

Supplementary information. The article has the following accompanying supplementary materials:

- Appendix A contains additional experiments for the moving window size hyperparameter;
- Appendix B offers definitions of the state-space models with tractable likelihoods that were used in the experiments;
- Appendix C provides implementation details for the user models used in the experiments, as well as technical information for the comparison methods;

Appendix A Moving window size analysis

The performance of our multi-objective LFI approach across different moving window lengths is summarised in Tables A1, A2, and A3. Our primary finding reveals that having merely two objectives within the moving window sufficiently maximises performance benefits, whereas the inclusion of more objectives increases computational time and leads to inconsistent performance results. This inconsistency worsens when the transition dynamics experience a high rate of change, rendering multi-objective modelling challenging due to increased dissimilarity between successive objectives. For example, LMC-BNN and LMC-qEHVI improved for cases with smoother trajectories (e.g., LG, NN) but struggled in scenarios with occasional erratic transitions (e.g., SV). As a result, we recommend adopting a moving window size of two objectives for unknown dynamics, as the potential improvements offered by larger windows are minimal and computationally expensive.

Furthermore, the influence of additional objectives varies with different acquisition methods. For LMC-BNNs, having more objectives allows the transition dynamics model to frequently sample from updated state posterior approximations. While beneficial for simple transition dynamics, this approach struggles when modelling a multitude of objectives, as seen in the SV case. The LMC-qEHVI method, on the other hand, focuses solely on optimising objectives within the moving window without accounting for future states. As such, it only performs well when future objectives closely mirror the current ones and falls short when transition dynamics display erratic behaviour (SV,

UMAP, and Gaze cases). Lastly, increasing the moving window size negatively impacts the LMC-BLR method due to local linearity diminishing as more objectives are included. Therefore, a smaller window size remains the preferred choice across all methods.

Appendix B The state-space models with tractable likelihoods

In this section, we elaborate on the specifications of the three SSMs with tractable likelihoods, which were employed in our experiments to evaluate the efficacy of state transition models. We operate under the assumption of known ground-truth transition dynamics, defining an observation model and establishing priors for the LFI of states. Observations were drawn from datasets comprising ten points, using their mean and standard deviation as summary statistics. The objective in the BO approach was the logarithm of the Euclidean distance between observed and simulated summary statistics. Figure B2 provides a visual representation of the auto-correlation function for SSM state transition dynamics, aiding in the understanding of their temporal correlations and characteristics.

B.1 Linear Gaussian (LG)

In the LG model, state transition dynamics (Figure B1a) and an observation model are both linear

$$\theta_t = 0.95 \times \theta_{t-1} + v_t, \quad x_t = \theta_t + w_t,$$

with added Gaussian noise $v_t \sim \mathcal{N}(0, 1^2)$ and $w_t \sim \mathcal{N}(0, 10^2)$. The initial state point for the transition dynamics is $\theta_0 = 100$. The prior for the states is $\theta \sim \text{Unif}(0, 15) \in \mathbb{R}$.

B.2 Non-linear non-Gaussian (NN)

The NN model is a popular non-linear SSM Kitagawa (1996), where the transition dynamics (Figure B1b) and observation model are

$$\theta_{h,t} = \frac{\theta_{h,t-1}}{2} + 25 \frac{\theta_{h,t-1}}{\theta_{h,t-1}^2 + 1} + 8 \cos(1.2t) + v_t,$$

Table A1: Comparison of LFI methods (rows) in different SSMS (columns) for the state inference task with different moving window lengths (a number in parentheses). The performance was measured with 95% confidence interval (CI) of the RMSE between estimated and ground truth state points for 50 time-steps. The best results in each column are highlighted in bold.

Methods	LG	NN	SV	UMAP	Gaze
LMC-BNN (2)	1.77 $\pm$ 0.12	6.92 $\pm$ 0.51	16.14 $\pm$ 3.27	58.24 $\pm$ 3.62	58.7 $\pm$ 5.4
LMC-BNN (3)	1.78 $\pm$ 0.12	6.95 $\pm$ 0.53	15.86 $\pm$ 2.6	56.99 $\pm$ 2.68	59.45 $\pm$ 5.02
LMC-BNN (5)	1.76 $\pm$ 0.11	6.84 $\pm$ 0.49	17.6 $\pm$ 2.69	58.89 $\pm$ 2.9	60.26 $\pm$ 5.31
LMC-BLR (2)	1.3 $\pm$ 0.1	6.86 $\pm$ 0.54	13.15 $\pm$ 3.25	59.19 $\pm$ 3.31	60.6 $\pm$ 5.8
LMC-BLR (3)	1.76 $\pm$ 0.13	6.84 $\pm$ 0.65	12.14 $\pm$ 2.46	61.68 $\pm$ 4.22	60.49 $\pm$ 5.01
LMC-BLR (5)	1.8 $\pm$ 0.16	7.12 $\pm$ 0.54	12.83 $\pm$ 2.78	61.8 $\pm$ 5	60.35 $\pm$ 5.29
LMC-qEHVI (2)	1.5 $\pm$ 0.1	7.03 $\pm$ 0.55	24.4 $\pm$ 2.5	64.96 $\pm$ 3.72	56.9 $\pm$ 4.5
LMC-qEHVI (3)	1.47 $\pm$ 0.1	7.37 $\pm$ 0.53	26.02 $\pm$ 2.5	68.75 $\pm$ 4.13	61.68 $\pm$ 4.85
LMC-qEHVI (5)	1.41 $\pm$ 0.07	6.92 $\pm$ 0.64	26.64 $\pm$ 2.7	67.42 $\pm$ 2.92	60.76 $\pm$ 4.74

Table A2: Comparison of transition dynamics models (rows) in different SSMS (columns) with different moving window lengths (a number in parentheses). The performance was measured with 95% CI of the RMSE between sampled and ground truth trajectories of length 50. The best results in each column are highlighted in bold.

Methods	LG	NN	SV	UMAP	Gaze
LMC-BNN (2)	210 $\pm$ 4	148 $\pm$ 2	117 $\pm$ 21	1394 $\pm$ 27	1365 $\pm$ 3
LMC-BNN (3)	209 $\pm$ 3	147 $\pm$ 2	123 $\pm$ 25	1408 $\pm$ 28	1367 $\pm$ 2
LMC-BNN (5)	208 $\pm$ 3	147 $\pm$ 1	134 $\pm$ 27	1416 $\pm$ 36	1366 $\pm$ 2
LMC-BLR (2)	64 $\pm$ 7	154 $\pm$ 4	100 $\pm$ 37	1409 $\pm$ 49	1372 $\pm$ 3
LMC-BLR (3)	197 $\pm$ 6	153 $\pm$ 4	79 $\pm$ 22	1387 $\pm$ 56	1367.2
LMC-BLR (5)	197 $\pm$ 5	150 $\pm$ 2	84 $\pm$ 22	1404 $\pm$ 67	1370 $\pm$ 5

$$x_t = \frac{\theta_{h,t}^2}{20} + w_t,$$

with added Gaussian noise $v_t \sim \mathcal{N}(0, 10)$ and $w_t \sim \mathcal{N}(0, 10)$. The initial state point for the noise standard deviation $\theta_{h,0}$ is 0 with the prior $\theta_h \sim \text{Unif}(-30, 30) \in \mathbb{R}$.

B.3 Stochastic volatility (SV)

SV models are widely used for predicting volatility of asset prices (Shephard, 1996, Taylor, 1994). Here, we use the model by Barndorff-Nielsen and Shephard (2002) that specifies transition dynamics (Figure B1c) of volatility θ_v as

$$\theta_{v,t+1} = (z_t - z_{t+1} + \sum_{j=1}^k e_j),$$

$$z_{t+1} = \exp(-\lambda)z_t + \sum_{j=1}^k \exp\{-\lambda(t+1-c_j)\}e_j,$$

with $c_{1:k} \sim \text{Unif}(t, t+1)$, $e_{1:k} \sim \text{Expon}(0.5/0.25^2)$ and $\lambda = 0.01$. The random increases of volatility are regulated by the Poisson distributed variable $k \sim \text{Poisson}(0.5\lambda^2/0.25^2)$. The observation model for the log-return of an asset $x_t \in \mathbb{R}$ follows

$$x_t = \theta_\mu + \theta_\beta \theta_{v,t} + (\theta_{v,t}^{0.5} + 10^{-5})\varepsilon_t,$$

where $\varepsilon_t \sim \mathcal{N}(0, 1)$; $\theta_\mu = 0$ and $\theta_\beta = 0$ remain the same, while the volatility θ_v follows the transition dynamics, starting with the initial value for the volatility $\theta_{v,0}$ of 1. We set the following priors for states: $\theta_\mu \sim \text{Unif}(-2, 2) \in \mathbb{R}$, $\theta_\beta \sim \text{Unif}(-5, 5) \in \mathbb{R}$, $\theta_v \sim \text{Unif}(0, 3) \in \mathbb{R}$.

Table A3: Time comparison of LFI methods (rows) in different SSMs (columns) with different moving window lengths (a number in parentheses) for training 50 time-steps. The running time is shown in minutes along with 95% CI. The best results in each column are highlighted in bold.

Methods	LG	NN	SV	UMAP	Gaze
LMC-BNN (2)	87.6 $\pm$ 2.6	79.2 $\pm$ 1	81 $\pm$ 2.9	171.2 $\pm$ 5	408 $\pm$ 8
LMC-BNN (3)	105.8 $\pm$ 5.8	96.8 $\pm$ 2	84.9 $\pm$ 5.7	201.4 $\pm$ 6.2	457.9 $\pm$ 6.4
LMC-BNN (5)	93.8 $\pm$ 2.2	97.3 $\pm$ 2.1	102.4 $\pm$ 3.6	198.2 $\pm$ 6.1	434.4 $\pm$ 8.1
LMC-BLR (2)	82.1 $\pm$ 5.5	48.2 $\pm$ 0.8	93.5 $\pm$ 4	149.4 $\pm$ 4.5	442.5 $\pm$ 8.8
LMC-BLR (3)	82.9 $\pm$ 1.6	58.5 $\pm$ 4.1	112.2 $\pm$ 1.4	189.5 $\pm$ 7.5	496.7 $\pm$ 13.2
LMC-BLR (5)	64 $\pm$ 1.8	62.8 $\pm$ 2.7	111.6 $\pm$ 2.2	190.3 $\pm$ 6.4	513.5 $\pm$ 11.5
LMC-qEHVI (2)	25.5 $\pm$ 1.5	23.9 $\pm$ 0.5	24.7 $\pm$ 0.7	116 $\pm$ 4.6	347.2 $\pm$ 7.3
LMC-qEHVI (3)	24.7 $\pm$ 1.5	27.7 $\pm$ 0.9	25.4 $\pm$ 1	159 $\pm$ 6	340.2 $\pm$ 7.1
LMC-qEHVI (5)	28.6 $\pm$ 0.8	41.3 $\pm$ 18.3	31.7 $\pm$ 0.4	141.6 $\pm$ 7.6	394.5 $\pm$ 10.1

Appendix C Technical details

In the following section, we delve into the intricacies and methodologies underpinning our experimental approaches. This section encompasses distinct computational models and algorithms employed in our research. We begin with the mechanism for eye movement control for gaze-based selection, simulating human-like decision-making using a reinforcement learning environment. Following this, the UMAP parameterization model is detailed, highlighting the methods and techniques used to generate and assess embeddings. Comparative methods and their implementations provide a comprehensive look into different algorithms and their efficiency in the context of likelihood-free inference. BOLFI’s details underscore its use of Gaussian Process (GP) surrogate and acquisition functions, while our novel method outlines its multi-objective components and surrogate techniques. Sequential neural estimators are elaborated, emphasizing their role in approximating postulations, with the section concluding with the exploration of recurrent state-space models employing GP transitions. Access the full implementation via the online repository at <https://github.com/AaltoPML/LFI-in-SSMs-with-UD>.

C.1 Eye movement control for gaze-based selection

In the gaze selection experiment, the observation model represents a simulated environment wherein a human subject is emulated via a reinforcement learning agent. This environment is built atop the

Open AI gym framework (Brockman et al., 2016), designed to emulate human-like decision-making in identifying a target on a 2D display. The target location, agent’s actions, observations, and beliefs are embodied by two coordinates c_1, c_2 on the display, with $c_1, c_2 \in [-1, 1]$.

Environment characteristics:

- **States:** Denoting the target’s position, states inhabit the 2D space, whereby potential positions span from the top-left coordinate $[-1, -1]$ to the bottom-right coordinate $[1, 1]$.
- **Actions:** Representing the agent’s fixation position, actions are bound by the identical spatial constraints as the states.
- **Observation:** Producing an approximation of the target’s location based on a singular observation.
- **Belief:** Offering an aggregated estimate of the target’s position by integrating all observations collected up to the current point.

For each environment episode e , the agent receives target observations distorted by noise $\mathbf{o}_e = \mathcal{N}(\mathbf{q}, \theta_s \times E^{(e)})$ and shifts the gaze to a new location $\mathbf{a}_e = \mathcal{N}(PPO(\mathbf{o}_e, \theta_{om} \cdot A^{(e)}))$. The agent’s beliefs $\mathbf{b}_e$ about the target location $\mathbf{q}$ are updated according to

$$\mathbf{b}_{e+1} = \frac{\sigma_{(o,e+1)}^2 \mathbf{o}_{e+1} + \sigma_{(b,e)}^2 \mathbf{b}_e}{\sigma_{(o,e+1)}^2 + \sigma_{(b,e)}^2},$$

$$\sigma_{(b,e+1)} = \sqrt{\frac{\sigma_{(o,e+1)}^2 \sigma_{(b,e)}^2}{\sigma_{(o,e+1)}^2 + \sigma_{(b,e)}^2}}.$$

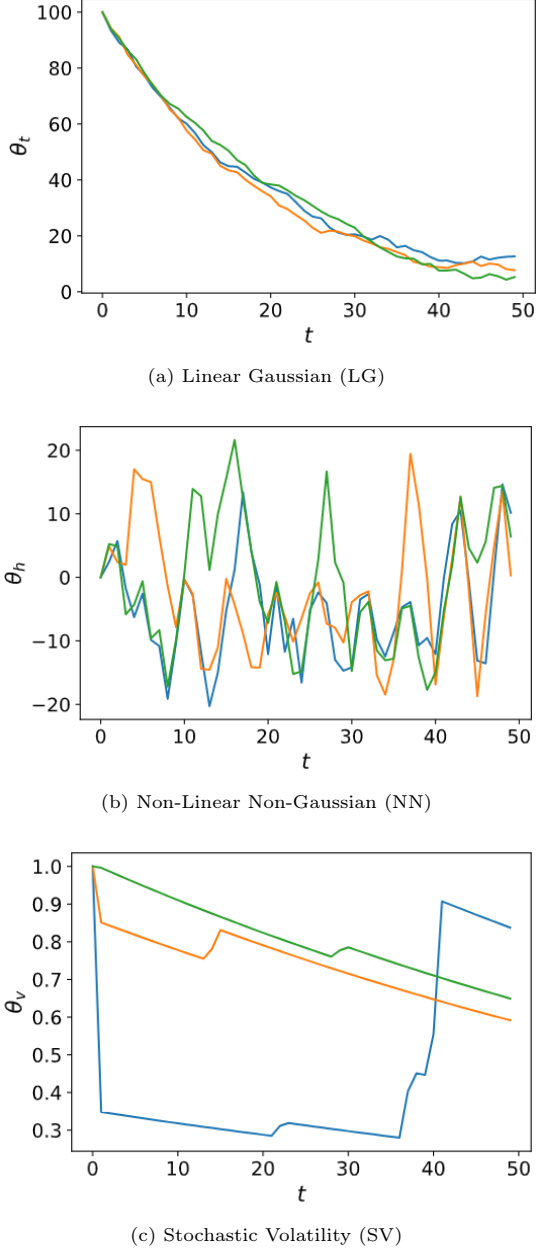


Figure B1: Transition trajectories (different colours) of states (y-axis) sampled from three SSM transition dynamics across 50 time-steps (x-axis) with different random seeds. The complexity of the dynamics gradually increases in SSMs, starting with the smooth LG (a) dynamics, where the difference between consecutive states is very small, followed by NN (b) with more erratic behaviour, and finishing with the SV model (c), whose dynamics have occasional drastic changes in state values.

Here $\sigma_{(o,e)}$ and $\sigma_{(b,e)}$ signify observation and belief uncertainties, respectively; $A^{(e)}$ represents the amplitude; and $E^{(e)}$ indicates the saccade’s eccentricity at episode e .

The user model was trained over 10,000 episodes utilising the Proximal Policy Optimisation (PPO) algorithm (Schulman et al., 2017), as facilitated by the Stable Baselines3 library (Raffin et al., 2019). By default, the training process spans up to $2 * 10^6$ epochs, deploying a multilayer perceptron policy and setting a clipping parameter to 0.15. The environment was constructed by Chen et al. (2021) utilising Python with the Open AI gym framework (Brockman et al., 2016).

C.2 UMAP parameterization

For the UMAP parameterization model, the ground truth states are not readily available as the human subject cannot define ideal embeddings. This typically necessitates millions of simulations at each time-step, but by utilising the weighted form of the preference score, we can apply the same simulations across all time-steps. We generate 1,500,000 embeddings with state points drawn from the prior and compute their corresponding $\mathcal{U}$ and $\mathcal{P}$ values. For each time-step t , we calculate the preference scores δ , retaining only a sparse subset, which is 0.06% of those states that showed the lowest δ values. Subsequently, a Gaussian kernel density estimator (KDE) is applied to these state points, with the ground truth estimations derived from maximising the corresponding PDFs. The kernel’s bandwidth b is determined following Scott’s rule (Scott, 2015) $b = n^{-\frac{1}{d+4}}$, where n is the number of data points and d represents dimensions.

Weighted preference scores:

The preference score was computed as a weighted sum between the relative validity $\mathcal{U}$ and the classification accuracy $\mathcal{P}$ on the validation set:

- **The relative validity $\mathcal{U}$** is an approximation of the density-based cluster validity (DBCV) score (Moulavi et al., 2014), which is often used as a quality measure of clustering. Intuitively, it shows how separable all the clusters are. We use the HDBSCAN* package (McInnes et al., 2017) and the HDBSCAN Boruvka KDTree (Campello et al., 2013) algorithm to cluster the points of the embeddings. We set the smallest size grouping to

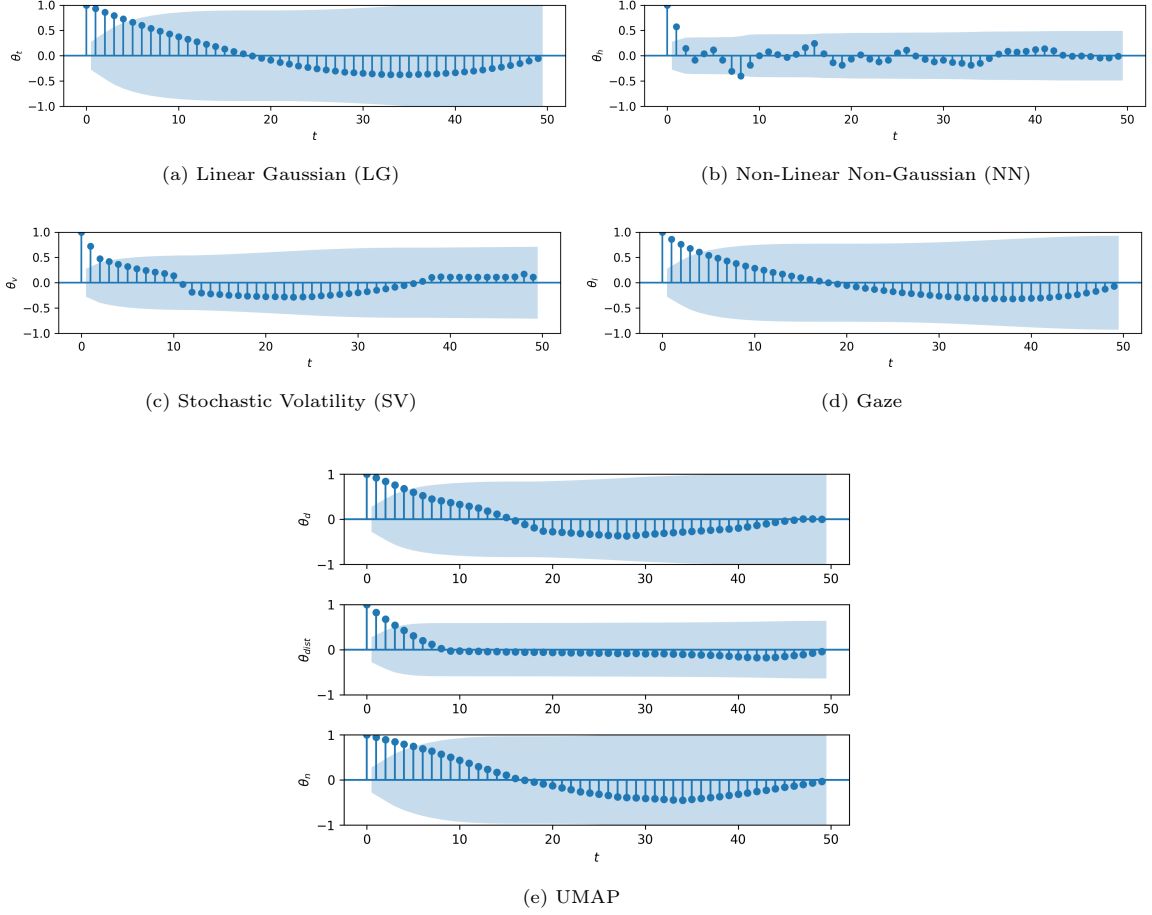


Figure B2: The figure presents the auto-correlation function (ACF) of ground-truth state transition dynamics (y-axis) across 50 time-steps (x-axis). Only state dimensions that vary over time for each SSM are shown; for a comprehensive list of latent states, refer to the subsections describing each specific SSM. The ACF plots illustrate how a time series correlates with its past values, where spikes at various lags represent the strength and direction of the correlation. Sharp declines or abrupt cut-offs in the ACF suggest non-random data, while gradual declines may point to long-term trends or cycles. A value close to 1 denotes a strong positive correlation, whereas a value near -1 signals a strong negative correlation. Conversely, values approaching 0 indicate negligible correlation between lags.

60, the density parameter of clusters to 10, and left the rest of the parameters at their default values.

- **The classification accuracy $\mathcal{P}$** was calculated for the C-Support Vector Classifier (SVC) (Boser et al., 1992, Cortes and Vapnik, 1995) with the Scikit-learn package (Pedregosa et al., 2011) and default parameters.

For experimental validation, embeddings for the UMAP parameterization model are computed

utilizing a dataset of handwritten digits (Alpaydin and Kaynak, 1998). This dataset is partitioned randomly into training and validation sets comprising 1198 and 599 images (each of size 8×8 pixels), respectively. These images, distributed across ten classes, serve as the empirical basis for the computational experiments. The UMAP algorithm, integral to the experimental setup, is implemented using the UMAP-learn package (McInnes et al., 2018b). This package offers a comprehensive environment

for dynamically and efficiently performing and evaluating simulations, embodying the cutting-edge in algorithmic research and development for UMAP parameterization.

C.3 Comparison methods

All methods in the following subsections were integrated in the engine for likelihood-free inference (ELFI) (Lintusaari et al., 2018) with the code available for application and further development through the link: <https://github.com/AaltoPML/LFI-in-SSMs-with-UD>.

C.3.1 BOLFI

For BOLFI, we used the GP surrogate in BO with the LCBSC acquisition function. The posterior samples were extracted with importance-weighted resampling. Below are the technical details for each of these BOLFI components:

1. **GP surrogate.** The implementation for the GP surrogate was provided by the GPy package (GPy, since 2012). It was initiated with a zero mean function and with the following priors for the RBF kernel lengthscale $\mathbf{l}$, its variance σ_k^2 , and the variance of the bias term σ_b^2 :

$$\begin{aligned} \mathbf{l} &\sim \text{Gamma}\left(\frac{\boldsymbol{\theta}_{\max} - \boldsymbol{\theta}_{\min}}{3}, \mathbf{1}\right), \\ \sigma_k^2 &\sim \text{Gamma}\left(\frac{\max(\delta^{(i)})^2}{9}, 1\right), \\ \sigma_b^2 &\sim \text{Gamma}\left(\frac{\max(\delta^{(i)})^2}{36}, 1\right), \end{aligned}$$

where $\boldsymbol{\theta}_{\min}, \boldsymbol{\theta}_{\max}$ are the lower and upper bounds for $\boldsymbol{\theta}^{(i)}$, $(\boldsymbol{\theta}^{(i)}, \delta^{(i)})$ are initial training points, and $\mathbf{1}$ is an all-ones vector. The GP was minimised by using the SCG optimizer (Andrej, 2007) on the GP negative log-likelihood with a maximum number of iterations of 50. All inputs $\boldsymbol{\theta}^{(i)}$ for the GP were centred and normalised.

2. **LCBSC acquisition.** The BOLFI implementation uses LCBSC (Brochu et al., 2010, Srinivas et al., 2009) as an acquisition function, which chooses points that minimise the lower confidence bound (LCB)

$$\text{LCB}(x) = \mu(x) - \beta_t^{1/2} \sigma(x),$$

$$\beta_t = 2\log(t^{2d+2}\pi^2/3\delta),$$

where β_t is the confidence parameter, $\delta = 0.1$ is the inverse exploration rate, and d is an input dimension.

3. **Posterior sampling.** The BOLFI posterior was obtained by weighting the prior samples and using the corresponding likelihoods as weights:

$$p(\mathbf{x}_*|\boldsymbol{\theta}) \propto F\left(\frac{\epsilon - \mu(\boldsymbol{\theta})}{\sqrt{\nu(\boldsymbol{\theta}) + \sigma^2}}\right),$$

with $F(\cdot)$ being a Gaussian CDF with the mean 0 and the variance 1, where ϵ is the minimum of the GP surrogate mean function $\mu(\cdot)$ obtained by using the L-BFGS-B optimizer (Byrd et al., 1995) with a maximum number of iterations of 1000.

C.3.2 Muti-objective LFI with transition model

For our method, we used an LMC surrogate, with the proposals for simulations coming from a Bayesian neural network (BNN) transition model. In the experiments, we also used the qEHVI multi-objective acquisition function and the Bayesian linear regression (BLR) alternatives for the transition model component. The technical details for the main components as well as for their alternatives are described below, along with a recommended default set of parameters that performed consistently well in all experiments. Algorithm 2 follows the algorithm from the main section but provides a lower-level description of the method.

1. **LMC surrogate.**

- *Initialisation.* The LMC model was implemented in BoTorch (Balandat et al., 2020). Its latent GPs were initialised with linear mean functions $\mu(x) = kx + b$, and RBF kernels. The lengthscales of the kernels were parameterized in log scale and initialised with 0 along with the constant b of the mean function. For the RBF kernel, ARD was also enabled to assign separate length scales for each input dimension. Furthermore, GP latents were initialised with 50 inducing points uniformly sampled inside the input bounds for each latent GP.

Algorithm 2 Multi-Objective LFI with Transition Model

Input: observations $\{\mathbf{x}_1, \dots, \mathbf{x}_T\}$, observation simulator g_θ , moving window size L , number of initial simulations B_0 , simulation budget B_{sim} , number of posterior samples I for states and K for state predictions, prior over states $p(\theta_*)$;

Output: state posteriors $\mathcal{P}$, transition model $\tilde{h}_{\theta_t}$;

Initialise an empty set for state posteriors $\mathcal{P} := \emptyset$;

Initialise the transition model $\tilde{h}_{\theta_t}$ as specified in Section C.3.2 (3a);

Simulate B_0 observations in $\mathcal{S}$ with parameters sampled from the prior $p(\theta_*)$:

$$\mathcal{S} := \{(\theta_*, \mathbf{x}_\theta) : \mathbf{x}_\theta = g_\theta(\theta_*), \theta_* \sim p(\theta_*)\}_{b=1}^{B_0};$$

Initialise the temporary variable for posterior samples $\phi_{\text{old}} := \emptyset$;

Initialise the start and end indexes for the moving window: $t_0 := 0, t := L$;

while $t < T$ **do**

 Calculate the discrepancies for parameters in $\mathcal{S}$ and form the training set $\mathcal{T}$:

$$\mathcal{T} := \{(\theta_*, \delta_w(\theta_*))\}_{w=t_0}^t : \forall (\theta_*, \mathbf{x}_\theta) \in \mathcal{S}\}_{s=1}^{|\mathcal{S}|};$$

 Train the multi-objective model $\tilde{\delta}_\theta$ as specified in Section C.3.2 (1) with $\mathcal{T}$;

 Extract I posterior samples over states from $\tilde{\delta}_\theta$ with Equation (1) as specified in Section C.3.2 (4):

$$\phi^{t_0:t} := \{(\theta_w : \theta_w \sim p(\theta_w | \mathbf{x}_w))\}_{w=t_0}^t\}_{i=1}^I;$$

 Use extracted samples to update the set of posteriors by replacing ϕ_{old} in $\mathcal{P}$:

$$\mathcal{P} := (\mathcal{P} \setminus \phi_{\text{old}}) \cup \phi^{t_0:t}, \text{ set } \phi_{\text{old}} := \phi^{t_0+1:t};$$

 Form the training set $\mathcal{D}$ by pairing K consecutive state samples from $\mathcal{P}$:

$$\mathcal{D} := \{(\theta_j, \theta_{j+1}) : \theta_j \in_R \phi^j, \phi^j \in \mathcal{P}\}_{j=1}^t\}_{k=1}^K;$$

 Update parameters of $\tilde{h}_{\theta_t}$ with data from $\mathcal{D}$ through training as specified in Section C.3.2 (3a);

 Simulate B_{sim} new observations from parameters proposed by $\tilde{h}_{\theta_t}$ and update $\mathcal{S}$:

$$\mathcal{S} := \mathcal{S} \cup \{(\theta_*, \mathbf{x}_\theta) : \mathbf{x}_\theta = g_\theta(\theta_*), \theta_* \sim \tilde{h}_{\theta_t}(\cdot | \theta_t)\}_{b=1}^{B_{\text{sim}}};$$

 Move the moving window by adjusting its indexes: $t_0 := t_0 + 1, t := t + 1$;

end while

- *Training:* The LMC training used the Adam optimizer from the tensor computation package PyTorch (Paszke et al., 2019) with a learning rate of 0.1 to minimise the variational evidence lower bound (ELBO). The optimised parameters included LMC coefficients, inducing point locations, and hyperparameters for the mean functions and kernels. Lastly, we used the default training step size and 1000 epochs for training, with all inputs for the LMC centred and normalised.
- 2. **qEHVI acquisition.** The qEHVI acquisition function used a quasi-MC sampler (Caffisch et al., 1998) with scrambled Sobol sequences (Owen, 1998) and a sample size of 128. The reference point that was used for calculating the hypervolume for each objective was set to -5. The acquisition points were acquired sequentially using successive conditioning (Wilson et al., 2018) with a maximum number of restarts of 20, a batch size limit of 10, and a maximum number of iterations for the optimizer of 200. It was also implemented in BoTorch.
- 3. **Transition dynamics surrogates:**
 - (a) **Bayesian neural network (BNN).**
 - *Initialisation:* BNN (Blundell et al., 2015) was constructed from stacked Bayesian layers implemented in the torch zoo (BLiTZ) (Esposito, 2020). We used an architecture with two hidden layers, where each layer had 256 nodes. The weights were initialised from a random normal distribution.
 - *Training:* We employed stochastic gradient descent (Sutskever et al., 2013) with a learning rate of 0.001 to minimise the ELBO loss with the squared L2 norm, which was based on 10 samples from the model for each of the 100 batches of training data in a single epoch. The

training data was randomly selected from previously stored approximated posterior samples from all states (1000 samples per state) with replacement, resulting in a total of 1 000 000 points, where one point was a pair of consecutive state points.

- *Updates:* Whenever the moving window shifted, the BNN was retrained using the new data. Previously learned weight distributions were used as priors that were updated with this new data.

(b) **Bayesian linear regression (BLR).**

- *Model Definition:* BLR is defined as $\theta_{t+1} = \theta_t^T \beta + \epsilon$, where $\epsilon \sim N(0, \sigma^2 I)$. The hyperparameters $\sigma \in \mathbb{R}$ and $\beta \in \mathbb{R}^{m \times m}$ were inferred with maximum likelihood estimation (MLE) (Myung, 2003) of

$$p(\theta_{t+1} | \theta_t) \propto \frac{1}{\sigma} \exp - \frac{(\theta_{t+1} - \theta_t \beta)^T (\theta_{t+1} - \theta_t \beta)}{2\sigma^2}.$$

- *Implementation and Training:* The BLR model was implemented with the probabilistic programming package PyMC3 (Salvatier et al., 2016), which used 100 random samples from the three latest inference steps. The (θ_{t-1}, θ_t) and $(\theta_{t-2}, \theta_{t-1})$ pairs were used as inputs for training, and the normal distribution was chosen as a prior family of the BLR parameters. The model was fitted by using the No-U-Turn sampler (NUTS) (Hoffman and Gelman, 2014) with two chains of 2000 samples, 2000 tuning iterations, and a target acceptance rate of 0.85.

4. **Posterior sampling.** Similar to BOLFI, the posterior was sampled by using importance-weighted resampling. Because the main model was implemented in PyTorch, we used the Adam optimizer to learn the threshold ϵ with a learning rate of 10^{-4} and 100 optimising iterations. The optimisation started at the parameter point that produced a synthetic observation with the smallest discrepancy.

C.3.3 Sequential neural estimators

All three SNEs (SNPE, SNLE, and SNRE) and their corresponding surrogate models were implemented in the simulation-based inference (Tejero-Cantero et al., 2020) and PyTorch (Paszke et al., 2019) packages with default parameters. In all three methods, the Adam optimizer with a learning rate of 5×10^{-4} and a training batch size of 50 in 20 epochs was used for training the surrogate. The total gradient norm was clipped in order to prevent exploding gradients at a value of 5.0, and z-score passing was used for surrogate model inputs and outputs.

For **SNPE** (Greenberg et al., 2019) and **SNLE** (Papamakarios et al., 2019), the masked autoregressive flow (MAF) surrogate was used for approximating the posterior $p(\theta | \mathbf{x})$ and likelihood, $p(\mathbf{x} | \theta)$ respectively. The MAF consisted of 5 transformations with 50 hidden features in each of the two blocks. Each autoregressive transformation had tanh activation along with batch normalisation. In contrast, **SNRE** (Durkan et al., 2020) approximated the ratio $p(\theta, \mathbf{x})/p(\theta)p(\mathbf{x})$, where a residual network (He et al., 2016) was used as a classifier trained to approximate likelihood ratios. The goal of the classifier was to predict which of the $(\theta, \mathbf{x})$ pairs was sampled from $p(\theta, \mathbf{x})$ and which from $p(\theta)p(\mathbf{x})$. The residual network had 50 hidden features in two residual blocks with 10 $(\theta, \mathbf{x})$ pairs to use for classification.

C.3.4 Recurrent state-space models with GP transitions

In the experiments, we used two variants of recurrent state-space models with GP transitions: **GP-SSM** with variationally coupled dynamics and trajectories, in which the variational inference scheme for GP transition dynamics is used (Ialongo et al., 2019), and the probabilistic recurrent state-space model **PR-SSM** (Doerr et al., 2018), which uses doubly stochastic variational inference for efficient incorporation of latent state temporal correlations. Both methods were implemented in the GPt package (Ialongo et al., 2019) with default parameters. The GPs were using Matern32 kernels with linear mean functions along with 50 randomly sampled inducing points. The number of latent dimensions was set equal to the number of simulator parameters of the observation model. The ELBO loss was calculated from 10 posterior samples and

optimised with Adam using a learning rate of 0.001
in 3000 iterations.