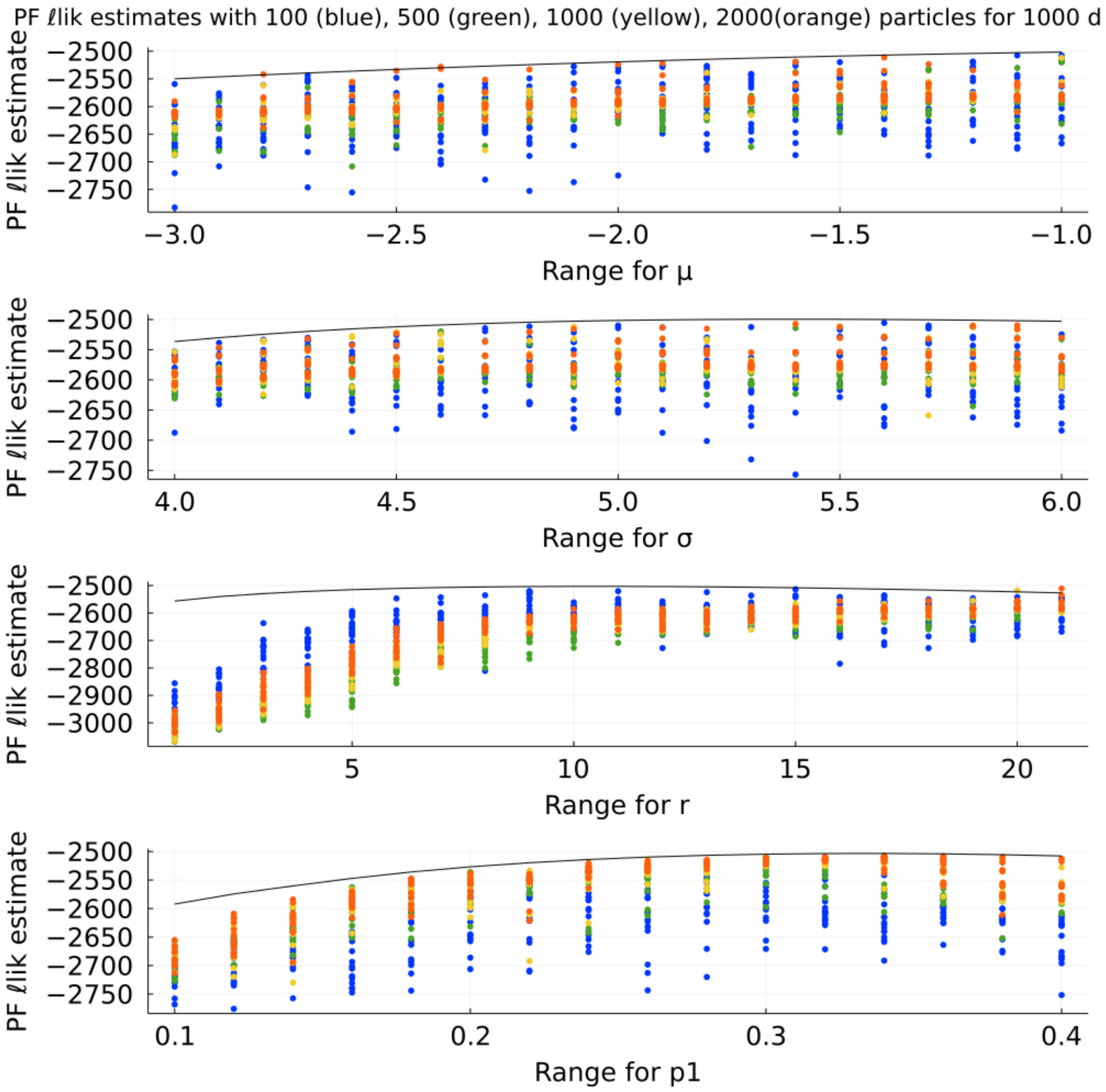

Figure 5: Pf likelihood estimate. This graph shows particle filter likelihood estimates for a range of different parameter values against the exact solution (black line). At each column, all parameter were kept constant except the labeled parameter at the x-axis. The different colors depict different amount of particles used for the computation: 100 (blue), 500 (green), 1000 (yellow), 2000 (orange) particles for sample data of size 1000.

$V(\theta) \propto p(\theta \mid e_{1:T})$. This joint system $\{\rho, \theta\}$ evolves via the Hamiltonian equations

$$\begin{aligned} \frac{d\theta}{dt} &= +\frac{\partial H}{\partial \rho} = +\frac{\partial T}{\partial \rho} \\ \frac{d\rho}{dt} &= -\frac{\partial H}{\partial \theta} = -\frac{\partial T}{\partial \theta} - \frac{\partial V}{\partial \theta}. \end{aligned} \tag{38}$$

To solve this two-state differential equations, one can use, e.g., a leapfrog integrator, which is described in more detail in pseudo algorithm 5. A drawback of this kernel are the additional tuning parameter M, ϵ and L. An automatic tuning version of this algorithm is known as No U-Turn Sampling (NUTS) and was proposed by Hoffman and Gelman [2014]. Notable improvements for this algorithm are suggested in Betancourt [2016]. It is out of the scope of this paper to explain all adjustments in detail, and we instead refer to said literature. Note that both the HMC and NUTS kernel require the target density to be fully differentiable with respect to the model parameter θ.

In the SSM case, targetting the marginal posterior distribution $p(\theta \mid e_{1:T}) \propto p_\theta(e_{1:T})\, p(\theta)$ is difficult or impossible via MCMC, as we have to integrate out latent variables in $p_\theta(e_{1:T}) = \int_{s_{1:T}} p_\theta(e_{1:T}, s_{1:T})$ in the proposal ratio. However, $p_\theta(y_{1:T}, s_{1:T})$ is usually computable pointwise, so we could instead try to target $p(s_{1:T}, \theta \mid y_{1:T})$. In this case, the acceptance ratio would look like this:

$$a((s^\star_{1:T}, \theta^\star), (s_{1:T}, \theta)) = \frac{p_{\theta^\star}(s^\star_{1:T}, e_{1:T})\, p(\theta^\star)\, f((s_{1:T}, \theta) \mid (s^\star_{1:T}, \theta^\star))}{p_\theta(s_{1:T}, e_{1:T})\, p(\theta)\, f((s^\star_{1:T}, \theta^\star) \mid (s_{1:T}, \theta))} \tag{39}$$

The proposal distribution can be stated as $f((s^\star_{1:T}, \theta^\star) \mid (s_{1:T}, \theta)) = f(\theta^\star \mid \theta)\, p_{\theta^\star}(s^\star_{1:T} \mid e_{1:T})$. Plugging this into the equation above yields

$$a((s^\star_{1:T}, \theta^\star), (s_{1:T}, \theta)) = \frac{p_{\theta^\star}(e_{1:T})\, p(\theta^\star)\, f(\theta \mid \theta^\star)}{p_\theta(e_{1:T})\, p(\theta)\, f(\theta^\star \mid \theta)}, \tag{40}$$

where we could marginalize out the latent trajectory. However, we came back to our initial problem: computing $p_\theta(e_{1:T})$ is still impossible, and sampling from $p_\theta(s_{1:T} \mid e_{1:T})$ is challenging. In the Particle Metropolis Hastings (PMH) case, formally introduced in [Andrieu et al., 2010] and shown in pseudo algorithm 6, a particle filter is used to obtain approximations for $p_\theta(e_{1:T})$ and $p_\theta(s_{1:T} \mid e_{1:T})$ as substitutes for the analytical solutions to target $p(s_{1:T}, \theta \mid e_{1:T})$ jointly. In this setting, [Andrieu and Roberts, 2009] have shown the puzzling result that one can do so and still target the exact posterior distribution of interest. A major difficulty for this method is finding a good MCMC kernel $K_{mcmc}(e_{1:T}, \theta)$, because the θ proposal will be accepted based on the particle filter likelihood estimate, so tuning might be very noisy. Moreover, more advanced gradient based MCMC sampler do not work in this case as we typically cannot evaluate $p_\theta(e_{1:T})$ pointwise. A common critique on PMH is thus that this algorithm is ill-suited for a higher dimensional model parameter vector θ. A variant that can mitigate this is known as Particle Gibbs with ancestors sampling (PGAS), see [Lindsten et al., 2014, 2015]. A pseudo algorithm is shown in algorithm 7. The CPF used in this setting is explained in more detail in algorithm 3. To account for sampling from an approximation via a particle filter and to preserve the invariance principle, a slightly adjusted $\hat{p}_{\theta^\star}(s^\star_{1:T} \mid s_{1:T}, e_{1:T})$ distribution is used to sample from the state trajectory. This method does not jointly estimate the state sequence and model parameter, but the the state sequence is fixed when the new model parameter $\theta^\star \sim p_\theta(\theta^\star \mid s_{1:T}, e_{1:T})$ are sampled. This step can be easily replaced with one of the MCMC kernels stated earlier as $p_{\theta^\star}(e_{1:T} \mid s_{1:T})$, which is easy to evaluate, replaces $p_{\theta^\star}(e_{1:T})$ in the acceptance ratio. Hence, more advanced MCMC kernels, such as the NUTS sampler, can be used to estimate model parameter θ in the particle Gibbs setting.

3.2.1 Tuning a PMCMC Kernel

Once a particle filter is designed, only a suitable MCMC kernel has to be chosen. As discussed in section 3, more advanced gradient based MCMC sampler do not work in the Particle Metropolis Hastings case, as we typically cannot evaluate $p_\theta(e_{1:T})$ pointwise. Thus, we will use Particle Gibbs with ancestors sampling and target $\theta^\star \sim p_\theta(\theta^\star \mid s_{1:T}, e_{1:T})$ in the MCMC step. As we can easily calculate gradients for $p_{\theta^\star}(e_{1:T} \mid s_{1:T})$ in this case, we choose the NUTS MCMC variant [Hoffman and Gelman, 2014, Betancourt, 2016] as MCMC kernel, as other kernel often take significantly more proposal steps to move towards the typical set. This is especially relevant for PMCMC, as we will run a particle filter after each MCMC proposal. For the NUTS implementation, to the best of our knowledge, we followed the papers mentioned above for hyperparameter tuning.

3.3 Sequential Monte Carlo Squared

In a times series setting, forecasting is of major relevance. The standard way for prediction in a Bayesian setting is simple: obtain the posterior predictive distribution by integrating out the model parameter θ and, in the SSM case, the state trajectories $S_{1:T}$:

$$\begin{aligned} p(e_{T+1} \mid e_{1:T}) &= \int p(e_{T+1}, s_{T+1}, s_{1:T}, \theta \mid e_{1:T})\, ds_{T+1}, s_{1:T}, \theta \\ &= \int p_\theta(e_{T+1} \mid s_{T+1}, s_{1:T}, e_{1:T})\, p_\theta(s_{T+1} \mid s_{1:T}, e_{1:T})\, p(s_{1:T}, \theta \mid e_{1:T})\, ds_{T+1}, s_{1:T}, \theta. \end{aligned} \tag{41}$$

Once a sample for $p(s_{1:T}, \theta \mid e_{1:T})$ is obtained, the predictive distributions for $S_{T+1} \mid s_{1:T}, e_{1:T}, \theta$ and $E_{T+1} \mid s_{T+1}, s_{1:T}, e_{1:T}, \theta$ are trivial to sample from. A major drawback of the Particle MCMC machinery is that, even though this algorithm primarily works for models suited to times series settings, it only works as batch estimation. Once additional data is observed, the algorithm needs to be run again to target $p(s_{1:T+1}, \theta \mid e_{1:T+1})$. A method that uses PMCMC in a sequential setting is known as Sequential Monte Carlo Squared, see Chopin et al. [2013]. In this case, n sequences of distributions $p(s^n_{0:t}, \theta^n \mid e_{1:t})$ for $t = 1, \dots, T$ are explored iteratively by using multiple Particle Filter and particle MCMC sampler. A pseudo algorithm can be seen in algorithm 8. In this case, N standard Particle Filter are run to obtain the incremental likelihoods $\hat{p}_{\theta^n}(e_{1:t} \mid e_{1:t-1}) = \hat{Z}_t = \frac{1}{N}\sum_{n=1}^{N} \alpha_k(s^n_{1:t}, e_{1:t})$ and state trajectories $s^n_{1:t} \sim \hat{p}_{\theta^n}(s^n_{1:t} \mid e_{1:t})$ at each iteration. If the estimates $\hat{p}_{\theta^n}(e_{1:t} \mid e_{1:t-1})$ are getting too noisy, the model parameter θ^n are jittered via PMCMC. Note that one may choose either Particle Gibbs or the Particle Metropolis Hastings variant for the PMCMC kernel in algorithm 8.

There are several advantages of this algorithm. First of all, all sequences can be targeted in parallel. Furthermore, most iterations can be performed online, as the particle filter can be propagated forward online if no resampling step has been performed at the previous iteration. Ideally, one jitters particles at the beginning, where computation is cheap in comparison to later stages, and updates only occasionally towards the end of the data series. Hence this algorithm, while covering a n sequences of distributions p($s^n_{1:t}, \theta^n \mid e_{1:t}$), may be faster to estimate than the standard PMCMC algorithm. Moreover, while iterating though all the distributions, posterior predictive distributions for S_{t+1} and E_{t+1} and an estimate for marginal likelihood $p(e_{1:t})$ for each $t = 1, \dots, T$ can be obtained as a by-product, which will be discussed more in detail in chapter 4.

3.3.1 Tuning SMC2

As described in pseudo-algorithm 8, SMC2 has a particle filter and a PGAS algorithm assigned for each θ particle. Unlike in the PF case, there is no time-variance tradeoff in the number of particles, as we can compute all relevant steps in parallel. Hence, in practice, the number of θ particles is limited by the computational resources, and there is no other trade-off, so we will assume that a user always uses the maximum number of θ particles that is achievable with their computational resources. Each PF and PGAS algorithm will be tuned as described above, and each corresponding initial model parameter is drawn from the prior. Another tuning parameter is the number of rejuvenation steps in case jittering has to be applied. As a guideline, we will continue jittering until the correlation of the old and new continuous model parameter is below 90%. The initial number of data points also has a significant role for the computational time. Few enough that initial computation is cheap and covers a broad range of values for each parameter, such that the algorithm does not depend on initial parameter, but enough data points that at least two states can be discovered for the input data. This depends on the underlying data and the duration parameter,and there is no general guideline we can provide. Once the initial data points have been determined, the PF and PGAS pairs for each θ particle will have a few test runs before the SMC2 propagation begins. Depending on the model, we recommend between 10 and 50 runs if a NUTS kernel is used.

3.4 Model Selection

Once parameter are estimated, how do we evaluate the performance of our model? A simple and reasonable solution in a time series setting is to assess the model performance based on its forecasting capabilities, which, in the Bayesian setting, is the posterior predictive distribution. So how do we compare probabilistic forecasts? The Continuous Ranked Probability Score (CRPS) [Matheson and Winkler, 1976, Jordan et al., 2019] is a non-parametric method to compare predictive distribution of models. For forecasts $X_i, i = 1, \dots, m$ and observation y, the CRPS can be calculated as

$$CRPS(\hat{F_m}, y) = \frac{1}{m}\sum_{i=1}^{m} |X_i - y| - \frac{1}{2m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} |X_i - X_j|. \tag{42}$$

Parameter	True value	Mean	MCSE	StdDev	Q2.5	Q25	Q50	Q75	Q97.5	ESS	$\hat{R}$
$\mu 1$	-2.0	-2.277	0.006	0.199	-2.673	-2.417	-2.278	-2.144	-1.883	1158.0	1.003
$\mu 2$	2.0	1.916	0.002	0.076	1.764	1.863	1.916	1.971	2.061	2425.0	1.001
$\sigma 1$	4.0	3.871	0.003	0.133	3.608	3.78	3.868	3.957	4.138	2347.0	1.001
$\sigma 2$	2.0	1.922	0.001	0.07	1.805	1.877	1.916	1.959	2.081	2229.0	1.002
r1	10.0	12.568	0.14	4.017	5.109	9.455	12.752	15.816	19.396	827.0	1.004
r2	15.0	28.923	0.656	16.325	9.23	17.419	24.633	35.314	73.467	619.0	1.002
p11	0.30	0.348	0.003	0.079	0.179	0.291	0.358	0.412	0.467	809.0	1.004
p12	0.30	0.394	0.005	0.121	0.183	0.305	0.384	0.473	0.65	722.0	1.002

Table 1: Posterior output statistics for 4 PMCMC chains on HSMM, 1000 iterations with 200 iterations burnin, resulting in 3600 total samples.

Parameter	True value	Mean	MCSE	StdDev	Q2.5	Q25	Q50	Q75	Q97.5
$\mu 1$	-2.0	-2.305	0.009	0.225	-2.715	-2.454	-2.31	-2.155	-1.822
$\mu 2$	2.0	1.932	0.003	0.083	1.764	1.878	1.93	1.986	2.102
$\sigma 1$	4.0	3.903	0.005	0.143	3.625	3.801	3.899	4.007	4.184
$\sigma 2$	2.0	1.948	0.003	0.08	1.828	1.897	1.939	1.987	2.142
r1	10.0	12.363	0.155	4.268	5.007	8.944	12.168	16.103	19.602
r2	15.0	28.781	0.612	17.266	7.962	16.125	24.094	37.453	76.215
p11	0.30	0.344	0.004	0.084	0.176	0.286	0.35	0.413	0.473
p12	0.30	0.385	0.005	0.132	0.162	0.278	0.374	0.486	0.654

Table 2: Posterior output statistics for 100 SCMC chains on HSMM at final iterations.

The lower the cumulative CRPS score over all forecasts, the better the model performs. This will be our guideline going forward. Alternatively, during each SMC2 step, one obtains an unbiased estimate for the marginal likelihood $p(e_{1:t}), t = t_0, ..., T$, as a byproduct in each iteration step. One can use these estimates to compare model validity for all models at not extra computation costs. We will show both criteria in section 5.

4 Simulation and Experimental Results

This section consists of simulation experiments conducted to study the performance of the PF, the PMCMC and the SMC2 algorithm. We first generate 1000 data points from a HSMM with Negative Binomial state duration distributions. The model dynamics and prior assignments can be found in appendix A, and a plot with sampled observed and latent data is shown in figure 6. The same figure depicts a PF estimate of the latent parameter - the advantages of explicitly modelling duration are visible on the third subplot, where durations in each state vary drastically.

4.1 Parameter and state estimation

If parameter θ are unknown, we can use both the PMCMC and SMC2 machinery. As the latter builds on the former, we first show estimation results for PMCMC. The traceplots of four chains for the continuous model parameter for PMCMC can be seen in figure 9, and for the latent state sequence in figure 10. After parameter are initialized from the prior distributions, they rapidly converge to the values used to generate the sample data. One might alternatively opt to use any form of parameter optimization to obtain an initial estimate, but given convergence occurs fast we did not proceed with this idea. Common MCMC output statistics are summarized in table 1.

SMC2 results can be seen on figure 11 for model parameter, figure 12 for latent variables, and figure 13 for predictions. We will use the predictions to compare different models in section 5. Just like in the PMCMC sampler, samples converges fast towards the typical set, and are discussed in more detail in the corresponding plots referenced above. Common MCMC output statistics on the final SMC2 can be seen on table 2, which shows similar diagnostics as the PMCMC output table.

4.2 Determining the number of states

Another interesting topic is choosing the number of states that fit the data. Often, it is a priori difficult to decide a number of states. There are several distinct methods to so, but for this paper we choose methods from the overfitting

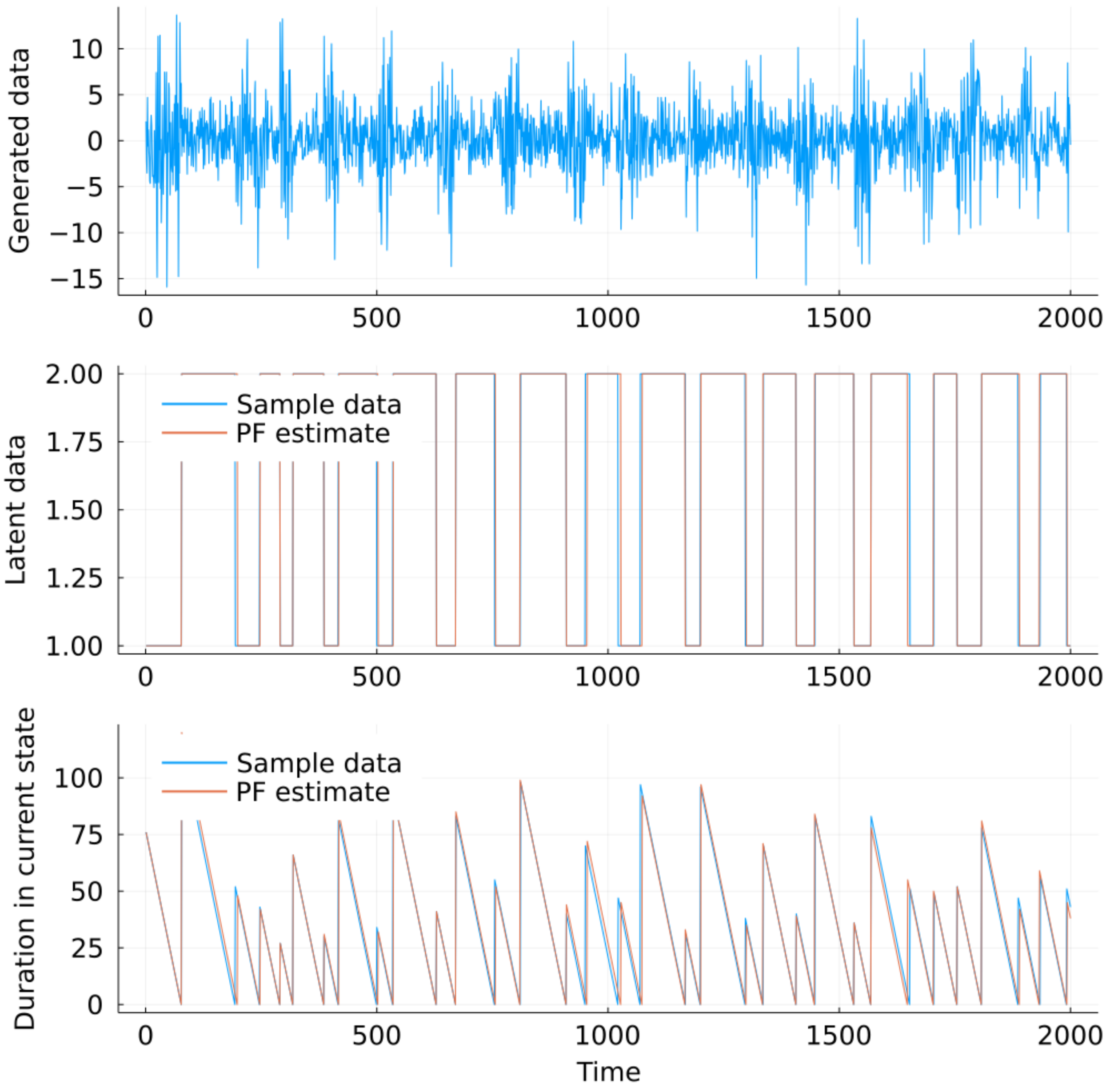

Figure 6: Generated observed and latent data for HSMM model. The upper graph shows generated sample data by the HSMM depicted in section 4. The middle plot shows the hidden state (blue) and a sample of filtered trajectory by a particle filter. The lower plot shows the remaining duration the current given state, and a sample of a filtered particles by a particle filter.

mixtures (see Rousseau and Mengersen [2011], Fruehwirth-Schnatter [2006], Fruehwirth-Schnatter et al. [2018]) literature. In particular, Rousseau and Mengersen [2011] show that the posterior distribution of a mixture model is much more stable by choosing prior weights of the mixture components that are concentrated on the boundary regions of the parameter space. We borrow this concept for SSMs and show that by assigning apropriate prior weights on the transition matrices of a HSMM, we can choose more states than necessarily describe the data and still have a interpretable structure returned, with superfluous latent states never visited. As an experiment, we fit a 2-, 3-, and 5-state HSMM to data that was generated by a 3-state HSMM. In section 3, we mentioned that the marginal likelihood is a byproduct of the inference procedure in SMC2. Hence, we we track the marginal likelihood at each iteration, and plotted the results in figure 7. One can see that the 3- and 5-state HSMMs marginal likelihood perfectly align to each other, hence we can conclude that it is possible to determing the number of states via SMC2, and even check for differences during the data propagation.

5 Applications

I need to add a bit more context here too - do egarch plots or just table? Should we add graphs for stylized financial facts? Maybe of a 2 state and a 5 state model with sampled data and its histogram? Switch hsmm graphs with hsmm-egarch plots.

Financial data is notoriously difficult to predict due to their sequential structure. Some popular empirical properties of it are known as stylized facts, which are illustrated in more detail in [Bulla and Bulla, 2006, McNeil et al., 2005]. Stylized facts differ based on the data's frequency, and in this section, we will largely focus on daily log-returns. How well can HSMMs model such properties? While financial returns are not independent, they do show little serial correlation. This can be readily modelled by most state space models. For example, a basic HMM fulfills this by having conditional independent, but unconditional, dependent observations on all past observations. Another property is that extreme returns tend to appear in clusters, which can be modelled by adding additional states into either tail. Moreover, volatility clusters and varies over time. This means that absolute or squared returns show profound serial correlation. A popular modelling approach to do this is by using stochastic volatility models. In this case, one models the volatility explicitly as auto-regressive process. This is similar to classical GARCH style models, in which case the volatility parameter is a function instead of a random variable. However, a major downside in this case is that one cannot model the state duration explicitly, which can be achieved by using HSMMs at the cost of modelling the volatility as a constant in each individual state. Hence, we combine both models, and model the volatility dynamics of a HSMM with Negative Binomial state duration distribution as an eGARCH model, which is fully defined in appendix A. Going forward, we will refer to this model as Hidden Semi-Markov Model with eGarch volatility dynamics (HSMM-eGARCH). Sample data generated by this model can be seen in figure 8.

In order to compare the validity of the HSMM-eGARCH, we will compare this model with different state duration distributions against other popular discrete state space models, such as the basic HSMM with various number of states and duration distributions, the basic HMM or a Markov Jump Model (MJM). All model dynamics and prior assignments can be seen in appendix A. A detailed methodology for model comparison can be found in section 3.4. For data collection, we first obtain the last 1000 daily end-of-day data points of the S&P500 Total Return Index as of October 1st 2021. The index data has been converted to returns, which can be seen in figure 14. We estimate model parameter for a HSMM with egarch volatility dynamics, HSMMs with different state duration distributions, a HMM and a MJM on this data via SMC2, and use the tuning mechanisms discussed in section 3. Detailed results for our proposed model can be seen on figure 15 for model parameter. The traceplot below shows all model parameter estimates at each point in time with the corresponding 95% credible interval. Interestingly, there seems to be a regime shift at the time COVID-19 first appears globally., which is captured very fast during the estimation process, which would be impossible for standard batch estimation. Figure 16 shows the filtered state trajectory of the latent variables at each time step. There is a clear distinction between a lower and higher volatility state, in which the latter seems to have a high autoregressive weight on the egarch dynamics parameter. A rescaled posterior mean of the latent variable at each time against the real data is shown in the bottom sub-plot, which shows the clear distinction between a volatile and a stable state in the times series. There is a bit more variation at the beginning, which makes sense because our continuous model parameter at the beginning of estimation is also more noisy. Figure 17 depicts model predictions against the realized data point at each time step. The cumulative CRPS over time used to evaluate model performances can be seen in figure 18, and a summary of all scores at the final time is shown in figure 3. The proposed HSMM-eGARCH obtains superior results consistently over time, both in terms of (predictive) CRPS score as well as in marginal likelihood.

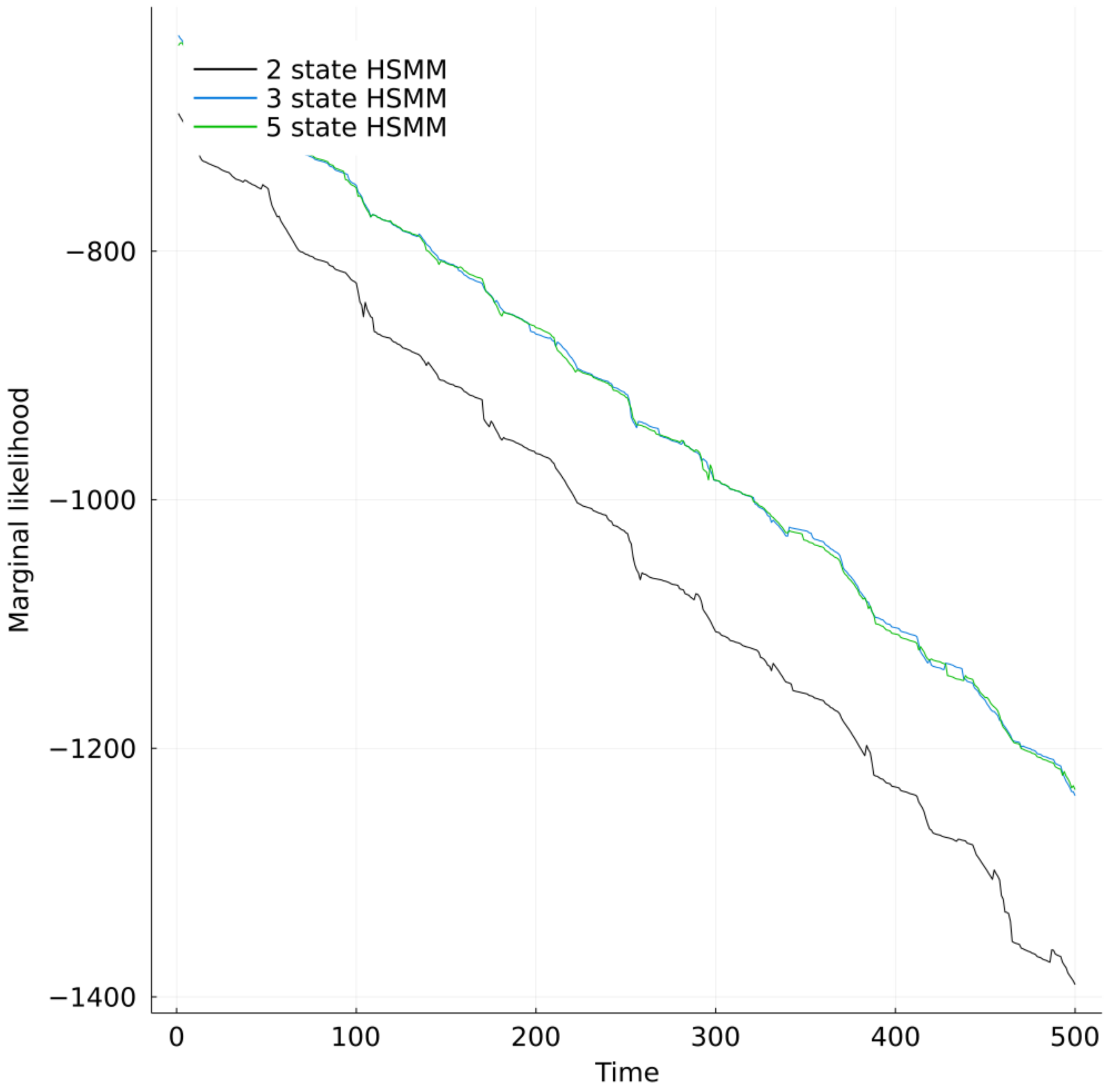

Figure 7: Marginal likelihood at each time step for a 2-, 3-, and 5-state HSMM. The underlying data was generated by a 3-state HSMM, and the 5-state HSMM returns the same marginal likelihood levels as the HSMM that generated the data, while the 2-state HSMM is not flexible enough to detect the data structure. Hence, we can ”overfit” the number of states and let the algorithm decide the number of states itself in case one has no prior knowledge about it.

Names	Cum. CRPS	Last Marginal Log Likelihood
HSMM - 2 State - NegBin Duration	485.782	-1486.71
HSMM - 3 State - NegBin Duration	480.818	-1427.44
HSMM - 2 State - Poisson Duration	493.804	-1776.75
HSMM - 3 State - Poisson Duration	515.374	-1421.9
HSMM-eGARCH - 2 State - NegBin Duration	473.218	-1416.54
HMM - 2 State	485.276	-1499.22
Markov Jump Model	502.117	-1597.96

Table 3: CRPS score and marginal log likelihood for several discrete state space models for data described in chapter 5 fitted via SMC2.

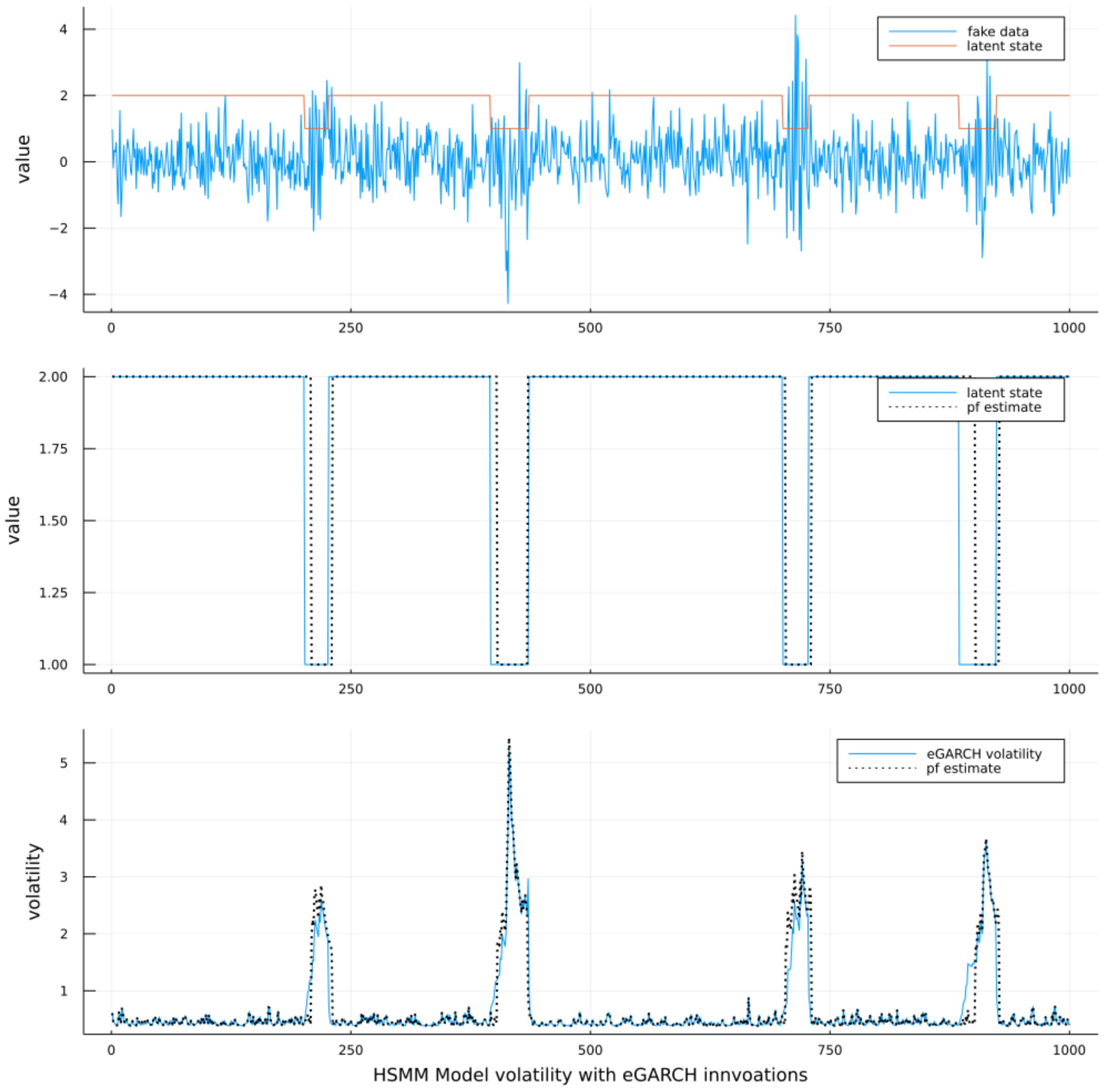

Figure 8: Observed and latent data generated by a HSMM with eGARCH volatility dynamics. The lowest plot shows the realized volatility at each time step. The dotted black lines show one sample obtained by a particle filter for known model parameter.

6 Conclusions

In this paper, we discuss sequential parameter estimation techniques for State Space Models, with a focus on Hidden Semi-Markov Models and propose various tuning mechanisms in chapter 4. While the additional duration variable in the HSMM typically makes parameter inference more challenging due to the increased computational costs of the likelihood function, it adds a lot more flexibility in modelling natural phenomena. Our proposed inference technique has the same computational costs for both the basic HMM and the HSMM and is particularly suitable for sequential data. We compare the predictive accuracy of various discrete SSMs on financial data, and conclude that the HSMM has a competitive predictive performance in addition to its attractive modelling assumptions.

Open research questions include finding the minimum number of initial data points for the initialization of the SMC2 algorithm. Few enough that initial computation is cheap and covers a broad range of values for each parameter, such that the algorithm does not depend on initial parameter, but enough data points that at least two states can be discovered for the input data. This will, of course, depend on the number of transitions to each state, which is unknown a priori. Another interesting observation we have is that the variance of the likelihood estimate for a particle filter is typically not constant across the whole parameter range for some parameter, so a more optimized tuning technique to choose the number of particles might lead to better results for both the PMCMC and SMC2 algorithm. Moreover, we did treat the tuning process for the MCMC sampler and the rest of the SMC2 algorithm independently, so there could be lots of improvement potential to combine all the different algorithm tuning processes in SMC2. Another potential improvement point is to come up with a better method to decide the number of jittering steps in the rejuventation process in SMC2. While the correlation between old and new parameter is a good indicator, we find that this sometimes results in way too much jittering steps for some models.

Acronyms

APF Auxiliary Particle Filter. 11, 21

BMI Basic Marginal Likelihood Identity. 9, 21

BPF Bootstrap Particle Filter. 11, 21

CPF Conditional Particle Filter. 11, 12, 14, 21

CRPS Continuous Ranked Probability Score. 15, 16, 18, 21

EDHMM Explicit-duration Hidden Markov Model. 6, 21

eGARCH Exponential Generalized Autoregressive Conditional Heteroskedasticity. 2, 18, 20, 21, 35–37

GARCH Generalized Autoregressive Conditional Heteroskedasticity. 21

HMC Hamiltonian Monte Carlo. 12, 14, 21

HMM Hidden Markov Model. 1–3, 5, 6, 8, 12, 18, 21

HSMM Hidden Semi-Markov Model. 1, 2, 6–8, 12, 16–18, 20, 21, 29–33, 35–37

HSMM-eGARCH Hidden Semi-Markov Model with eGarch volatility dynamics. 18, 21

IS Importance Sampling. 9, 21

MCMC Markov Chain Monte Carlo. 2, 3, 8, 12, 14, 15, 21

MJM Markov Jump Model. 18, 21

NUTS No U-Turn Sampling. 14, 15, 21

PF Particle Filter. 2, 9–12, 15, 16, 21

PGAS Particle Gibbs with ancestors sampling. 14, 15, 21

PMCMC Particle MCMC. 2, 9, 12, 15, 16, 21, 29, 30

PMH Particle Metropolis Hastings. 14, 21

SIR Sequential Importance Resampling. 10, 21
SIS Sequential Importance Sampling. 9, 10, 21
SMC Sequential Monte Carlo. 2, 21, 31–33, 35–37
SMC2 Sequential Monte Carlo Squared. 2, 12, 15, 16, 18, 19, 21
SSM State Space Model. 1–3, 9–12, 14, 15, 18, 21
SVM Stochastic Volatility Model. 21

A Model distributions and prior assumptions

- **HSMM with eGARCH volatility dynamics and Negative Binomial duration**,
  Model parameter: $\theta = \{\mu, r, p, k, \omega, \alpha, \beta, \delta, \nu\}$.
  - Model dynamics:
    - Data $E_t \sim N(\mu_{s_t}, h_t^2)$,
    - Latent state $S_t \sim \begin{cases} \delta(s_t, s_{t-1}) & d_{t-1} > 0 \\ Categorical(k_{s_{t-1}}) & d_{t-1} = 0 \end{cases}$
    - Latent duration $D_t \sim \begin{cases} \delta(d_t, d_{t-1} - 1) & d_{t-1} > 0 \\ NegativeBinomial(r_{s_t}, p_{s_t}) & d_{t-1} = 0 \end{cases}$,
    - $log(h_t^2) = \omega_{s_t} + \sum_k \alpha_{k,s_t} \times log(h_{t-k}^2) + \sum_j [\beta_{j,s_t} \times (|z_{t-j,s_{t-j}}| - E[|z_{t-j}| \mid s_{t-j}]) + \delta_{j,s_t} \times z_{t-j,s_{t-j}}]$.
    - $z_{t,s_t} = \frac{\epsilon_t}{h_t} = \frac{e_t - \mu_{s_t}}{h_t}$.
    - $E[|z_{t-j}| \mid s_{t-j}] = \frac{\Gamma(\frac{2}{\nu_{s_{t-j}}})}{[\Gamma(\frac{1}{\nu_{s_{t-j}}}) \times \Gamma(\frac{3}{\nu_{s_{t-j}}})]^{\frac{1}{2}}}$, where Γ is the Gamma function.
  - Model parameter priors:
    - $\mu \sim Normal(\mu = 0, \sigma = 10^5)$
    - $r \sim Normal_{(0,50]}(\mu = 20, \sigma = 10^5)$
    - $p \sim Beta(\alpha = 1, \beta = 2)$
    - $k \sim Dirichlet(\alpha_1 = \alpha_2 = ... = \alpha_k = k$, where $k =$ number of latent states.
    - $\omega \sim Normal_{(-5,5)}(\mu = 0, \sigma = 10^5)$
    - $\alpha \sim Normal_{(-1,1)}(\mu = 0, \sigma = 10^5)$
    - $\beta \sim Normal_{(-1,1)}(\mu = 0, \sigma = 10^5)$
    - $\delta \sim Normal_{(-1,1)}(\mu = 0, \sigma = 10^5)$
    - $\nu \sim Normal_{(0.1,10)}(\mu = 2, \sigma = 10^5)$
- **HSMM with Negative Binomial duration**,
  Model parameter: $\theta = \{\mu, \sigma, r, p, k\}$:
  - Model dynamics:
    - Data $E_t \sim N(\mu_{s_t}, \sigma_{s_t}^2)$,
    - Latent state $S_t \sim \begin{cases} \delta(s_t, s_{t-1}) & d_{t-1} > 0 \\ Categorical(k_{s_{t-1}}) & d_{t-1} = 0 \end{cases}$
    - Latent duration $D_t \sim \begin{cases} \delta(d_t, d_{t-1} - 1) & d_{t-1} > 0 \\ NegativeBinomial(r_{s_t}, p_{s_t}) & d_{t-1} = 0 \end{cases}$,
  - Model parameter priors:
    - $\mu \sim Normal(\mu = 0, \sigma = 10^5)$
    - $\sigma \sim Normal_{(0,10]}(\mu = 2, \sigma = 10^5)$
    - $r \sim Normal_{(0,50]}(\mu = 20, \sigma = 10^5)$
    - $p \sim Beta(\alpha = 1, \beta = 2)$
    - $k \sim Dirichlet(\alpha_1 = \alpha_2 = ... = \alpha_k = k$, where $k =$ number of latent states.
- **HSMM with Poisson duration**,
  Model parameter: $\theta = \{\mu, \sigma, \lambda, k\}$:
  - Model dynamics:
    - Data $E_t \sim N(\mu_{s_t}, \sigma_{s_t}^2)$,
    - Latent state $S_t \sim \begin{cases} \delta(s_t, s_{t-1}) & d_{t-1} > 0 \\ Categorical(k_{s_{t-1}}) & d_{t-1} = 0 \end{cases}$
    - Latent duration $D_t \sim \begin{cases} \delta(d_t, d_{t-1} - 1) & d_{t-1} > 0 \\ Poisson(\lambda_{s_t}) & d_{t-1} = 0 \end{cases}$,
  - Model parameter priors:
    - $\mu \sim Normal(\mu = 0, \sigma = 10^5)$

        * $\sigma \sim Normal_{(0,10]}(\mu = 2, \sigma = 10^5)$
        * $\lambda \sim Normal_{(0,50]}(\mu = 20, \sigma = 10^5)$
        * $k \sim Dirichlet(\alpha_1 = \alpha_2 = ... = \alpha_k = k$, where $k =$ number of latent states.

- **HMM**,
  Model parameter: $\theta = \{\mu, \sigma, k\}$:
    - Model dynamics:
        * Data $E_t \sim Normal(\mu_{s_t}, \sigma_{s_t})$,
        * Latent state $S_t \sim Categorical(k_{s_{t-1}})$,
    - Model parameter priors:
        * $\mu \sim Normal(\mu = 0, \sigma = 10^5)$
        * $\sigma \sim Normal_{(0,10]}(\mu = 2, \sigma = 10^5)$
        * $k \sim Dirichlet(\alpha_1 = \alpha_2 = ... = \alpha_k = k$, where $k =$ number of latent states.
- **MJM**,
  Model parameter: $\theta = \{\mu, \sigma, \mu_j, \sigma_j, \lambda\}$:
    - Model dynamics:
        * Data $E_t \sim Normal(\mu + N_t\mu_j, \sigma^2 + N_t\sigma_j^2)$,
        * Latent jump state $N_t \sim Bernoulli(\lambda)$,
        * $\lambda \sim U(0, 1)$.
    - Model parameter priors:
        * $\mu \sim Normal(\mu = 0, \sigma = 10^5)$
        * $\sigma \sim Normal_{(0,10]}(\mu = 2, \sigma = 10^5)$
        * $\mu_j \sim Normal(\mu = 0, \sigma = 10^5)$
        * $\sigma_j \sim Normal_{(0,10]}(\mu = 2, \sigma = 10^5)$
        * $\lambda \sim Beta(\alpha = 1, \beta = 4)$

B Pseudo Algorithms

Algorithm 1: Sampling an EDHMM

```
input  : Parameter $\theta$, number of iterations $T$
output: Samples for latent states $\{s_{1:T}, d_{1:T}\}$ and observations $e_{1:T}$
// Initialize:
1  $S_1 \sim \pi_\theta(s_1)$, $D_1 \sim \pi_\theta(d_1)$, $E_1 \mid s_1 \sim g_\theta(e_1 \mid s_1)$. ;
// Forward sample up to T
2  for $t \leftarrow 2$ to $T$ do
3      $$S_t \mid s_{t-1}, d_{t-1} \sim \begin{cases} \delta(s_t, s_{t-1}) & d_{t-1} > 0 \\ f_\theta(s_t \mid s_{t-1}, d_{t-1}) & d_{t-1} = 0 \end{cases}$$
4      $$D_t \mid s_t, d_{t-1} \sim \begin{cases} \delta(d_t, d_{t-1} - 1) & d_{t-1} > 0 \\ h_\theta(s_t \mid s_t, d_{t-1}) & d_{t-1} = 0 \end{cases}$$
5      $$E_t \mid s_t \sim g_\theta(e_t \mid s_t)$$
```

Algorithm 2: Standard particle filter

```
input              : data $e_{1:T}$, model parameter $\theta$
output             : log-likelihood estimate $\hat{\ell}(\theta) = log\ \hat{p}_\theta(e_{1:T})$ and sample $s_{1:T} \sim \hat{p}(s_{1:T} \mid e_{1:T})$
tuning parameter: proposal distribution $q$, number of particles $N$
function           : particle filter $pf(e_{1:T}, \theta)$
// Initialization:
1  for $n \leftarrow 1$ to $N$ do
2      Initiate particle $S_1^n \sim \pi_\theta(s_1)$.
3      Compute $\alpha_1^n(s_1^n, e_1) = \frac{p_\theta(e_1|s_1^n)\ p_\theta(s_1^n)}{q(s_1^n|e_1)}$.
4  Normalize weights $\tilde{\alpha}_1^i \propto \alpha_1^i(s_1^n, e_1)$ for $i = 1 : N$, s.t. $\sum_{i=1}^N \tilde{\alpha}_1^i = 1$.
5  Compute log likelihood increment $\hat{\ell}(\theta) = log\ \frac{1}{N} \sum_{i=1}^N \alpha_1^i(s_1^i, e_1)$
// Forward propagation:
6  for $t \leftarrow 2$ to $T$ do
7      if Resampling required then
8          Sample ancestor $a_t^n$ for particle trajectory $s_{1:t-1}^n$ for $n = 1$ to $N$ according to normalized weights $\tilde{\alpha}_{t-1}$.
9      else
10         Set $a_t^n = n$ for $n = 1$ to $N$.
11     for $n \leftarrow 1$ to $N$ do
12         Sample $s_t^n \sim q(s_t^n \mid s_{1:t-1}^{a_t^n}, e_{1:t})$.
13         Set $s_{1:t}^n := (s_{1:t-1}^{a_t^n}, s_t^n)$.
14         Calculate incremental weight:
               $$\alpha_t(s_{1:t}^n, e_{1:t}) = \frac{p_\theta(e_t \mid s_{1:t}^n, e_{1:t-1})\ p_\theta(s_t^n \mid s_{1:t-1}^n, e_{1:t-1})}{q(s_t^n \mid s_{1:t-1}^n, e_{1:t})}$$
15     Normalize weights $\tilde{\alpha}_t^i \propto \alpha_t^i(s_{1:t}^i, e_{1:t})$ for $i = 1$ to $N$, s.t. $\sum_{i=1}^N \tilde{\alpha}_t^i = 1$.
16     Add incremental weights to log likelihood: $\hat{\ell}(\theta) = \hat{\ell}(\theta) + log\ \frac{1}{N} \sum_{i=1}^N \alpha_t^i(s_{1:t}^i, e_{1:t})$.
// Return log likelihood estimate and particle trajectories:
17 Draw k with $P(k = i) \propto \tilde{\alpha}_T^i$.
18 return $\hat{\ell}(\theta)$ and $s_{1:T}^k$.
```

Algorithm 3: Conditional particle filter with ancestor sampling

```
input             : Reference $s^{'}_{1:T}$, data $e_{1:T}$, model parameter $\theta$
output            : Log-likelihood estimate $\hat{\ell}(\theta) = log\ \hat{p}_\theta(e_{1:T})$ and sample $s_{1:T} \sim \hat{p}(s_{1:T} \mid s^{'}_{t:T}, e_{1:T})$
tuning parameter: proposal distribution $q$, number of particles $N$
function          : particle filter $cpf(s^{'}_{1:T}, e_{1:T}, \theta)$
   // Initialization:
1  for $n \leftarrow 1$ to $N-1$ do
2      Initiate particle $S^n_1 \sim \pi_\theta(s_1)$.
3      Compute $\alpha^n_1(s^n_1, e_1) = \frac{p_\theta(e_1|s^n_1)\ p_\theta(s^n_1)}{q(s^n_1|e_1)}$.
4  Set $s^N_1 := s^{'}_1$ and compute $\alpha^N_1(s^N_1, e_1) \propto \frac{p_\theta(e_1|s^N_1)\ p_\theta(s^N_1)}{q(s^N_1|e_1)}$.
5  Normalize weights $\tilde{\alpha}^i_1 \propto \alpha^i_1(s^n_1, e_1)$ *for* $i = 1 : N$, *s.t.* $\sum_{i=1}^N \tilde{\alpha}^i_1 = 1$.
6  Compute log likelihood increment $\hat{\ell}(\theta) = log\ \frac{1}{N}\sum_{i=1}^N \alpha^i_1(s^i_1, e_1)$
   // Forward propagation:
7  for $t \leftarrow 2$ to $T$ do
8      if *Resampling required* then
9          Sample ancestor $a^n_t$ for particle trajectory $s^n_{1:t-1}$ for $n = 1$ to $N-1$ according to normalized weights
           $\tilde{\alpha}_{t-1}$.
10         Set $a^N_t = k$, with $P(k = i) \sim \alpha^i_{t-1}(s^i_{1:t-1}, e_{1:t-1})\ p_\theta(s^{'}_{t:T}, e_{t:T} \mid s^i_{1:t-1}, e_{1:t-1})$.
11     else
12         Set $a^n_t = n$ for $n = 1$ to $N$.
13     for $n \leftarrow 1$ to $N$ do
14         Sample $s^n_t \sim q(s^n_t \mid s^{a^n_t}_{1:t-1}, e_{1:t})$.
15         Set $s^n_{1:t} := (s^{a^n_t}_{1:t-1}, s^n_t)$.
16         Calculate incremental weight:
               $$\alpha_t(s^n_{1:t}, e_{1:t}) = \frac{p_\theta(e_t \mid s^n_{1:t}, e_{1:t-1})\ p_\theta(s^n_t \mid s^n_{1:t-1}, e_{1:t-1})}{q(s^n_t \mid s^n_{1:t-1}, e_{1:t})}$$
17     Normalize weights $\tilde{\alpha}^i_t \propto \alpha^i_t(s^i_{1:t}, e_{1:t})$ for $i = 1$ to $N$, s.t. $\sum_{i=1}^N \tilde{\alpha}^i_t = 1$.
18     Add incremental weights to log likelihood: $\hat{\ell}(\theta) = \hat{\ell}(\theta) + log\ \frac{1}{N}\sum_{i=1}^N \alpha^i_t(s^i_{1:t}, e_{1:t})$.
   // Return log likelihood estimate and particle trajectories:
19 Draw k with $P(k = i) \propto \tilde{\alpha}^i_T$.
20 return $\hat{\ell}(\theta)$ and $s^k_{1:T}$.
```

Algorithm 4: Metropolis Hastings (MH) Kernel

```
input             : data $e_{1:T}$, current model parameter $\theta$
output            : model parameter $\theta \sim p(\theta \mid e_{1:T})$
tuning parameter: proposal distribution $f$
function          : MCMC Kernel $K_{mh}(e_{1:T}, \theta)$
1  Propose $\theta^\star \sim f(\theta^\star \mid \theta)$.
2  Set $\theta := \theta^\star$ with acceptance probability $min(1, a(\theta^\star, \theta))$, where
       $$a(\theta^\star, \theta) = \frac{p(\theta \mid e_{1:T})\ f(\theta \mid \theta^\star)}{p(\theta \mid e_{1:T})\ f(\theta^\star \mid \theta)} = \frac{p_{\theta^\star}(e_{1:T})\ p(\theta^\star)\ f(\theta \mid \theta^\star)}{p_\theta(e_{1:T})\ p(\theta)\ f(\theta^\star \mid \theta)}.$$
3  return $\theta$.
```

```
Algorithm 5: Hamiltonian Monte Carlo (HMC) Kernel
input           : data $e_{1:T}$, current model parameter $\theta$
output          : model parameter $\theta \sim p(\theta \mid e_{1:T})$
tuning parameter: Mass matrix $M$, stepsize $\epsilon$, number of leapfrog steps $L$.
function        : MCMC Kernel $K_{HMC}(e_{1:T}, \theta)$
1  Sample $\rho \sim MvNormal(0, M)$ and set $(\theta^\star, \rho^\star) := (\theta, \rho)$.
2  for $i \leftarrow 1$ to $L$ do
3      $\theta^\star, \rho^\star = Leapfrog(\theta^\star, \rho^\star, M, \epsilon)$
4  Set $\theta := \theta^\star$ with acceptance probability $min(1, a(\theta^\star, \theta))$, where
       $$a(\theta^\star, \theta) = exp(H(\rho, \theta) - H(\rho^\star, \theta^\star))$$
5  return $\theta$.
6  Function Leapfrog($\theta_t, \rho_t, M, \epsilon$):
7      $\rho_{t+\frac{\epsilon}{2}} \leftarrow \rho_t + \frac{\epsilon}{2} \frac{\partial log\ p(\theta|e_{1:T})}{\partial \theta}(\theta_t)$
8      $\theta_{t+\epsilon} \leftarrow \theta_t + \epsilon M^{-1} \rho_{t+\frac{\epsilon}{2}}$
9      $\rho_{t+\epsilon} \leftarrow \rho_{t+\frac{\epsilon}{2}} + \frac{\epsilon}{2} \frac{\partial log\ p(\theta|e_{1:T})}{\partial \theta}(\theta_{t+\epsilon})$
10     return $\theta_{t+\epsilon}, \rho_{t+\epsilon}$.
```

```
Algorithm 6: Particle Metropolis Hastings Kernel
input           : current state trajectory $s_{1:T}$, data $e_{1:T}$, current model parameter $\theta$
output          : model parameter $\theta$ and state trajectory $s_{1:T}$, $(\theta, s_{1:T}) \sim p(\theta, s_{1:T} \mid e_{1:T})$
tuning parameter: MCMC kernel $K_{mcmc}$, particle filter $pf$, number of iterations $N$
function        : PMCMC kernel $K_{pmh}(e_{1:T}, s_{1:T}, \theta)$
1  Propose $\theta^\star \sim K_{mcmc}(e_{1:T}, \theta)$
2  Run particle filter $pf$ to obtain $\hat{p}_{\theta^\star}(e_{1:T})$ and $s^\star_{1:T} \sim \hat{p}_{\theta^\star}(s^\star_{1:T} \mid e_{1:T})$.
3  Set $(\theta, s_{1:T}) := (\theta^\star, s^\star_{1:T})$ with acceptance probability $min(1, a(\theta^\star, \theta))$, where
       $$a(\theta^\star, \theta) = \frac{\hat{p}_{\theta^\star}(e_{1:T})\, p(\theta^\star)\, f_{mcmc}(\theta \mid \theta^\star)}{\hat{p}_\theta(e_{1:T})\, p(\theta)\, f_{mcmc}(\theta^\star \mid \theta)} \qquad (43)$$
4  return $(\theta, s_{1:T})$
```

```
Algorithm 7: Particle Gibbs Kernel
input           : reference trajectory $s_{1:T}$, data $e_{1:T}$, current model parameter $\theta$
output          : model parameter $\theta$ and state trajectory $s_{1:T}$, $(\theta, s_{1:T}) \sim p(\theta, s_{1:T} \mid e_{1:T})$
tuning parameter: conditional particle filter $cpf$
function        : PMCMC kernel $K_{pgibbs}(e_{1:T}, s_{1:T}, \theta)$
1  Propose $\theta^\star \sim p_\theta(\theta^\star \mid s_{1:T}, e_{1:T})$
2  Run a conditional particle filter $cpf$ to obtain $s^\star_{1:T} \sim \hat{p}_{\theta^\star}(s^\star_{1:T} \mid s_{1:T}, e_{1:T})$.
3  Set $(\theta, s_{1:T}) := (\theta^\star, s^\star_{1:T})$
4  return $(\theta, s_{1:T})$
```

Algorithm 8: Sequential Monte Carlo Squared algorithm

```
input              : Data $e_{1:T}$
output             : model parameter $\theta$ and state trajectory $s_{1:t}$, $(\theta^i, s^i_{1:t})_{i=1:N} \sim p(\theta^i, s^i_{1:t} \mid e_{1:t})$ for $t = 1$ to $T$
tuning parameter: number of particles $N$, N particle filter $(pf_i)_{i=1:N}$, N PMCMC kernel $(K_{pmcmc,i})_{i=1:N}$
function           : SMC2 sampler $smc^2(e_{1:T})$

   // Initialization:
1  for $n \leftarrow 1$ to $N$ do
2      Initiate parameter vector $\theta_n \sim p(\theta_n)$.
3      Run particle filter $pf_n$ to obtain $s^n_{1:t_0} \sim \hat{p}_{\theta_n}(s^n_{1:t_0} \mid e_{1:t_0})$.
   // Transition:
4  for $t \leftarrow t_0 + 1$ to $T$ do
5      if Resampled at t-1 then
6          for $n \leftarrow 1$ to $N$ do
7              Run particle filter $pf_n$ to obtain $\hat{p}_{\theta^n}(e_{1:t} \mid e_{1:t-1})$ and $s^n_{1:t} \sim \hat{p}_{\theta^n}(s^n_{1:t} \mid e_{1:t})$.
8      else
9          for $n \leftarrow 1$ to $N$ do
10             Propagate particle filter $pf_n$ forward to obtain $\hat{p}_{\theta^n}(e_{1:t} \mid e_{1:t-1})$ and $s^n_t \sim \hat{p}_{\theta^n}(s^n_t \mid s^n_{1:t-1}, e_{1:t})$.
11             Set $s^n_{1:t} := (s^n_{1:t-1}, s^n_t)$.
12     Normalize incremental weights $\tilde{\alpha}^i_t \propto \hat{p}_{\theta^n}(e_{1:t} \mid e_{1:t-1})$ for $i = 1$ to $N$, s.t. $\sum_{i=1}^{N} \tilde{\alpha}^i_t = 1$.
13     if Resampling required then
14         for $n \leftarrow 1$ to $N$ do
15             Draw k with $P(k = i) \propto \tilde{\alpha}^i_t$.
16             Propose $(\theta^\star, s^\star_{1:t}) \sim K_{pmcmc,k}(e_{1:t}, s^k_{1:t}, \theta^k)$.
17             Set $(\theta^n, s^n_{1:t}) := (\theta^\star, s^\star_{1:t})$
18 return $(\theta^i, s^i_{1:t})_{i=1:N}$ for $t = 1$ to $T$.
```

C Plots

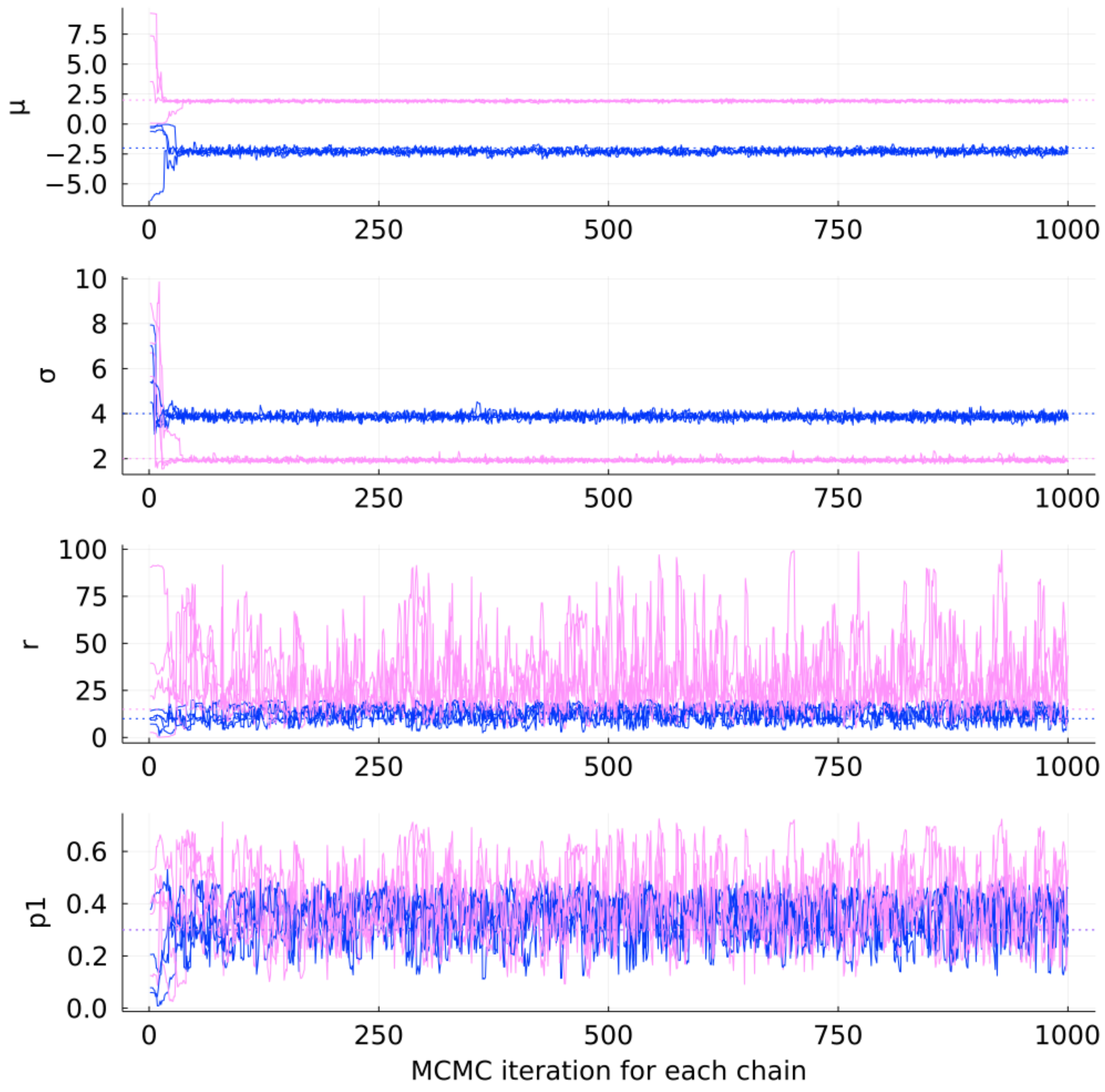

Figure 9: PMCMC chain. Particle MCMC traceplots of four chains for the HSMM in section 4. Parameter used to generate sample data are shown as dashed horizontal lines.

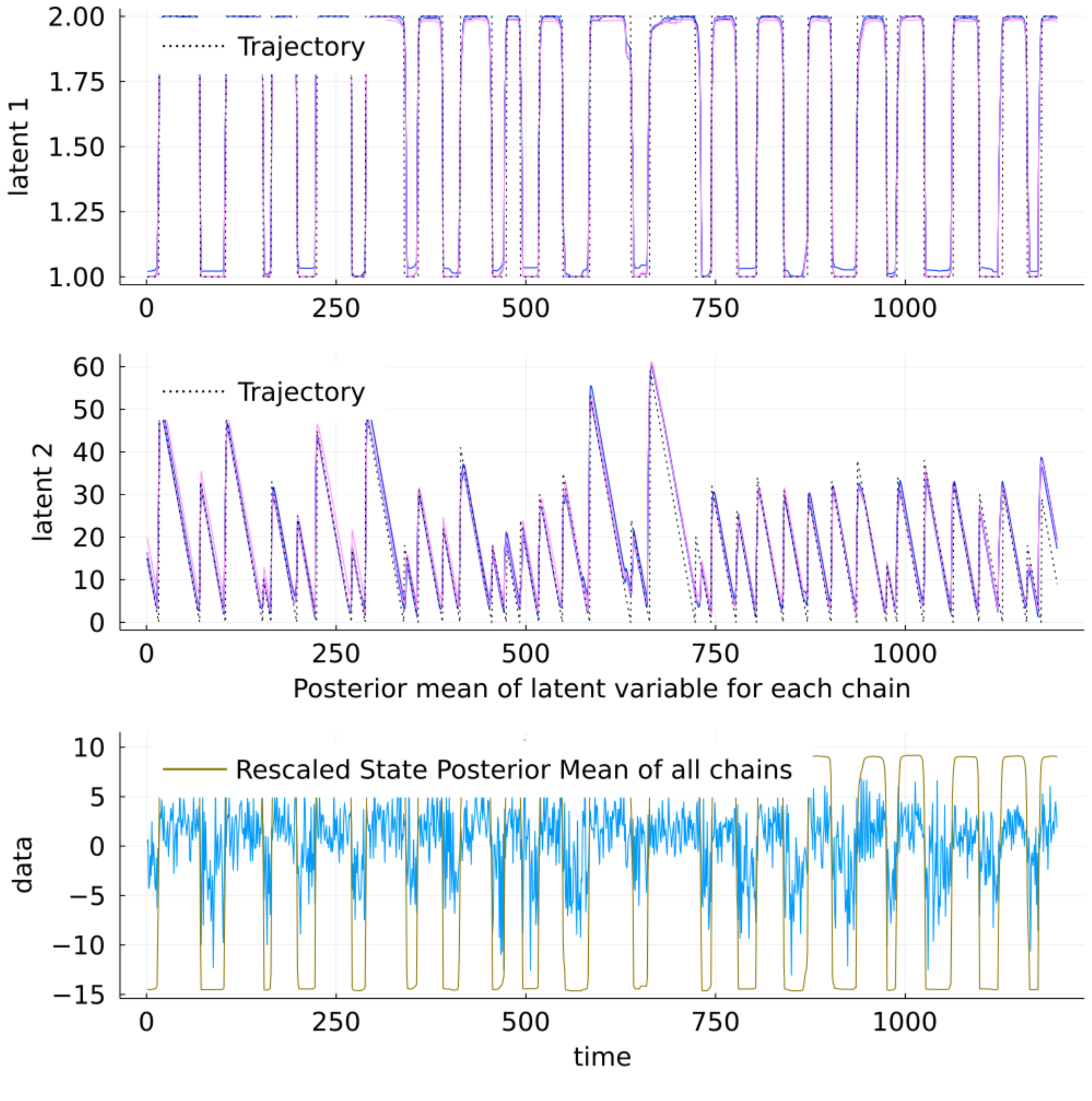

Figure 10: PMCMC latent. Particle MCMC posterior estimates of the filtered latent state trajectory of four chains for the HSMM in section 4. Parameter used to generate sample data are shown as dashed horizontal lines. The bottom plot shows rescaled posterior means

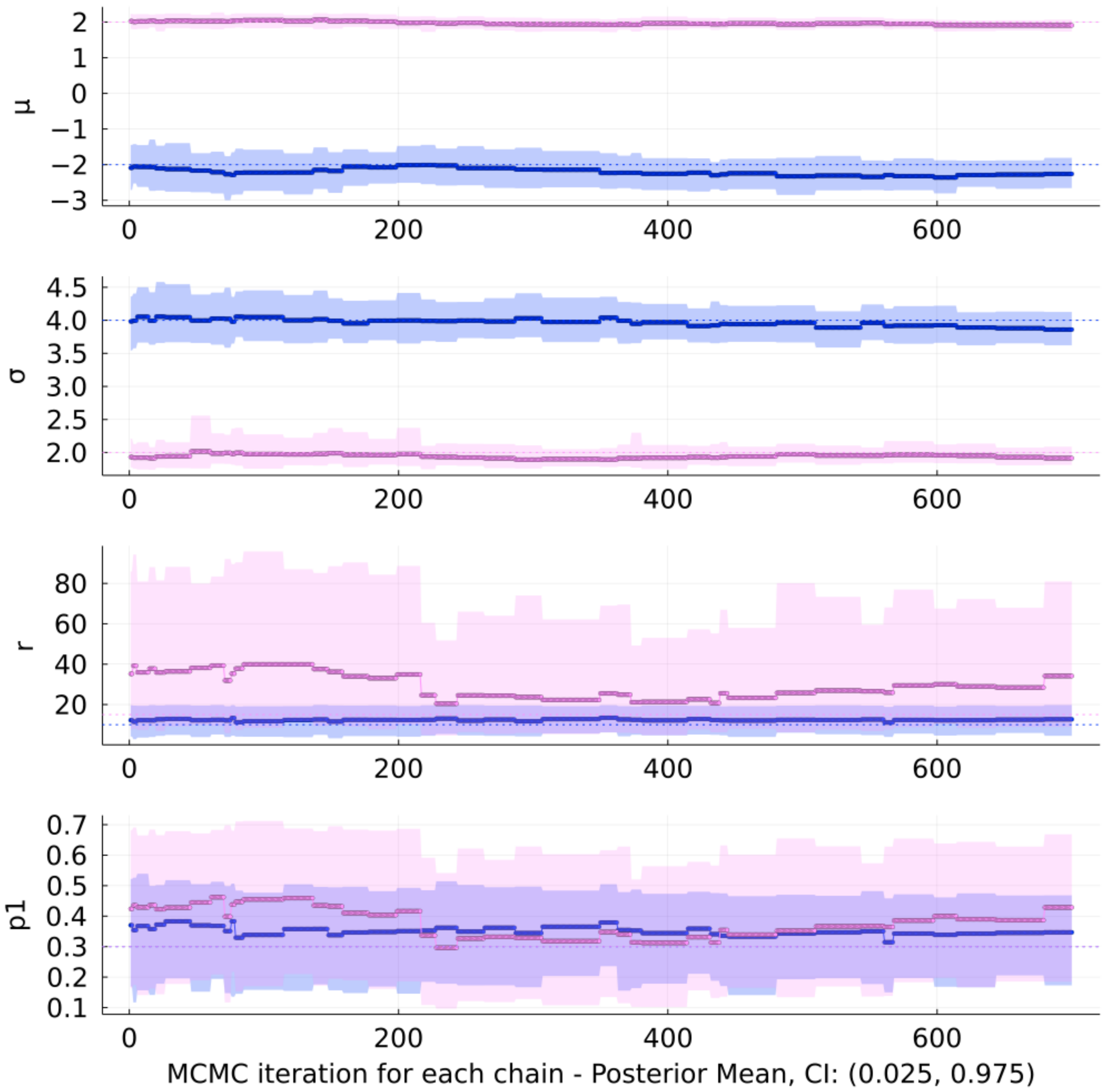

Figure 11: SMC chain. Sequential Monte Carlo posterior estimates of the continuous model parameter of four chains for the HSMM in section 4. Parameter used to generate sample data are shown as dashed horizontal lines.

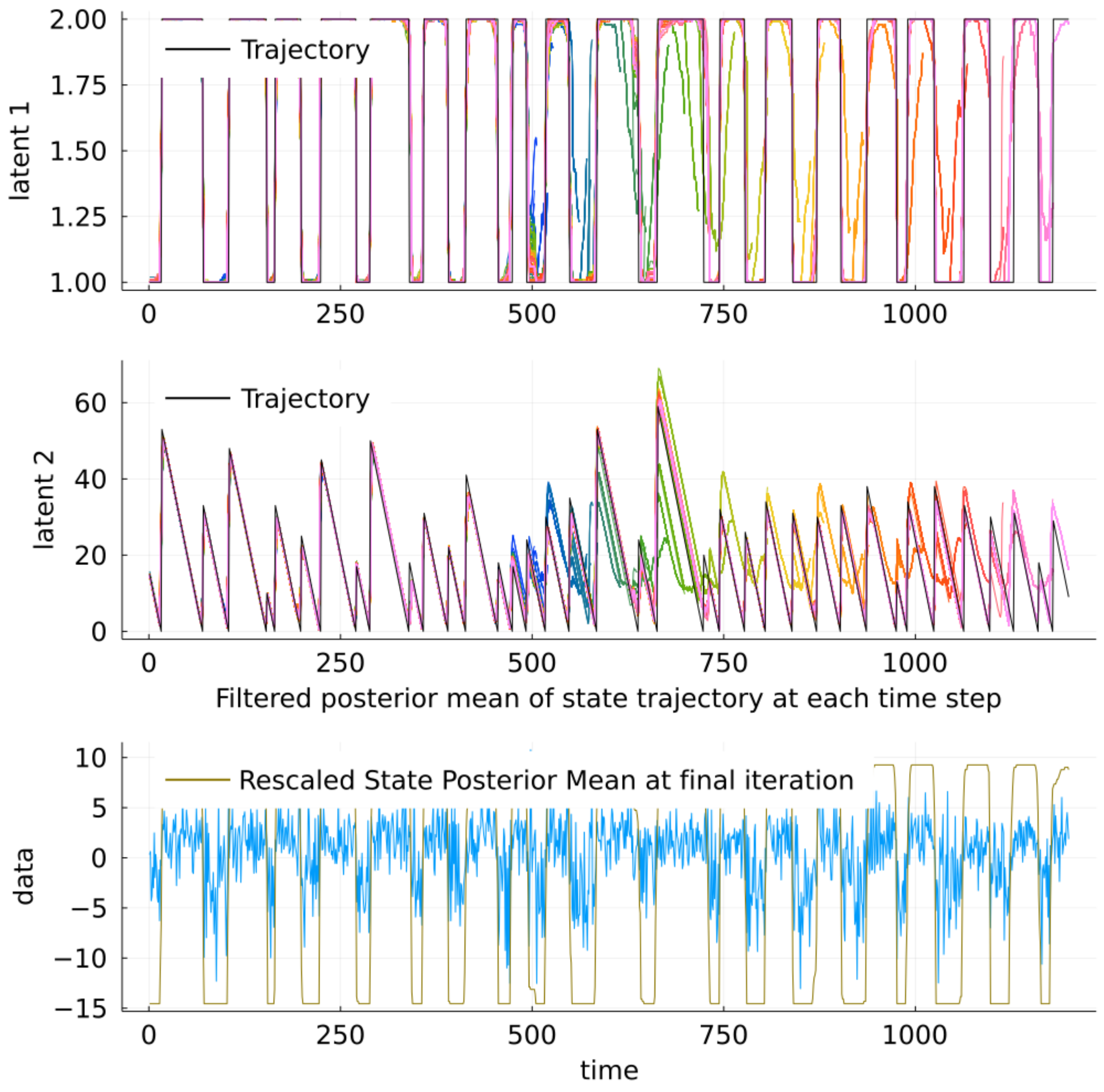

Figure 12: SMC latent. Sequential Monte Carlo posterior estimates of the filtered latent state trajectory of four chains for the HSMM in section 4. Parameter used to generate sample data are shown as dashed horizontal lines.

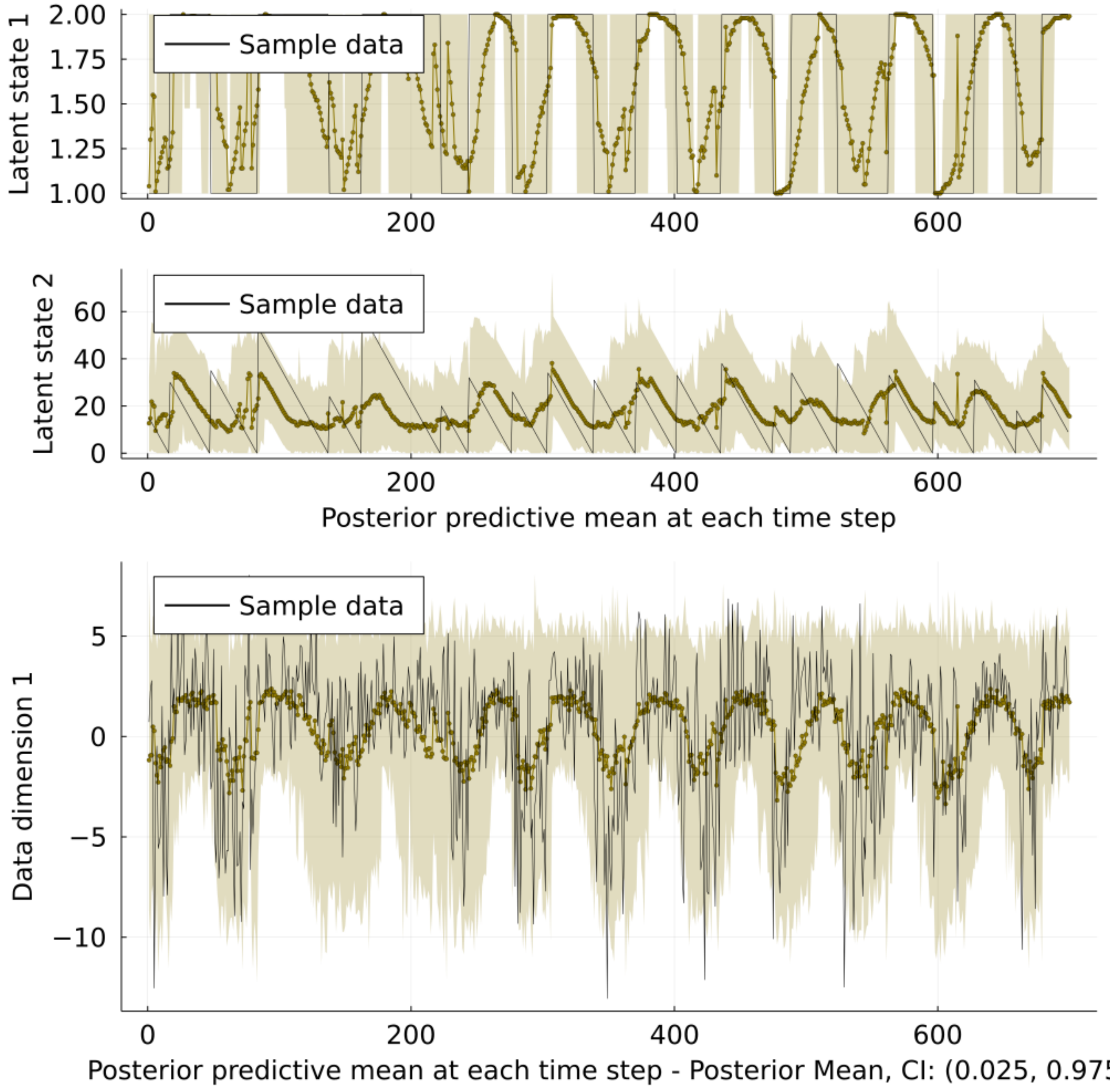

Figure 13: SMC predictions. Sequential Monte Carlo posterior predictive samples for the HSMM in section 4. Black line depicts the realized future value against predictions in gold. The top 2 graph are predictions for the state and duration variables, and the bottom plot shows predictions for the observed data.

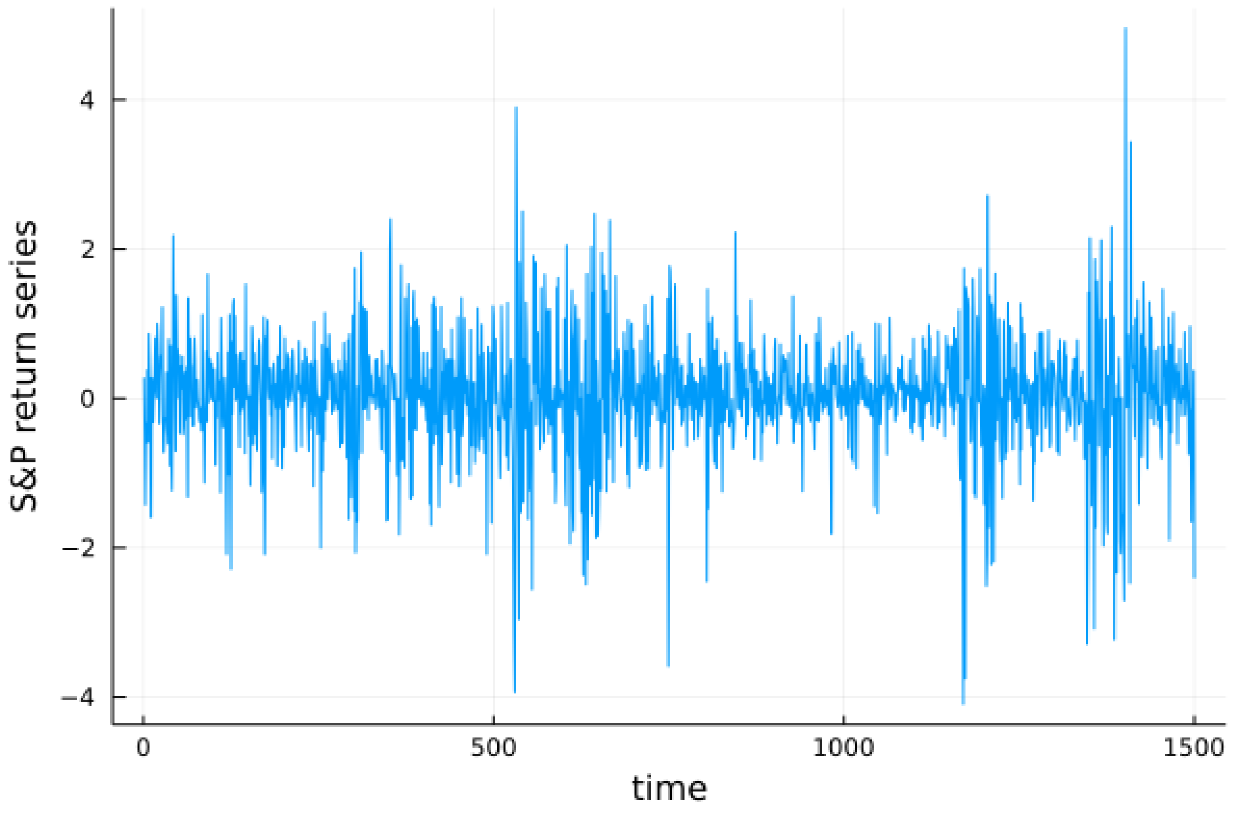

Figure 14: S&P500 total return return data as of October 1st 2021.

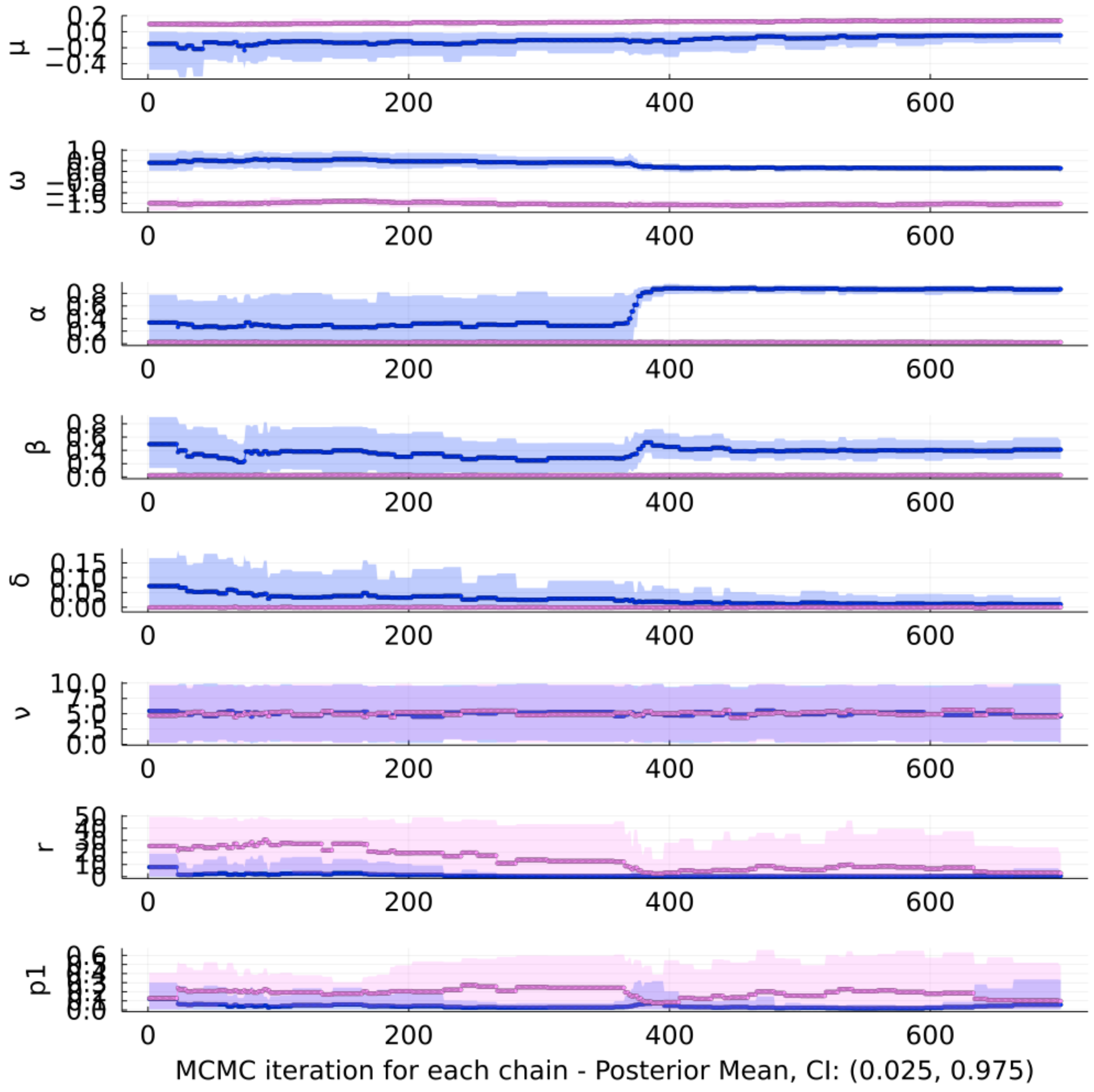

Figure 15: SMC2 Chain. Sequential Monte Carlo posterior estimates of the continuous model parameter of four chains for the HSMM with eGARCH volatility dynamics in section 5. S&P500 return data used as input.

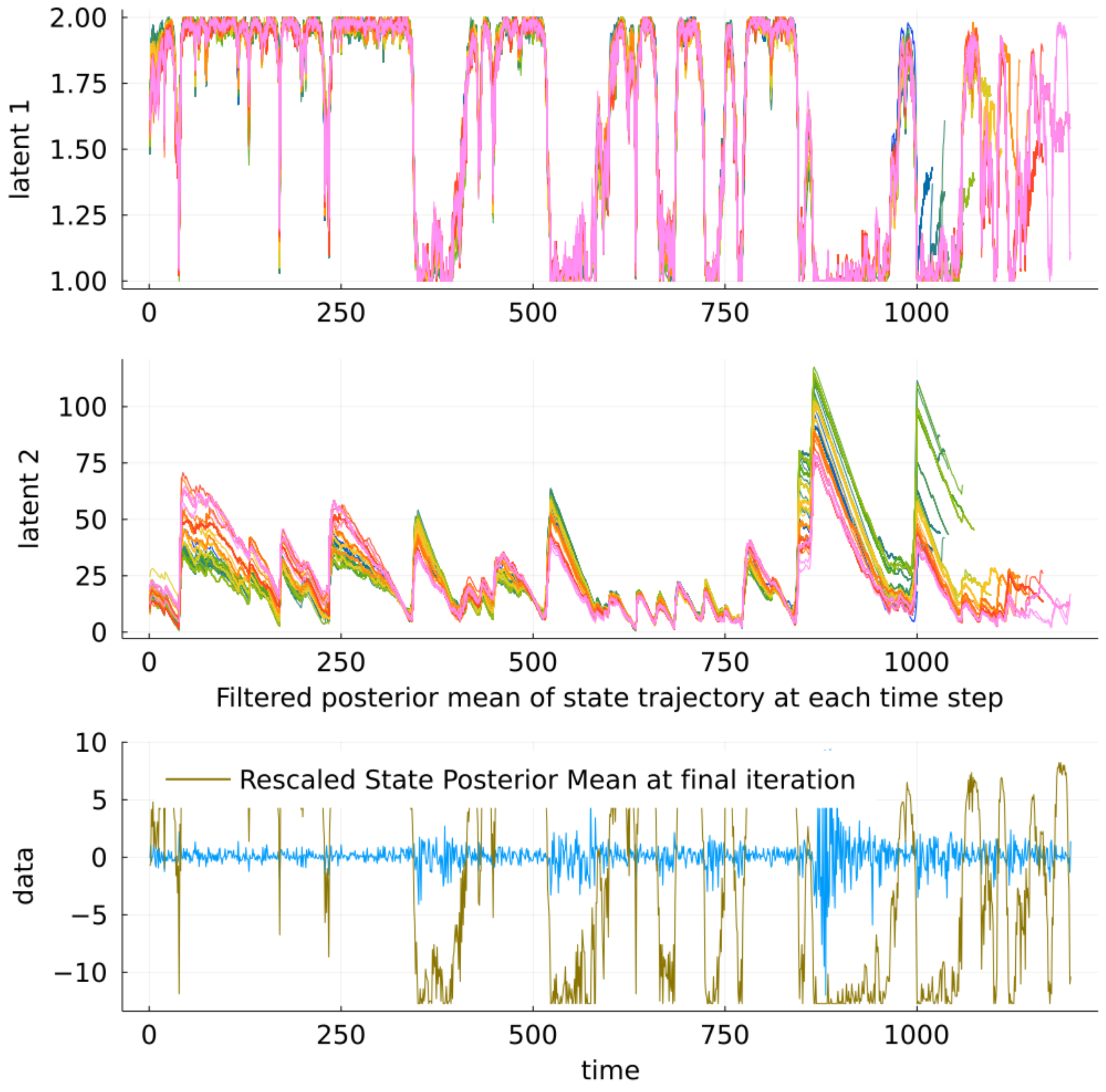

Figure 16: SMC2 Latent. Sequential Monte Carlo posterior estimates of the filtered latent state trajectory of four chains for the HSMM with eGARCH volatility dynamics in section 5. S&P500 return data used as input

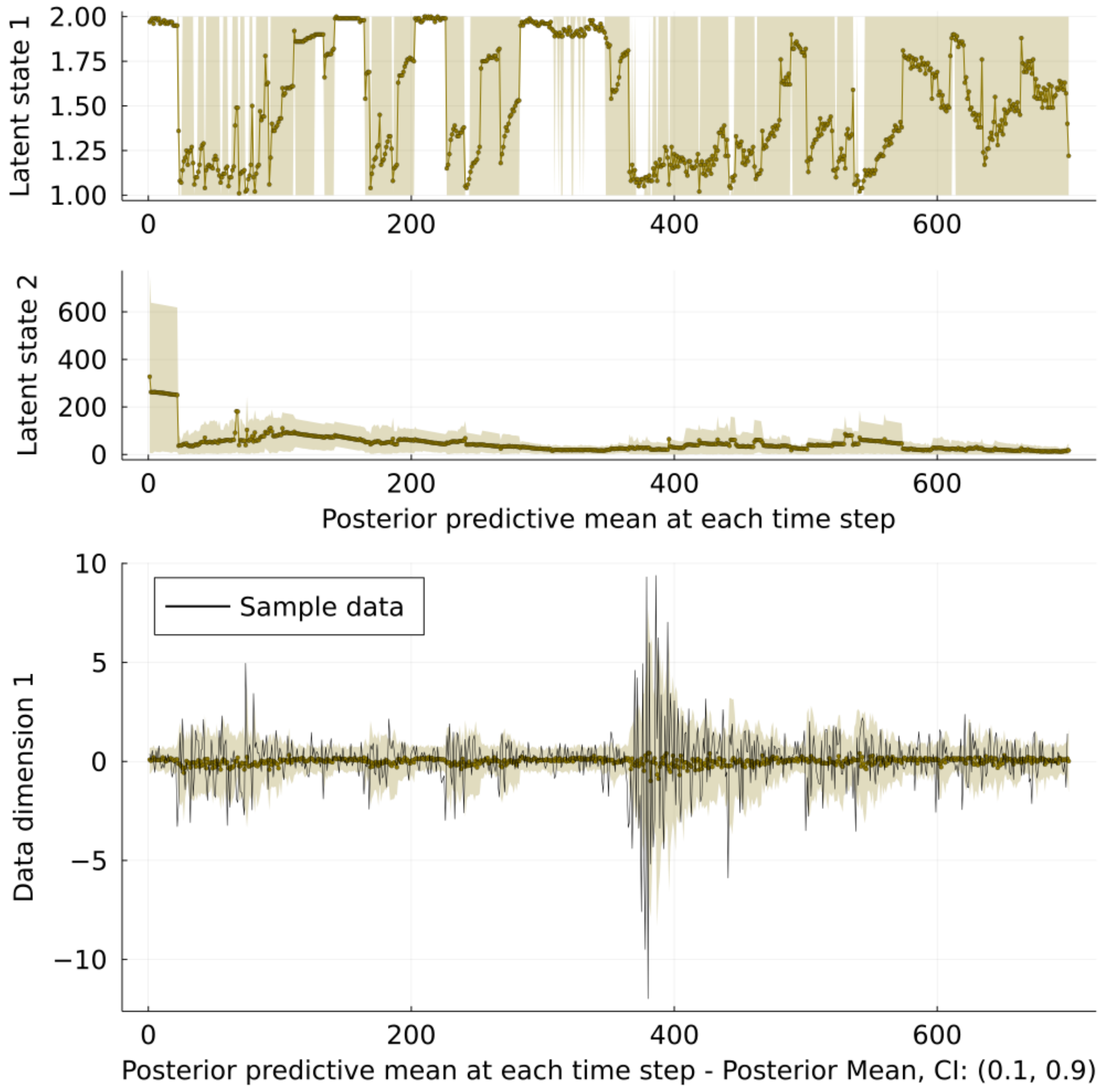

Figure 17: SMC2 Predictions. Sequential Monte Carlo posterior predictive samples for the HSMM with eGARCH volatility dynamics in section 5. Black line depicts the realized future value against predictions in gold. The top 2 graph are predictions for the state and duration variables, and the bottom plot shows predictions for the observed data.

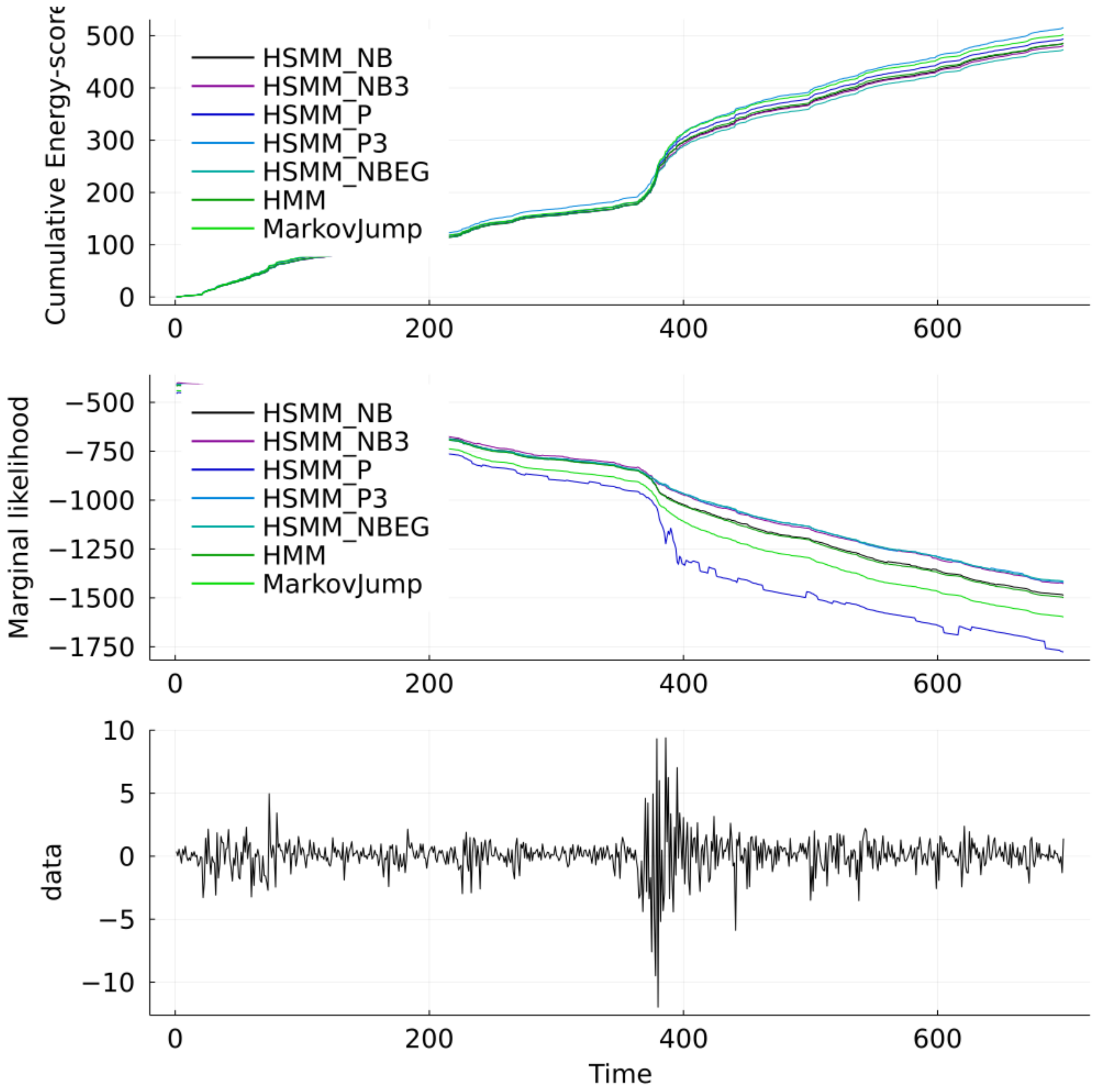

Figure 18: Cumulative e-score for various models at the top graph over time for the models used to fit the financial data described in section 5. At the bottom, the corresponding

References

C. Andrieu and G. O. Roberts. The pseudo-marginal approach for efficient monte carlo computations. *Ann. Statist.*, 37(2):697–725, 04 2009. doi: 10.1214/07-AOS574. URL `https://doi.org/10.1214/07-AOS574`.

C. Andrieu, A. Doucet, and R. Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society Series B*, 72(3):269–342, 2010. URL `https://EconPapers.repec.org/RePEc:bla:jorssb:v:72:y:2010:i:3:p:269-342`.

L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37:1554–1563, 1966. doi: http://dx.doi.org/10.1214/aoms/1177699147.

Michael Betancourt. Identifying the optimal integration time in hamiltonian monte carlo, 2016.

Michael Betancourt. A conceptual introduction to hamiltonian monte carlo, 2018.

Alexander Buchholz, Nicolas Chopin, and Pierre E. Jacob. Adaptive tuning of hamiltonian monte carlo within sequential monte carlo, 2020.

I. Bulla and J. Bulla. Stylized facts of financial time series and hidden semi-markov models. *Computational Statistics & Data Analysis*, 51:2192–2209, 2006. doi: http://EconPapers.repec.org/RePEc:eee:csdana:v:51:y:2006:i:4:p:2192-2209.

James Carpenter, Peter Cliffordy, and Paul Fearnhead. An improved particle filter for non-linear problems. *IEE Proc. Radar, Sonar Navig.*, 146:2–7, 09 2000.

S. Chib. Marginal likelihood from the gibbs output. *Journal of the American Statistical Association*, 90 (432):1313–1321, 1995. URL `https://amstat.tandfonline.com/doi/abs/10.1080/01621459.1995.10476635`.

Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 08 2002. ISSN 0006-3444. doi: 10.1093/biomet/89.3.539. URL `https://doi.org/10.1093/biomet/89.3.539`.

Nicolas Chopin and Omiros Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer International Publishing, 2020. ISBN 978-3-030-47844-5. doi: 10.1007/978-3-030-47845-2.

Nicolas Chopin, Pierre E. Jacob, and Omiros Papaspiliopoulos. Smc2: an efficient algorithm for sequential analysis of state-space models, 2013.

Adrien Corenflos, James Thornton, George Deligiannidis, and Arnaud Doucet. Differentiable particle filtering via entropy-regularized optimal transport, 2021.

Radu V. Craiu and Jeffrey S. Rosenthal. Bayesian computation via markov chain monte carlo. *Annual Review of Statistics and Its Application*, 1(1):179–201, 2014. doi: 10.1146/annurev-statistics-022513-115540. URL `https://doi.org/10.1146/annurev-statistics-022513-115540`.

Dan Crisan and Joaquin Miguez. Nested particle filters for online parameter estimation in discrete-time state-space markov models, 2017.

Remi Daviet. Inference with hamiltonian sequential monte carlo simulators, 2018.

M. Dewar, C. Wiggins, and F. Wood. Inference in hidden markov models with explicit state duration distributions. *IEEE Signal Processing Letters*, 19:235–238, Apr 2012. doi: 10.1109/LSP.2012.2184795.

R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering, 2005.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later, 2011.

Paul Fearnhead and Benjamin M. Taylor. An adaptive sequential monte carlo sampler, 2010.

S. Fruehwirth-Schnatter. *Finite mixture and Markov switching models*. Springer, 1st edition, 2006.

Sylvia Fruehwirth-Schnatter, Gilles Celeux, and Christian P. Robert. *Handbook of Mixture Analysis*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, Nov 2018.

Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: a language for flexible probabilistic inference. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pages 1682–1690, 2018. URL `http://proceedings.mlr.press/v84/ge18b.html`.

Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *The Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

M. J. Johnson. *Bayesian Time Series Models and Scalable Inference*. PhD thesis, MIT, 2014.

M. J. Johnson and A. S. Willsky. Bayesian nonparametric hidden semi-markov models. *Journal of Machine Learning Research*, 14:673–701, 2013. doi: https://arxiv.org/abs/1203.1365.

Alexander Jordan, Fabian Krüger, and Sebastian Lerch. Evaluating probabilistic forecasts with scoringrules. *Journal of Statistical Software, Articles*, 90(12):1–37, 2019. ISSN 1548-7660. doi: 10.18637/jss.v090.i12. URL `https://www.jstatsoft.org/v090/i12`.

Nikolas Kantas, Arnaud Doucet, Sumeetpal S. Singh, Jan Maciejowski, and Nicolas Chopin. On particle methods for parameter estimation in state-space models. *Statistical Science*, 30(3):328–351, Aug 2015. ISSN 0883-4237. doi: 10.1214/14-sts511. URL `http://dx.doi.org/10.1214/14-STS511`.

Gregor Kastner. Dealing with stochastic volatility in time series using the r package stochvol. *Journal of Statistical Software, Articles*, 69(5):1–30, 2016. ISSN 1548-7660. doi: 10.18637/jss.v069.i05. URL `https://www.jstatsoft.org/v069/i05`.

Fredrik Lindsten and Thomas Schön. Backward simulation methods for monte carlo statistical inference. *Foundations and Trends in Machine Learning*, 6:1–142, 01 2013. doi: 10.1561/2200000045.

Fredrik Lindsten, Michael I. Jordan, and Thomas B. Schön. Particle gibbs with ancestor sampling, 2014.

Fredrik Lindsten, Pete Bunch, Sumeetpal S. Singh, and Thomas B. Schön. Particle ancestor sampling for near-degenerate or intractable state transition models, 2015.

James E. Matheson and Robert L. Winkler. Scoring rules for continuous probability distributions. *Management Science*, 22(10):1087–1096, 1976. URL `https://EconPapers.repec.org/RePEc:inm:ormnsc:v:22:y:1976:i:10:p:1087-1096`.

Alexander McNeil, Ruediger Frey, and Paul Embrechts. *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton University Press, USA, 2005.

Kevin Murphy. Hidden semi-markov models (hsmms), 01 2002.

Radford M. Neal. Mcmc using hamiltonian dynamics, 2012.

M. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999. URL `http://www.jstor.org/stable/2670179`.

Jennifer Pohle, Timo Adam, and Larissa T. Beumer. Flexible estimation of the state dwell-time distribution in hidden semi-markov models, 2021.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. doi: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.2084.

Judith Rousseau and Kerrie Mengersen. Asymptotic behaviour of the posterior distribution in overfitted mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(5):689–710, 2011. doi: https://doi.org/10.1111/j.1467-9868.2011.00781.x.

John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. Probabilistic programming in python using PyMC3. *PeerJ Computer Science*, 2:e55, apr 2016. doi: 10.7717/peerj-cs.55. URL `https://doi.org/10.7717/peerj-cs.55`.

Stan Development Team. Stan modeling language users guide and reference manual, 2021. URL `http://mc-stan.org/`.

Gian Marco Visani, Alexandra Hope Lee, Cuong Nguyen, David M. Kent, John B. Wong, Joshua T. Cohen, and Michael C. Hughes. Approximate bayesian computation for an explicit-duration hidden markov model of covid-19 hospital trajectories, 2021.

S. Yu. *Hidden Semi-Markov Models: Theory, Algorithms and Applications*. Elsevier, Boston, 2016.

Shun-Zheng Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215–243, 2010. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2009.11.011. URL `https://www.sciencedirect.com/science/article/pii/S0004370209001416`. Special Review Issue.