Week 6 & 7 - Exploring the Neuroscience model

Max Richards

2025-08-25

Summary of Week 5

Last week was a particularly interesting one, as I dedicated a lot of time to exploring more complicated state space models. A significant portion of my efforts was spent learning about and trying to fully understand the rather peculiar model presented in the original paper that first introduced the bootstrap particle filter. This model stood out both in terms of structure and motivation, and digging into it gave me valuable perspective on the foundations of particle filtering.

Overall, the week proved to be extremely useful in helping me build a deeper understanding of how these models perform in more uncertain settings. It allowed me to clearly see where the different filters I have been experimenting with either succeed or struggle, depending on the context. The set of methods I worked with included the (extended) Kalman filter, the traditional bootstrap particle filter, my own grid-based approaches, as well as the more advanced adaptive grid filters. Comparing these side by side was particularly illuminating and gave me a much clearer picture of their relative strengths and weaknesses.

Towards the end of the week, I also devoted time to studying different measures of error in the one-dimensional setting. Along with the standard RMSE, I examined the normalized RMSE, but what proved most exciting was delving into divergence-based measures of error. In particular, I worked with KL divergence and JS divergence. These were relatively unfamiliar concepts to me at the outset, but they turned out to be both fascinating and very useful, especially in the context of more complex, non-linear state space models where conventional error measures can sometimes fail to capture the full picture.

In addition to the technical work, I also had an online meeting with Adam and Rocco, who will be acting as my supervisor and main point of contact over the next two weeks or so. In this meeting, we reviewed what I have learned from the project so far, discussed areas of particular interest, and considered some logical next steps. Adam suggested that I look into a neuroscience model that combines the use of a non-linear state space model with real-world data. This immediately struck me as a particularly exciting direction to take, as it offers the chance to apply the theory I have been working with to real data in a meaningful way. Over the coming weeks, I am very much looking forward to diving deeper into this model and exploring what insights it can provide.

So what is the model and where is it from?

The model is one used in neuroscience often which defines a time-homogeneous state space, such that at each time step t, the observation y_t represents the number of activated neurons. We take it from the paper titled ***"Online Rolling Controlled Sequential Monte Carlo"*** written by *Liwen Xue, Axel Finke* and *Adam M. Johansen*.

Motivation for ORCSMC

Sequential Monte Carlo (SMC) methods, such as the bootstrap particle filter (BPF), provide a flexible framework for approximating the posterior distribution of latent states in general state-space models. In particular, the BPF recursively propagates a set of particles through the state dynamics and reweights them according to the likelihood of the observations. While simple and widely applicable, the BPF often suffers from particle degeneracy and high variance, particularly in high-dimensional or highly informative observation settings.

For linear Gaussian models, the Kalman filter (KF) provides an exact and computationally efficient solution by analytically computing the posterior mean and covariance of the latent states. However, the KF is restricted to linear dynamics with Gaussian noise and cannot be applied directly to models with nonlinear observations, such as the Binomial likelihood in the neuroscience model considered here.

The ORCSMC method extends the standard SMC framework by introducing *controlled sequential Monte Carlo* updates within a *rolling-window* online framework. At each time step, ORCSMC constructs a sequence of auxiliary *control policies* that minimize the variance of importance weights, effectively reducing particle degeneracy. By rolling this procedure over a fixed window of recent observations, ORCSMC balances computational efficiency with the accuracy of online posterior estimates, making it particularly suitable for applications with streaming or high-frequency data.

Advantages over Traditional Filters

Compared to the BPF, ORCSMC maintains higher effective sample sizes (ESS) by adaptively controlling the proposal distributions. Unlike the KF, ORCSMC can handle nonlinear and non-Gaussian observation models while still leveraging structure in the latent dynamics to improve efficiency. In the neuroscience example, the combination of nonlinear Binomial observations and temporal autocorrelation in the latent states highlights the need for a filtering method that is both flexible and variance-reducing, a role which ORCSMC fulfills effectively.

Neuroscience State-Space Model and Filtering Approach

To evaluate the performance of the proposed Online Rolling Controlled Sequential Monte Carlo (ORCSMC) method, we consider a state-space model inspired by neuroscience experiments *heng2020*. Let $\{x_t\}_{t\geq 0}$ denote the latent state process and $\{y_t\}_{t\geq 1}$ the corresponding observations.

Model Specification

In the univariate setting, each observation $y_t \in \{0, 1, \ldots, M\}$ represents the number of activated neurons observed in M repeated trials at time t, with $M = 50$. The observation model is defined as a Binomial distribution with a logistic link function:

$$y_t \mid x_t \sim \text{Binomial}(M, \kappa(x_t)), \tag{1}$$

$$\kappa(x_t) = \frac{1}{1 + \exp(-x_t)}, \tag{2}$$

where $\kappa(x_t)$ maps the latent state to the probability of neuronal activation.

The latent dynamics are specified as a linear Gaussian autoregressive process:

$$x_0 \sim \mathcal{N}(0, 1), \tag{3}$$

$$x_t \mid x_{t-1} \sim \mathcal{N}(\alpha x_{t-1}, \sigma^2), \tag{4}$$

where $\alpha = 0.99$ and $\sigma^2 = 0.11$. This formulation captures the temporal dependence of neuronal activity over consecutive trials.

For higher-dimensional extensions, we consider a multivariate state $x_t = (x_{t,1}, \ldots, x_{t,d}) \in \mathbb{R}^d$ and corresponding observations $y_t = (y_{t,1}, \ldots, y_{t,d})$. The observation likelihood factorizes across dimensions:

$$y_{t,j} \mid x_{t,j} \sim \text{Binomial}(M, \kappa(x_{t,j})), \quad j = 1, \ldots, d, \tag{5}$$

while the latent dynamics remain independent across dimensions:

$$x_{t,j} \mid x_{t-1,j} \sim \mathcal{N}(\alpha x_{t-1,j}, \sigma^2). \tag{6}$$

Simulation and Filtering of a Binomial State Space Model

In this section, we describe the simulation of a simple state space model and the implementation of three particle filtering methods to estimate the latent state: the *bootstrap filter*, the *ORC-SMC* (Optimal Resampling Conditional Sequential Monte Carlo), and the *ORC-fixed* method. We aim to illustrate both the practical and mathematical aspects of these filters.

Simulating the State Space Model

We are interested in simulating a simple state space model consisting of a latent AR(1) process observed through a binomial likelihood with a logistic link. Specifically, the latent state evolves according to:

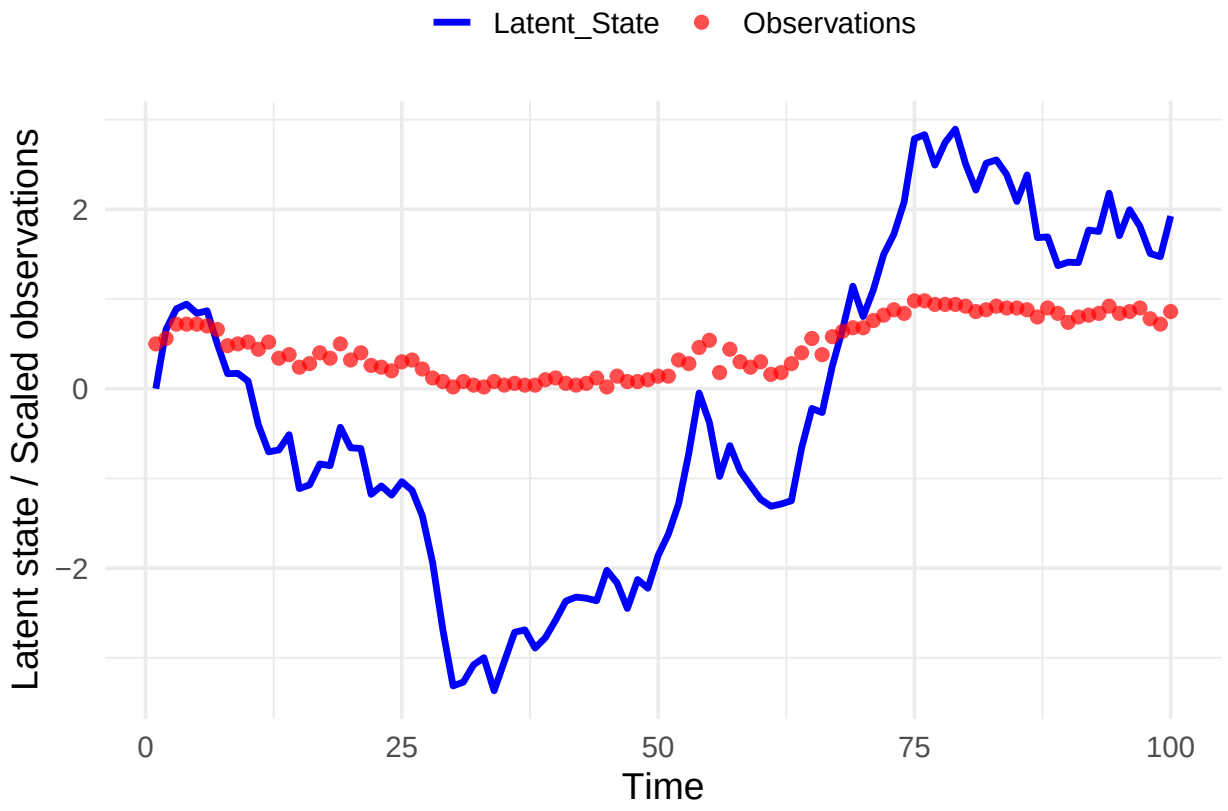
$$x_t = \alpha x_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2),$$

where we choose $\alpha = 0.99$ and $\sigma^2 = 0.11$. This defines a smooth autoregressive process for the latent state x_t.

The observations are generated as binomial counts with success probability given by a logistic transformation of the latent state:

$$y_t \sim \text{Binomial}(M, p_t), \quad p_t = \frac{1}{1 + e^{-x_t}}.$$

Here, M is the number of trials at each time step, and p_t maps the real-valued latent state to the interval $[0, 1]$. In R, we will simulate both x_t and y_t over a chosen time horizon and visualize their behavior.



Understanding the Distinct Trails of Latent States and Observations

When plotting the latent states and the scaled observations y_t/M, we often see two distinct trails:

1. **Latent State (x_t)** — The latent process is continuous and can take values across the real line, including negative numbers. Its smooth AR(1) dynamics allow gradual increases or decreases over time.
2. **Observations (y_t/M)** — The observations are counts transformed to proportions by dividing by M, and are constrained to $[0, 1]$. Even if x_t changes smoothly, the binomial randomness makes y_t/M discrete and noisy. The logistic function squashes the latent state into a probability, so extreme negative x_t values produce $y_t/M \approx 0$ and extreme positive x_t values produce $y_t/M \approx 1$.

Mathematically, the observation process introduces stochasticity as

$$\frac{y_t}{M} \sim \frac{\text{Binomial}\Big(M, \frac{1}{1+e^{-x_t}}\Big)}{M},$$

so even when x_t moves smoothly, y_t/M can appear to "stick' ' near 0 or 1 due to the discreteness and bounded nature of the binomial distribution.

As a result, the plot shows two clearly separated trails: the smooth latent trajectory and the jagged, probabilistically constrained observations.

Latent State Estimation Using ORCSMC and The Fixed ORC

In this section, we extend the previously defined binomial state-space model by implementing an *Optimal/Online Resample-Controlled Sequential Monte Carlo* (ORCSMC) method to estimate the latent states from noisy observations.

The latent process is modeled as an AR(1) Gaussian process:

$$x_t = \alpha x_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2), \quad t = 1, \ldots, T,$$

with $\alpha = 0.99$ and $\sigma^2 = 0.11$, producing a smoothly varying latent trajectory. Observations y_t are generated according to a binomial distribution with a logistic link:

$$y_t \sim \text{Binomial}(M, p_t), \quad p_t = \frac{1}{1 + e^{-x_t}}, \quad M = 50.$$

The ORCSMC algorithm approximates the posterior distribution of x_t given the observations $\{y_1, \ldots, y_T\}$ using N particles.

The Fixed ORC Approach

In the first implementation presented below, a *fixed policy function* is used to guide particle propagation, constituting a version of the so-called *ORC fixed method*. This method is a simplification of the fully adaptive ORC-SMC: while the propagation of particles uses a policy designed to steer particles toward regions of high posterior probability, the resampling schedule is predetermined rather than adaptive. This makes implementation simpler while retaining many benefits of the ORC framework.

Specifically, particles are propagated according to a fixed nudging function:

$$x_t^{(i)} \sim \mathcal{N}\Big(\alpha x_{t-1}^{(i)} + \gamma\big(\frac{y_{t-1}}{M} - \text{logistic}(x_{t-1}^{(i)})\big), \sigma^2\Big),$$

where γ is a fixed scaling parameter that adjusts the particle locations toward values of x_t that are more consistent with the previous observation y_{t-1}. Intuitively, this "nudging" implements an approximate form of the optimal proposal distribution in ORC-SMC by increasing the probability that particles lie in high-likelihood regions, without adaptively updating the policy at each time step.

After propagation, particles are assigned importance weights according to the binomial likelihood:

$$w_t^{(i)} \propto \text{Binomial}\big(y_t \mid M, \text{logistic}(x_t^{(i)})\big),$$

or equivalently in terms of the ORC-SMC optimal proposal:

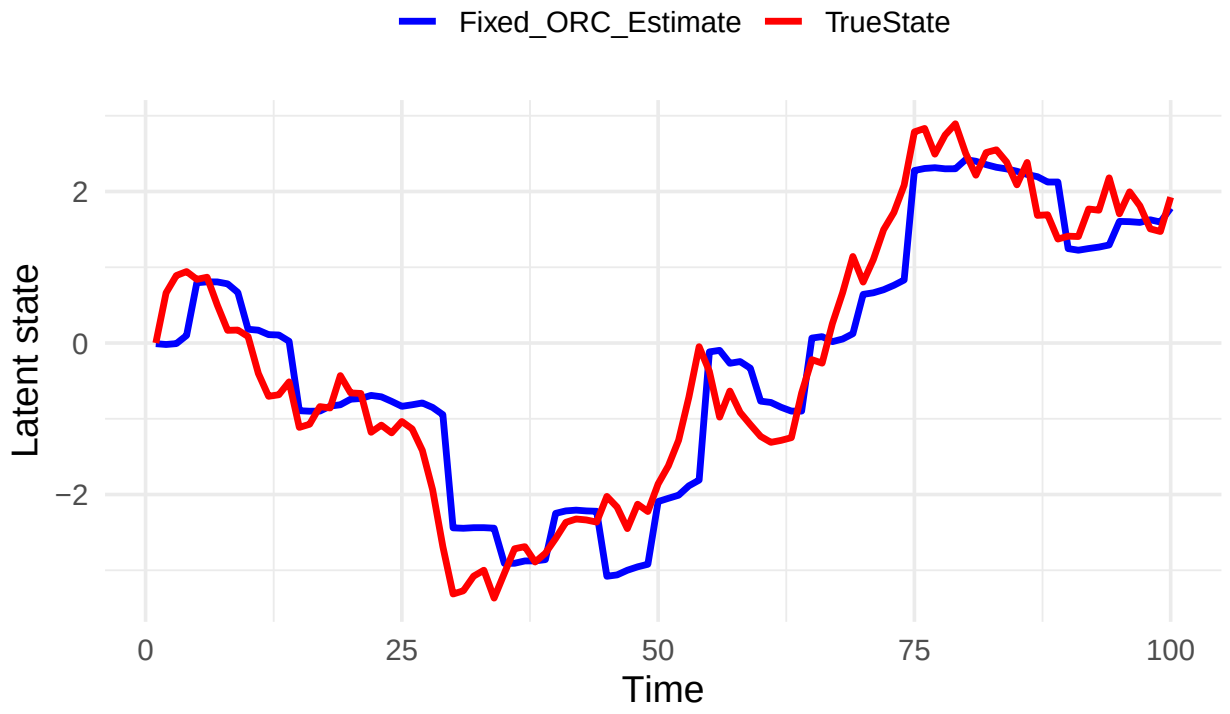
$$w_t^{(i)} \propto \frac{p(y_t \mid x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)})}{q^*(x_t^{(i)} \mid x_{t-1}^{(i)}, y_t)}, \tag{7}$$

where q^* denotes the ideal proposal distribution that incorporates the current observation. In the fixed ORC method, q^* is approximated by the fixed policy above.

Weights are then normalized and resampling is performed at predetermined intervals (every 5 time steps in my implementation) to mitigate particle degeneracy. Systematic resampling ensures that the particle cloud remains representative of the posterior distribution, while avoiding excessive variance caused by repeated resampling at every time step.

The result of this procedure is a posterior estimate of the latent state at each time step, obtained as the weighted average of the particles. Due to the nonlinear mapping of latent states to observations via the logistic function, this posterior estimate provides a smoothed approximation of the true latent trajectory, capturing both the AR(1) dynamics and the observation noise.

Overall, the fixed ORC approach allows for efficient online estimation of latent states in a probabilistic framework. It combines a principled particle nudging strategy with a simple, predetermined resampling schedule, making it suitable for applications where latent dynamics are indirectly observed through noisy measurements.



Particle Filtering Methods

Following the implementation of the simplified ORCSMC method, i.e. the Fixed ORC method, I decided to implement the real ORCSMC algorithm, as well as a bootstrap particle filter, for the sake of comparison.

Bootstrap Filter:

Having used this extensively in the past weeks, it is worth recalling the bootstrap particle filter, which remains a cornerstone in sequential Monte Carlo methods. The algorithm approximates the posterior distribution of a latent state x_t given observations $y_{1:t}$ by a set of weighted particles. At time $t-1$, suppose we have particles $\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N}$ representing the filtering distribution $p(x_{t-1} \mid y_{1:t-1})$. The bootstrap filter proceeds in two key steps:

1. **Propagation (Prediction):** Each particle is propagated forward through the state transition dynamics:
$$x_t^{(i)} \sim p(x_t \mid x_{t-1}^{(i)}) = \mathcal{N}(\alpha x_{t-1}^{(i)}, \sigma^2),$$
which provides an empirical approximation to the predictive distribution $p(x_t \mid y_{1:t-1})$.

2. **Weighting (Correction):** Particles are reweighted according to the likelihood of the new observation y_t, giving an updated approximation of $p(x_t \mid y_{1:t})$:
$$w_t^{(i)} \propto w_{t-1}^{(i)} p(y_t \mid x_t^{(i)}) = w_{t-1}^{(i)} \binom{M}{y_t} \left(p_t^{(i)}\right)^{y_t} \left(1 - p_t^{(i)}\right)^{M-y_t}, \quad p_t^{(i)} = \frac{1}{1 + e^{-x_t^{(i)}}}.$$
This step implements Bayes' theorem in a discrete, particle-based sense: the prior predictive is corrected by the observation likelihood to form the posterior.

Resampling: To mitigate weight degeneracy—where a few particles carry almost all the weight—we monitor the effective sample size,
$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (w_t^{(i)})^2},$$
and resample when it drops below a threshold. Resampling replaces low-weight particles with duplicates of high-weight particles, preserving diversity in the particle set while maintaining the empirical posterior approximation.

Overall, the bootstrap filter provides a recursive, simulation-based approximation of the filtering distribution, balancing the propagation of uncertainty through the dynamics with the assimilation of new observational information. It serves as a fundamental building block for more advanced particle filtering approaches.

ORCSMC:

The *Online Rolling Controlled Sequential Monte Carlo* (ORCSMC) is an advanced particle filtering algorithm designed to approximate the posterior distribution of latent states while reducing weight degeneracy and controlling variance over time. Unlike the *fixed ORC* approach, which applies a predetermined policy shift and resamples at fixed intervals, ORCSMC dynamically adapts its policy and resampling schedule using both past and future information within a rolling time window.

Let $x_{1:t}$ denote the latent states up to time t, and $y_{1:t}$ the corresponding observations. ORCSMC maintains a particle system H_t with N particles, each representing a possible trajectory $X_{1:t}^{(n)}$. At each time step, the algorithm proceeds as follows:

1. **Rolling window selection:** Define a rolling window of length L, and let $t_0 = \max\{1, t-L+1\}$. This window ensures that only a limited recent history is used to adapt the proposal, keeping the algorithm computationally tractable.

2. **Policy learning:** Starting from the current time t and moving backward to t_0, update the policy functions ψ_s recursively using the auxiliary particle filter:

$$\psi_s \leftarrow \text{learn-}\psi(s, \psi_{s+1}, \tilde{H}_s),$$

   where $\tilde{H}_s$ is the intermediate particle system at time s. This backward recursion incorporates future observations within the rolling window to optimize particle propagation.

3. **Particle propagation:** Using the learned policies ψ_s, particles are propagated forward from t_0 to t via a ψ-controlled auxiliary particle filter:

$$\tilde{H}_s \sim \psi\text{-apf}(s, \psi_s, \tilde{H}_{s-1}), \quad H_s \sim \psi\text{-apf}(s, \psi_s, H_{s-1}),$$

   where the ψ-controlled proposal adaptively nudges particles toward regions of high posterior probability.

4. **Weighting and resampling:** Importance weights are assigned to each particle according to the standard SMC formula:

$$w_t^{(n)} \propto \frac{p(y_t \mid X_t^{(n)})\, p(X_t^{(n)} \mid X_{t-1}^{(n)})}{q_\psi(X_t^{(n)} \mid X_{t-1}^{(n)}, y_t)},$$

   where q_ψ is the ψ-controlled proposal. Particles are then resampled conditionally to reduce variance, with resampling frequency implicitly determined by the learned policies.

The key differences from the *fixed ORC* method are:

- **Adaptive policy:** Instead of using a fixed shift γ, ORCSMC learns a time-varying control policy ψ_t that adapts to both the observed data and the latent dynamics within a rolling window.
- **Rolling backward-forward update:** ORCSMC performs a backward recursion to refine policies before propagating particles forward, improving approximation of the posterior within the window.
- **Conditional resampling:** Resampling is guided by the learned policies, rather than being performed at fixed intervals, which reduces weight degeneracy more efficiently.

The output of ORCSMC is a set of weighted particles approximating the posterior distribution $p(x_{1:t} \mid y_{1:t})$, enabling online latent state estimation with lower variance and better representation of high-probability regions compared to fixed ORC methods.

ORC-Fixed:

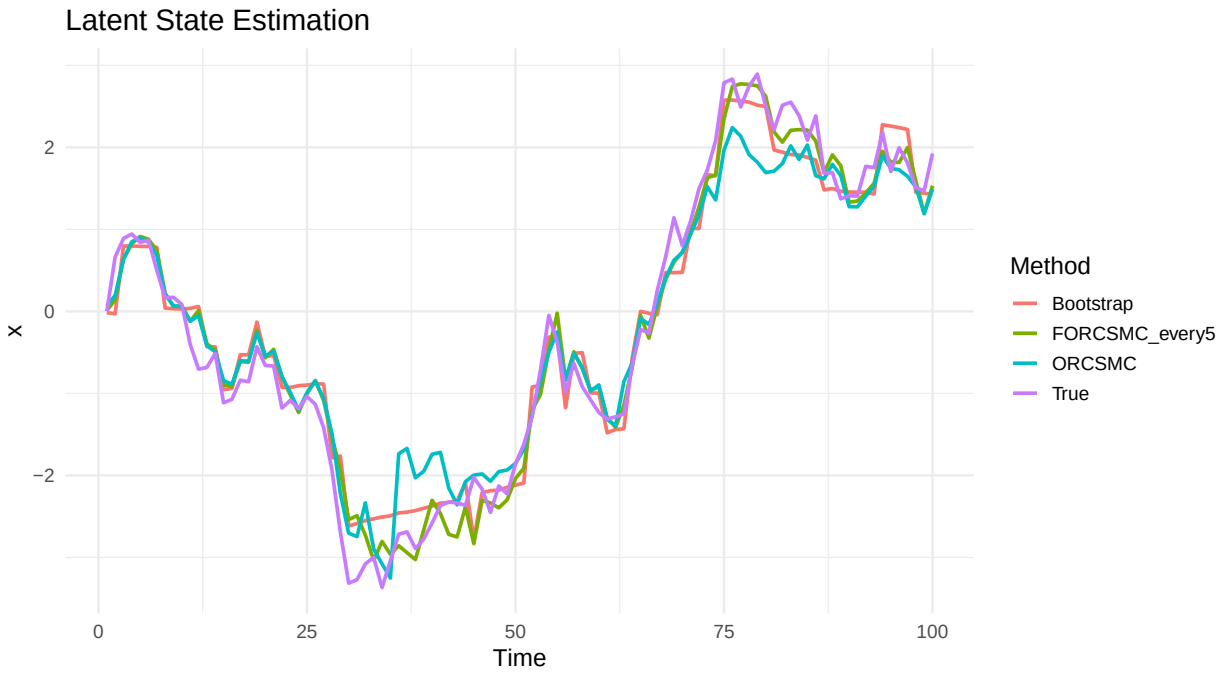
I include this here just for sake of reminder, but as we saw above the ORC-fixed method is a variation of the ORC-SMC in which the resampling schedule is predetermined, rather than adaptive. While the propagation of particles uses the same optimal proposal as ORC-SMC, resampling is performed at fixed intervals. This can simplify implementation while still retaining some benefits of the ORC approach:

$$w_t^{(i)} \propto \frac{p(y_t \mid x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)})}{q^*(x_t^{(i)} \mid x_{t-1}^{(i)}, y_t)}. \tag{8}$$

Note that in this implemented version though, the fixed ORCSMC resamples every 5 time steps, as opposed to the regular ORCSMC implementation, which resamples whenever ESS $< \frac{N}{2}$.

Summary

All three filters aim to approximate the posterior distribution $p(x_t \mid y_{1:t})$ using a set of weighted particles. The bootstrap filter is simple but suffers from weight degeneracy in highly informative models, while ORC-SMC and ORC-fixed exploit optimal proposals to improve efficiency. By simulating the state space model and applying these filters, we can assess their performance in tracking the latent AR(1) process through noisy binomial observations.



Results and Comparison

By running both filters on the same simulated observations, we obtain two posterior mean estimates of the latent states. This comparison highlights the potential benefit of ORCSMC: by incorporating a guided proposal distribution, it can reduce variance and improve tracking of the latent process, particularly when observations are sparse or highly nonlinear. The figure above illustrates the true latent trajectory alongside the BPF and ORCSMC estimates over time.

Table 1: NRMSE Comparison

Method	NRMSE
ORCSMC	0.2245
FORCSMC_every5	0.1607
Bootstrap	0.1916

Effective Sample Size (ESS)

Remember, that in particle filtering, the *Effective Sample Size* (ESS) is a widely used diagnostic to quantify the degeneracy of particle weights at each time step. Specifically, ESS measures the number of independent particles that effectively contribute to the approximation of the posterior distribution. Mathematically, at time t, ESS is defined as

$$\text{ESS}_t = \frac{1}{\sum_{i=1}^{N} w_{i,t}^2},$$

where $w_{i,t}$ are the normalized weights of the particles at that time, and N is the total number of particles.

A low ESS indicates that most of the weight is concentrated in only a few particles, which can lead to poor approximation and necessitates resampling. Conversely, a high ESS suggests a more uniform contribution from particles, indicating a healthier filter.

In this work, we calculate the ESS over time for the standard bootstrap particle filter (BPF) the Optimal Resampling Controlled Sequential Monte Carlo (ORCSMC) method, and the fixed ORC method. The resulting ESS trajectories are plotted below, allowing for a direct comparison of particle diversity and weight degeneracy between the two filtering approaches.

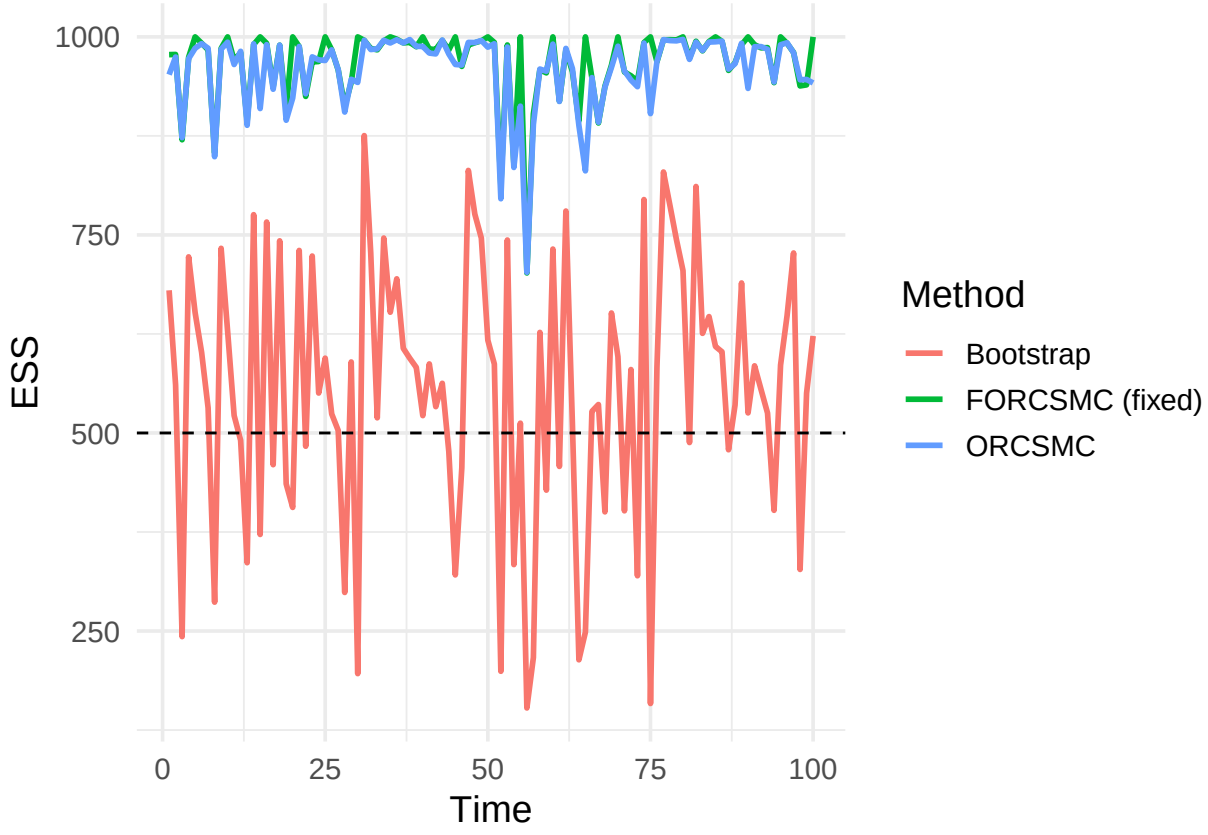


Table 2: ESS values for each method over all time steps

Time	Bootstrap	FORCSMC..fixed.	ORCSMC
1	680.08	977.57	952.34
2	560.58	977.76	974.56
3	242.92	870.09	872.78
4	722.32	974.21	972.48
5	651.89	1000.00	985.63
6	601.64	992.00	991.44
7	531.69	983.50	985.62
8	286.40	853.51	848.71
9	733.10	985.46	984.43
10	628.31	1000.00	993.28
11	522.00	970.38	965.09
12	490.74	980.01	981.78
13	336.19	893.78	888.38
14	775.59	990.30	991.16
15	371.89	1000.00	909.50
16	766.27	991.85	990.16
17	459.93	937.00	933.89
18	742.56	989.52	989.55
19	435.77	907.35	894.75
20	406.01	1000.00	922.78
21	730.45	987.71	987.87
22	483.21	924.84	928.16
23	723.36	968.30	974.79
24	550.38	968.88	970.94
25	594.43	1000.00	970.18
26	524.01	983.67	983.63
27	503.12	958.17	958.59
28	298.66	912.47	905.05
29	589.64	942.50	946.48
30	196.20	1000.00	942.50
31	875.37	995.65	995.79
32	727.00	985.38	984.02
33	519.10	983.64	984.83
34	746.09	994.95	995.68
35	652.39	1000.00	992.28
36	694.57	997.05	996.37
37	606.82	992.39	992.37
38	594.29	993.01	996.62
39	582.76	987.49	987.97
40	521.88	1000.00	987.98
41	587.17	984.53	979.53
42	533.20	984.15	978.47
43	562.71	995.51	996.02
44	476.69	983.50	978.46
45	320.80	1000.00	965.16
46	455.88	962.70	964.48
47	831.43	989.61	993.04
48	776.19	992.59	992.86
49	746.01	994.91	995.46
50	617.09	1000.00	986.98

Time	Bootstrap	FORCSMC..fixed.	ORCSMC
51	587.01	992.92	991.26
52	199.17	812.93	795.62
53	743.60	989.78	987.17
54	333.84	839.93	835.11
55	512.37	1000.00	912.53
56	152.84	701.95	702.84
57	216.51	902.78	889.55
58	626.97	958.50	959.39
59	427.81	954.51	956.64
60	732.17	1000.00	990.87
61	457.78	918.80	918.42
62	780.16	983.55	985.46
63	512.03	956.18	960.07
64	213.64	893.98	888.14
65	248.69	1000.00	830.86
66	527.23	948.11	948.60
67	535.92	891.25	892.25
68	400.37	937.60	936.48
69	651.47	964.14	961.46
70	596.25	1000.00	988.22
71	401.71	955.64	956.87
72	580.07	950.89	945.48
73	319.72	944.12	937.09
74	794.62	992.85	992.19
75	158.50	1000.00	903.13
76	583.51	968.62	969.22
77	829.42	996.30	996.53
78	787.06	996.41	995.31
79	743.83	996.16	994.96
80	705.00	1000.00	996.75
81	487.90	973.12	971.55
82	811.12	994.49	993.37
83	625.70	982.35	982.76
84	646.93	994.13	993.43
85	609.12	1000.00	993.77
86	602.43	994.29	994.57
87	478.77	957.64	959.33
88	535.20	967.30	966.04
89	689.37	989.95	991.91
90	525.30	1000.00	934.94
91	584.79	990.02	987.58
92	554.76	986.11	987.71
93	525.05	986.40	984.25
94	402.34	942.18	943.00
95	585.69	1000.00	989.99
96	647.93	991.87	992.57
97	727.23	980.31	980.04
98	327.70	938.07	945.80
99	550.31	939.38	946.24
100	622.45	1000.00	941.85

What do we notice?

In this work, we implement both a standard bootstrap particle filter (BPF) and a simplified ORCSMC algorithm to estimate the latent state of a univariate AR(1) process with binomial observations. Interestingly, in our 1D setting, we observe that both methods produce very similar results in terms of the estimated latent trajectory and normalized RMSE over time.

This similarity can be explained as follows. The latent process is one-dimensional and relatively well-behaved, with moderately low process noise ($\sigma = \sqrt{0.11}$) and reasonably informative observations ($M = 50$). In such a setting, the posterior distribution of the latent state at each time step is fairly concentrated and close to Gaussian, meaning that even the simple proposal used in the bootstrap particle filter is sufficient to accurately capture the posterior.

The ORCSMC algorithm incorporates a guided proposal of the form:

$$x_t^{(i)} \sim \mathcal{N}\Big(\alpha x_{t-1}^{(i)} + 0.5\big(y_{t-1}/M - \text{logistic}(x_{t-1}^{(i)})\big), \sigma^2\Big),$$

which, in this 1D setting, is not dramatically different from the standard BPF proposal $x_t^{(i)} \sim \mathcal{N}(\alpha x_{t-1}^{(i)}, \sigma^2)$. Consequently, both filters place particles in similar regions of the state space, resulting in comparable latent state estimates.

It is important to note that differences between BPF and ORCSMC typically become pronounced in more challenging scenarios, such as:

- Higher-dimensional latent states, where standard BPF suffers from weight degeneracy.
- Highly nonlinear or multimodal posterior distributions.
- Sparse or highly noisy observations, where a guided proposal improves particle placement.

In the present 1D setting, the posterior is sufficiently simple and informative that the bootstrap particle filter already performs very well, and the benefits of ORCSMC are not immediately apparent. Increasing process noise, reducing the number of observations, or moving to a higher-dimensional system would likely demonstrate the advantages of the guided proposal more clearly.

Below, we will see the establishment of this state space model in higher dimensional settings, and will be implementing numerous filters, such as the BPF, the ORCSMC/FORCSMC algorithm, and some numerical integration grid-based approaches.

Defining the Higher Dimensional Model

Building on the univariate setting introduced previously, we now extend the model to a higher-dimensional context in order to investigate interactions and dynamics across multiple latent neuronal populations.

Let $X_t := (X_{t,1}, \ldots, X_{t,d}) \in \mathbb{R}^d$ denote the d-dimensional latent state at time t, where each dimension $X_{t,j}$ represents a distinct type of neuron or independent neuronal population. Correspondingly, let $Y_t := (Y_{t,1}, \ldots, Y_{t,d})$ denote the observed neuronal activations.

Observation model. Each component $Y_{t,j}$ is modeled as a Binomial count arising from M independent Bernoulli trials with success probability given by a logistic function of the latent state:

$$Y_{t,j} \mid X_{t,j} \sim \text{Binomial}(M, \kappa(X_{t,j})), \quad j = 1, \ldots, d, \tag{9}$$

$$\kappa(x) := \frac{1}{1 + \exp(-x)}. \tag{10}$$

Here, M retains its original interpretation as the number of independent trials observed at each time step for a given latent dimension. Importantly, M is *internal to each latent state*: even in the multivariate setting, each latent component generates its own Binomial observation through M independent events. Thus, the counts $Y_{t,1}$ and $Y_{t,2}$ (for $d = 2$) are not derived from the same set of trials but rather represent independent observations corresponding to distinct neuronal types.

Latent dynamics. The latent state evolves according to a multivariate linear Gaussian autoregressive process:

$$X_0 \sim \mathcal{N}(0_d, I_d), \tag{11}$$

$$X_t \mid X_{t-1} \sim \mathcal{N}(\alpha X_{t-1}, \sigma^2 I_d), \tag{12}$$

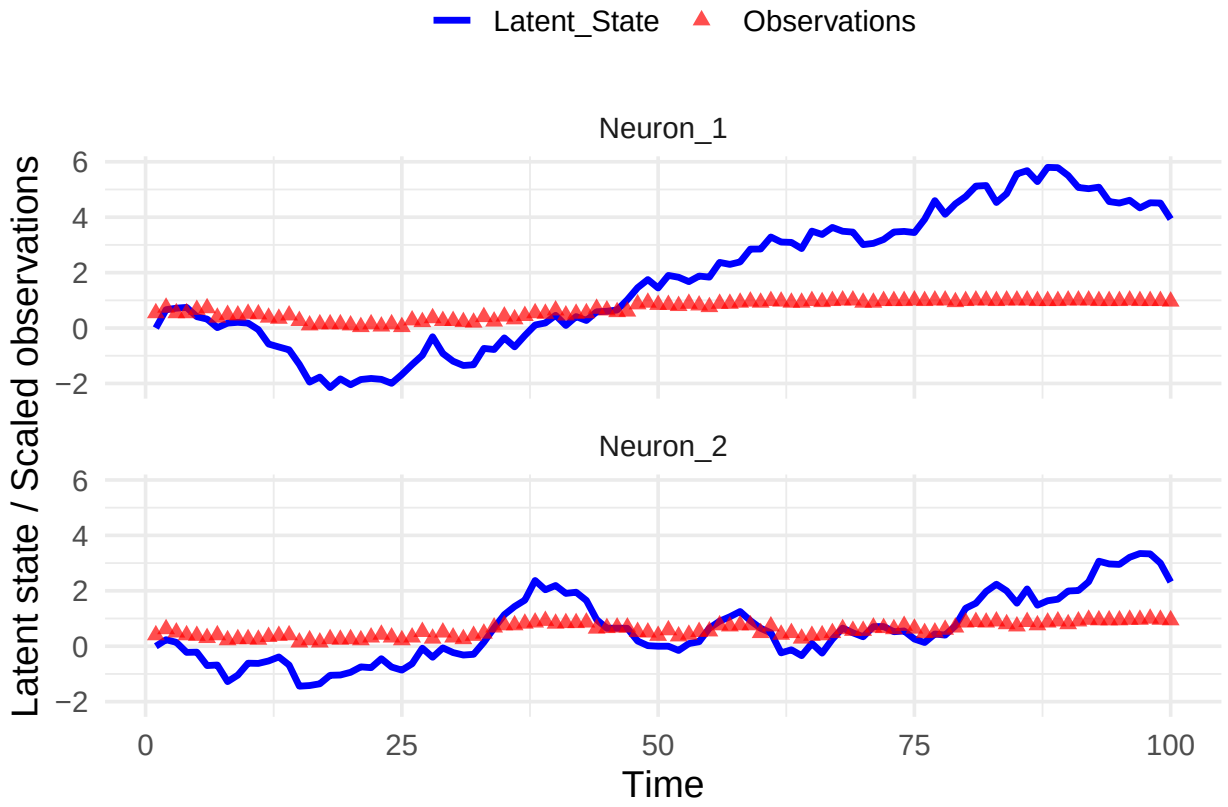
where 0_d is the d-dimensional zero vector, I_d is the $d \times d$ identity matrix, $\alpha \in (0, 1)$ is the AR(1) coefficient, and σ^2 is the process variance. This formulation implies that each latent component evolves independently over time, maintaining the simplicity of the univariate dynamics while allowing for simultaneous modeling of multiple neuronal populations.

Subtleties. It is crucial to distinguish between the role of M and the dimensionality d. While M controls the number of Bernoulli trials for a single latent component, d represents the number of distinct latent processes being modeled. For instance, in a 2-dimensional setting ($d = 2$), $X_{t,1}$ and $X_{t,2}$ can be interpreted as two different types of neurons or neuronal populations, each generating independent Binomial observations $Y_{t,1}$ and $Y_{t,2}$, respectively, via their own M trials. This distinction ensures that the model captures variability both within and across neuronal types.

Factorization of the likelihood. Due to the conditional independence across dimensions given the latent states, the joint observation likelihood factorizes as

$$p(Y_t \mid X_t) = \prod_{j=1}^{d} \text{Binomial}\big(Y_{t,j}; M, \kappa(X_{t,j})\big), \tag{13}$$

allowing for straightforward computation of the multivariate likelihood while preserving the interpretability of each latent dimension as an independent neuronal process.


Latent_State
Observations
Neuron_1
Neuron_2
Latent state / Scaled observations
6
4
2
0
−2
0
25
50
75
100
Time