Statistical Methods 2 Group Project: Sequential Monte Carlo

Ben Anson
School of Mathematics
University of Bristol
ben.anson@bristol.ac.uk

Sam Bowyer
School of Mathematics
University of Bristol
sam.bowyer@bristol.ac.uk

Emma Ceccherini
School of Mathematics
University of Bristol
emma.ceccherini@bristol.ac.uk

Abstract

Sequential Monte Carlo (SMC) methods, also known as particle filters, are a family of algorithms that are commonly used to solve filtering problems in state space models. They are particularly useful when no analytic solution is possible, such as in a generic non-linear non-Gaussian continuous state space model, representing the posterior distribution of a latent Markov process, given noisy observations, via a set of particles and associated likelihood weights. In this report we motivate and implement one of the simplest SMC algorithms, the bootstrap filter. We apply the bootstrap filter to two inference problems, one with simulated data and one with real data, and discuss the details of an efficient, parallelised implementation in Rcpp. For the real data, we implement SMC on the cases and deaths from COVID-19 in the UK from January 2020 to May 2022.

1 Introduction

State space models are extremely versatile and have various applications, such as econometrics and signal processing. They possess significant descriptive capabilities, although they are challenging to handle analytically. Unless dealing with specific simple scenarios (finite state space, linear Gaussian model), it is impossible to obtain analytic solutions to the inference problems of interest. Sequential Monte Carlo (SMC) methods, also known as "particle methods" solve this by providing approximate solutions to these intractable inference problems. In general these methods work by updating and propagating a set of random samples sequentially through distributions depending on our model observations. The desired posterior density can then be represented by the final set of particles obtained in this way, along with associated likelihood weights. Benefits of SMC samplers compared to other inference methods (such as standard Markov chain Monte Carlo (MCMC) methods (1) or variational inference (2)) include a robustness to multimodality, easily computable estimates of model evidence (which can be used in Bayesian model choice), and an easily parallelisable nature.

The first SMC method is generally regard to be that of the bootstrap filter, presented in 1993 by (3) and implemented in this report. However, in the time since, much work has been done to improve upon and expand the particle methods literature. One popular extension is the auxiliary particle filter (4), which introduces an extra sampling step in the updating of particles. Much work has also been done on combining SMC with MCMC methods, such as Sequential Hamiltonian Monte Carlo (5) and a range of particle MCMC methods provided by (6) based on Gibbs sampling and Metropolis-Hastings algorithms. The sequential nature of particle methods also easily lends itself to online learning problems (e.g. (7; 8)), whilst their general versatility has also led to them being applied in a wide variety of problems, such as motion tracking in video (9), Gaussian process kernel selection (10) and the training of neural networks (11).

There are many other approaches using recursive Bayesian estimation to solve the filtering problem, in which we aim to find the posterior distribution over a latent process at a time T given noisy observations from the process up to that time T. With a hidden Markov model (HMM) we can use the forward algorithm (12) for this (and proceed to smoothing and parameter estimation with the forward-backward algorithm and Baum-Welch algorithm respectively, the latter being a special case of the Expectation-Maximisation algorithm); however, this assumes a discrete latent state space which greatly limits the problems that can be tackled. Kalman filters (13) can be used in continuous state spaces, however, they are limited by the assumptions of linear transitions in the latent process and Gaussian observation noise. In comparison, particle filters are able to fit non-linear models with non-Gaussian observation noise, however, they are often more computationally expensive than Kalman filters.

2 Methods

This section is heavily based on (6).

2.1 State space Models

A state space model, also known as a hidden Markov model, is composed of two components: a Markov Chain of hidden variables and observed variables related to the hidden ones. These two components can be represented by two equations (a diagram is also given in Fig. 1):

- **State equation** $\{X_n; n \geq 1\} \subset \mathcal{X}^{\mathbb{N}}$ is an hidden Markov state process with initial probability $X_1 \sim \mu_\theta(\cdot)$ and transition probability $X_{n+1}|(X_n = x) \sim f_\theta(\cdot|x)$ for some static parameter $\theta \in \Theta$.
- **observation equation** $\{Y_n; n \geq 1\} \subset \mathcal{Y}^{\mathbb{N}}$ is an observation process where the observations are conditionally independent given $\{X_n; n \geq 1\}$ and the marginal probability density is of the form $Y_n|(X_n = x) \sim g_\theta(\cdot|x)$.

The likelihood can be easily obtained by exploiting the conditional independence assumption

$$p_\theta(y_{1:T}|x_{1:T}) = \prod_{n=1}^{T} g_\theta(y_n|x_n)$$

for some $T \geq 1$.

Our aim is to estimate $\{X_n; n \geq 1\}$ given $\{Y_n; n \geq 1\}$ with Bayesian inference. Namely we want the posterior density

$$p_\theta(x_{1:T}|y_{1:T}) \propto p_\theta(x_{1:T},_{1:T})$$

where

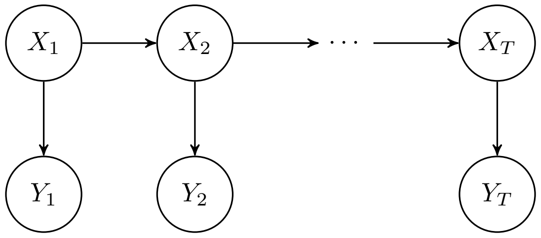
$$p_\theta(x_{1:T},_{1:T}) = \mu_\theta(x_1) \prod_{n=2}^{T} f_\theta(x_n|x_{n-1}) \prod_{n=1}^{T} g_\theta(y_n|x_n).$$

Finally, if θ is unknown we perform inference on the joint density

$$p(\theta, x_{1:T}|y_{1:T}) \propto p_\theta(x_{1:T},_{1:T})p(\theta)$$

where $p(\theta)$ is the prior for θ.

Often, if our model is non-linear and non-gaussian, there is no closed-form solution for the posterior distribution and we need to resort to approximate solutions.



$$\begin{tikzcd} X_1 \arrow[r] \arrow[d] & X_2 \arrow[r] \arrow[d] & \cdots \arrow[r] & X_T \arrow[d] \\ Y_1 & Y_2 & & Y_T \end{tikzcd}$$

Figure 1: Diagram of a state space model with latent process $X = (X_1, ..., X_T)$ and observations $Y = (Y_1, ..., Y_T)$.

2.2 Sequential Monte Carlo

When the target distribution is multivariate and non-standard classic Monte Carlo methods can't be implemented. Additionally, even in restricted cases where it is possible to sample directly from the posterior, the computational complexity of such a sequential sampling scheme would increase linearly with n. Markov chain Monte Carlo methods can be used to solve these challenges in some situations but are not well suited to sequential problems. Examples are time series analysis, online learning, and filtering problems.

Sequential Monte Carlo (SMC) methods are a broad category of Monte Carlo techniques that enable approximate sampling from a sequence of posterior probability densities $\{p(\theta, x_{1:n}|y_{1:n}); n \geq 1\}$ and a sequence of marginal likelihoods $\{p(\theta, y_{1:n}); n \geq 1\}$ in a sequential manner using a collection of N weighted random samples referred to as particles. Therefore these methods are well-suited for sequential problems.

The key component of SMC methods is sequential importance sampling (SIS) which is a sequential extension of classic importance sampling. In sequential importance sampling, the proposal distribution is defined sequentially and the weights are evaluated sequentially, to compute an estimate of the target distribution at time t using the past simulated values. At step 1, we approximate the first filtering distribution $p_\theta(x_1|y_1)$ using importance sampling with proposal density $q_\theta(x_1|y_1)$ and normalized importance weights $\{W_1^k\}$. Then the normalized importance sampling (NIS) approximation of $p_\theta(x_1|y_1)$ is given by

$$\hat{p}_\theta(dx_1|y_1) = \sum_{k=1}^{N} W_1^k \delta_{X_1^k}(dx_1).$$

Similarly, an estimator of the marginal likelihood $p_\theta(y_1)$ can be obtained. At step 2, we aim to approximate $p_\theta(x_{1:2}|y_{1:2})$. We have that:

$$p_\theta(x_{1:2}|y_{1:2}) \propto p_\theta(x_1|y_1) f_\theta(x_2|x_1) g_\theta(y_2|x_2).$$

To reuse the sample and the proposal from step one we introduce a new proposal distribution of the form $q_\theta(x_{1:2}|y_{1:2}) = q_\theta(x_1|y_1) q_\theta(x_2|y_{1:2}, x_1)$. Then we sample $X_2^k|X_1^k$ from $q_\theta(x_2|y_{1:2}, X_1^k)$ with associated normalized importance weights $\{W_2^k\}$ and we set $X_{1:2}^k = (X_1^k, X_2^k)$. Then the approximation of $p_\theta(x_{1:2}|y_{1:2})$ is given by

$$\hat{p}_\theta(dx_{1:2}|y_{1:2}) = \sum_{k=1}^{N} W_2^k \delta_{X_{1:2}^k}(dx_{1:2}).$$

These steps are repeated until time T.

Although the problem with computational complexity is solved, SIS has some severe drawbacks. The main one is weight degeneracy: as n grows, the weights degenerate until only a few particles have a significant weight. This leads to the variance of the estimates growing with n. This problem can be solved by introducing a resampling component. These methods are called Sequential Monte Carlo (SMC) methods or particle filters.

Resampling is the key ingredient of particle methods because it prevents the accumulation of errors by getting rid of samples with low weights. The idea is that particles with low weight at step n will have

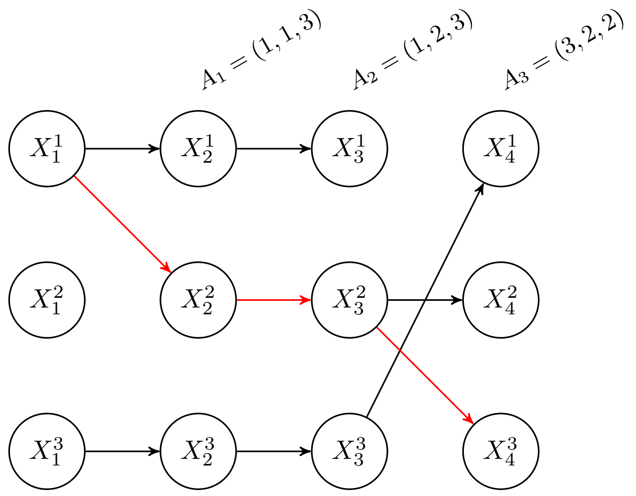


Figure 2: Example of the ancestry for a particle filter with $N = 3, T = 4$. The particles in the red path represent $X_{1:4}^3 = (X_1^1, X_2^2, X_3^2, X_4^3)$ with ancestral lineage $B_{1:4}^3 = (1, 2, 2, 3)$.

lower weights in the following steps leading to a growing number of samples that don't contribute to the estimates. This partially solves the weight degeneracy. Formally, a general sequential Monte Carlo method is given in Algorithm 1.

Algorithm 1 A sequential Monte Carlo algorithm

```
Sample $X_1^k \sim q_\theta(\cdot|y_1)$
Compute the normalized weights:
    $$w_1(X_1^k) := \frac{p_\theta(X_1^k, y_1)}{q_\theta(X_1^k, y_1)}$$
    $$W_1^k := \frac{w_1(X_1^k)}{\sum_{m=1}^N w_1(X_1^m)}$$
for n = 1,...,T do
    Sample $A_{n-1^k} \sim \mathcal{F}(\cdot|W_{n-1})$
    Sample $X_n^k \sim q_\theta(\cdot|y_n, X_{n-1}^{A_{n-1}^k})$ and set $X_{1:n}^k := (X_{n-1}^{A_{n-1}^k}, X_n^k)$
    Compute the normalized weights:
        $$w_1(X_1^k) := \frac{p_\theta(X_{1:n}^k, y_{1:n})}{p_\theta(X_{1:n-1}^{A_{n-1}^k}, y_{1:n-1}) q_\theta(X_n^k|y_n, X_{n-1}^{A_{n-1}^k})}$$
        $$W_n^k := \frac{w_n(X_{1:n}^k)}{\sum_{m=1}^N w_n(X_{1:n}^m)}.$$
end for
```

In this algorithm, the meaning of the index k is for all $k \in \{1, ..., N\}$ The variable A_{n-1}^k represents the index of the parent at time $n - 1$ of particle X_{n-1}^k; this genealogy of particles is shown in Figure 2.

The numerous particle filter methods differ in the choice of proposal distribution and resampling method. The bootstrap filter (3) was the first particle filter method to be proposed, it simply uses multinomial sampling and the latent transition prior as the proposal. Another option is the auxiliary particle filter (4) which uses the predictive likelihood and has an extra resampling step.

3 Experiments

In the previous section, we described a generic SMC algorithm. For our experiments, we use a bootstrap filter, in which the proposal distribution is the latent transition prior: $q_\theta(x_n|y_{1:n}, x_{n-1}) = p(x_n|x_{n-1})$.

3.1 Toy Model

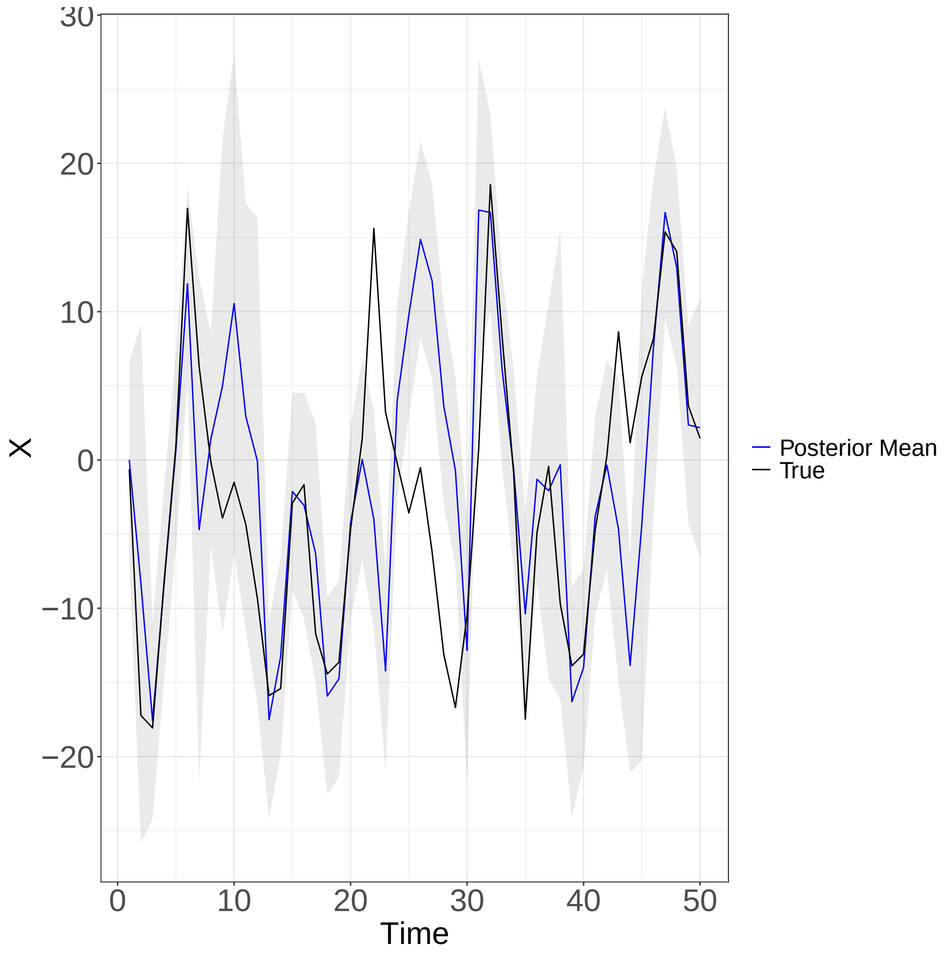


Figure 3: Posterior mean (blue), true mean (black) and error bars (grey) for particle filter on the toy model over time.

The first model we consider is given by:

$$X_n = \frac{X_{n-1}}{2} + 25\frac{X_{n-1}}{1 + X_{n-1}^2} + 8\cos(1.2n) + V_n$$

$$Y_n = \frac{X_n^2}{20} + W_n$$

where $X_1 \sim \mathcal{N}(0, \sigma_X^2)$, $V_n \sim^{\text{IID}} \mathcal{N}(0, \sigma_V^2)$ and $W_n \sim^{\text{IID}} \mathcal{N}(0, \sigma_W^2)$. This is a popular toy SSM in the literature (3; 6) because the sign-uncertainty caused by observations of the squared latent value leads to a highly multimodal posterior $p(x_{1:T}|y_{1:T})$.

We generate a realisation of this model with $\sigma_X^2 = 1, \sigma_V^2 = 10, \sigma_W^2 = 3$ and apply our bootstrap filter, plotting the results in Fig. 3.

This plot shows the posterior mean of the Bootstrap particle filter with confidence intervals which are $\pm 2\hat{\sigma}$. We can see that the true mean is always inside the error bars except for a couple of time points. The posterior mean matches closely the true mean up to sign. For example, at time $t = 24$ the posterior mean is the correct magnitude but the wrong sign, which is a common behaviour for this model.

3.2 Covid Model

We next apply our bootstrap filter to a model describing Covid-19 infections (14) that extends the susceptible-infectious-recovered (SIR) model by adding the mortality rate. The SIRD model divides the population into four groups: the susceptible, the infected, the recovered and the deaths, with each group described by one equation. The SIRD model assumes:

$$\begin{aligned}
S_t &= \frac{\exp(s_t)}{1+\exp(s_t)+\exp(i_t)} \\
I_t &= \frac{\exp(i_t)}{1+\exp(s_t)+\exp(i_t)} \\
\alpha_t &= \frac{\exp(a_t)}{1+\exp(a_t)} \\
\beta_t &= \frac{\exp(b_t)}{1+\exp(b_t)} \\
\mu_t &= \frac{1}{2}\frac{\exp(m_t)}{1+\exp(m_t)}
\end{aligned}$$

where S_t and I_t are the proportion of susceptible and infected, respectively; and α_t, β_t and μ_t are the reporting, the infection, and the mortality rate, respectively. A logit transformation is used to ensure that each rate is bounded in the interval $[0, 1]$. This model can be seen as a state-space model, the observation equations are:

$$\begin{aligned}
\log\left(\frac{NC_t}{\text{Pop}}\right) &= \log(\alpha_t) + \log(\beta_t) + \log(S_t) + \log(I_t) + w_t^1 \\
\log\left(\frac{ND_t}{\text{Pop}}\right) &= \log(\gamma) + \log(\mu_t) + \log(I_t) + w_t^2
\end{aligned}$$

where 'Pop' is the total population which is assumed to be constant over time, NC_t is the number of new registered cases at time t, ND_t is the of deaths at time t and w_t is zero mean Gaussian noise. The state equations are:

$$\begin{aligned}
s_{t-1} &= s_{t-2} + \log\left(1 - \frac{\beta_t \exp(i_{t-2})}{1+\exp(s_{t-2})+\exp(i_{t-2})}\right) - \log(1-\gamma\exp(i_{t-2})) + v_{1,t}, \\
i_{t-1} &= i_{t-2} + \log\left(1 - \frac{\beta_t \exp(s_{t-2})}{1+\exp(s_{t-2})+\exp(i_{t-2})}\right) - \log(1-\gamma\exp(i_{t-2})) + v_{2,t}, \\
a_t &= a_{t-1} + v_{3,t}, \\
b_t &= b_{t-1} + v_{4,t}, \\
m_t &= m_{t-1} + v_{5,t},
\end{aligned}$$

where v_t is a zero mean Gaussian noise independent of w_t and γ is the removal rate, which is assumed to be constant over time.

This model is applied to Covid-19 cases (15) and deaths (16) from January 30th 2020 to May 19th 2022 in the United Kingdom. Specifically, we consider the log proportion of the population who had a positive test on each day and the log proportion of the population who died on each day. We implemented the SIRD model (14) with $\gamma = 1/31$ and a normal prior on the initial states with mean $(3, -5, -3, -3, 0.5)$ and diagonal covariance matrix with diagonal elements equal to 2. The observation noise vector w is a zero mean Gaussian with a diagonal covariance matrix with diagonal entries equal 0.1 and the process noise vector v is a zero mean Gaussian with a diagonal covariance matrix with diagonal entries equal 0.003.

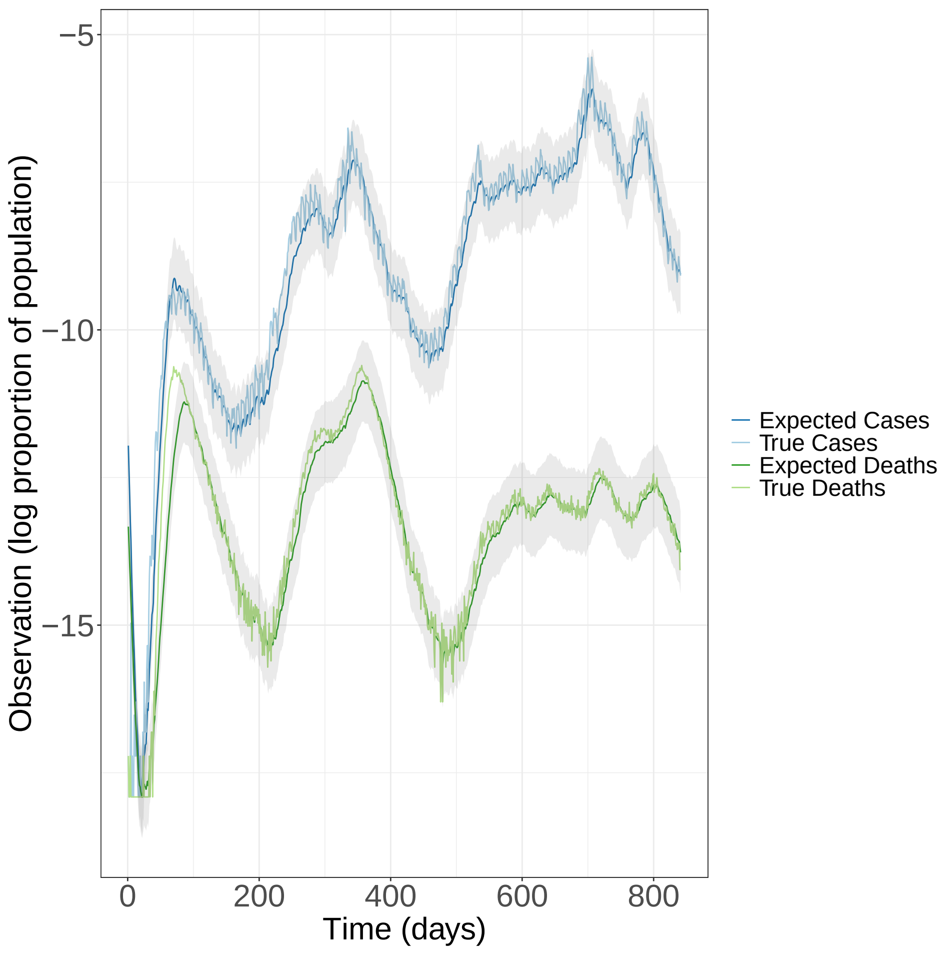


Figure 4: Posterior mean (dark green) with confidence intervals (grey) and the true values (light green) for deaths over time. Posterior mean (dark blue) with confidence intervals (grey) and the true values (light blue) for cases over time.

Figure 4 show the resulting posterior mean with confidence intervals and the true values for cases and deaths. From this plot, we can see that the posterior mean roughly coincides with the true values both for cases and deaths. The expected deaths are marginally different from the true deaths before $t = 100$, this is possibly explained by the fact that the model didn't have enough information to produce satisfactory results yet. Indeed, the performances of SMC improve over time, as the model accumulates information.

3.3 Implementation Details

The SMC algorithm outlined in Section 2.2 was implemented in `R` and C++ using `RcppParallel`, via a package provided at `https://github.com/lippirk/smc`. The parallelisation is done during the resampling step, where each proposal sample can be generated independently.

Below in Table 1 we show the time taken to run the bootstrap filter in our toy example from section 3.1 for varying numbers of particles, N, in different implementations: one in serial `R`, one in serial `Rcpp` and one with parallel `Rcpp` with different numbers of cores. In particular, we can observe that for $N = 10$ the serial `Rcpp` implementation achieves the fastest runtime whilst for larger N we observe moderate efficiency gains with the parallel implementation, increasing with the number of threads, taking roughly two-thirds the time of the pure `R` implementation with $N = 100,000$.

	Serial		RccpParallel (No. Threads)		
N	R	Rcpp	4	16	64
10	0.0029	0.0002	0.0018	0.0034	0.0348
100	0.0074	0.0027	0.0026	0.0033	0.0369
1000	0.0360	0.0275	0.0212	0.0223	0.0538
10000	0.3324	0.3264	0.2234	0.2647	0.2644
100000	3.1785	3.7478	2.2483	2.4912	2.1894

Table 1: Mean runtime (in seconds) of SCM for the Toy model in different implementations.

4 Conclusion

In this paper we have motivated and discussed a generic sequential Monte Carlo algorithm, the bootstrap filter, and provided an efficient parallelised implementation. We used this to perform Bayesian inference on two non-linear state space models and found that we were able to obtain good estimates of the posterior distribution of the latent processes and thus also of the observations produced from these latent distributions. Furthermore, we analysed the computational cost of different SMC implementations for varying number of particles, observing gains in speed using our parallel `Rcpp` implementation when using a large number of threads and a large number of particles.

Particle methods represent an extremely active and wide area of research, and as such, this work could easily be extended to an analysis of other particle methods, such as auxiliary particle filters (4), particle Gibbs sampling (6) or Rao-Blackwellised particle filters (17). Another natural extension is to further impove the parallel `Rcpp` implementations and hopefully realise even greater gains in computational efficiency.

References

[1] W. K. Hastings, “Monte Carlo Sampling Methods Using Markov Chains and Their Applications,” 1970.

[2] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational Inference: A Review for Statisticians,” *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, Apr. 2017, arXiv:1601.00670 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1601.00670

[3] N. Gordon, D. Salmond, and A. Smith, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” in *IEE Proceedings F Radar and Signal Processing*, vol. 140, 1993, p. 107, iSSN: 0956375X Issue: 2 Journal Abbreviation: IEE Proc. F Radar Signal Process. UK. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/ip-f-2.1993.0015

[4] M. Pitt and N. Shephard, “Filtering via Simulation: Auxiliary Particle Filters,” *Journal of the American Statistical Association*, vol. 94, Nov. 1997.

[5] R. Daviet, “Inference with Hamiltonian Sequential Monte Carlo Simulators,” Dec. 2018, arXiv:1812.07978 [stat]. [Online]. Available: http://arxiv.org/abs/1812.07978

[6] C. Andrieu, A. Doucet, and R. Holenstein, “Particle Markov Chain Monte Carlo Methods,” *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 72, no. 3, pp. 269–342, Jun. 2010. [Online]. Available: https://academic.oup.com/jrsssb/article/72/3/269/7076437

[7] T. Kurihara, Y. Nakada, K. Yosui, and T. Matsumoto, “Bayesian on-line learning: a sequential Monte Carlo with importance resampling,” in *Neural Networks for Signal Processing XI: Proceedings of the 2001 IEEE Signal Processing Society Workshop (IEEE Cat. No.01TH8584)*, Sep. 2001, pp. 163–172, iSSN: 1089-3555.

[8] M. M. Zhang, B. Dumitrascu, S. A. Williamson, and B. E. Engelhardt, “Sequential Gaussian Processes for Online Learning of Nonstationary Functions,” *IEEE Transactions on Signal Processing*, vol. 71, pp. 1539–1550, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10103648/

[9] M. Abbasi and M. R. Khosravi, “A Robust and Accurate Particle Filter-Based Pupil Detection Method for Big Datasets of Eye Video,” *Journal of Grid Computing*, vol. 18, no. 2, pp. 305–325, Jun. 2020. [Online]. Available: https://doi.org/10.1007/s10723-019-09502-1

[10] A. B. Abdessalem, N. Dervilis, D. J. Wagg, and K. Worden, “Automatic Kernel Selection for Gaussian Processes Regression with Approximate Bayesian Computation and Sequential Monte Carlo,” *Frontiers in Built Environment*, vol. 3, 2017. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fbuil.2017.00052

[11] M. Halimeh, A. Brendel, and W. Kellermann, *Neural Networks Sequential Training Using Variational Gaussian Particle Filter*, May 2019, pages: 3006.

[12] L. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[13] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[14] G. M. Athayde and A. P. Alencar, “Forecasting Covid-19 in the United Kingdom: A dynamic SIRD model,” *PLoS ONE*, vol. 17, no. 8, p. e0271577, Aug. 2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9365164/

[15] U. H. S. Agency, “Coronavirus (covid-19) in the UK - Cases,” 2023. [Online]. Available: https://coronavirus.data.gov.uk/details/cases?areaType=overview&areaName=United%20Kingdom

[16] ——, “Coronavirus (covid-19) in the UK - Deaths with COVID-19 on the death certificate,” 2023. [Online]. Available: https://coronavirus.data.gov.uk/details/deaths?areaType=overview&areaName=United%20Kingdom

[17] A. Doucett, N. de Freitast, K. Murphyt, and S. Russent, “Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks,” 2000.