Sequential and adaptive Bayesian computation for inference and optimization

Ömer Deniz Akyıldız

March 11, 2019

PhD thesis presentation
Advisor: Joaquín Míguez

uc3m
Universidad Carlos III de Madrid

Talk summary

A brief introduction

Nudged particle filter

Stochastic optimization as Bayesian inference

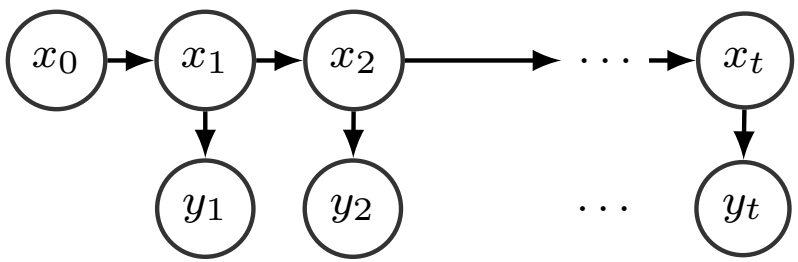
Future work

Sequential computation for inference and optimization

Inference: Given a state space model,

$$
\begin{aligned}
x_0 &\sim \tau_0(\mathrm{d}x_0), \\
x_t|x_{t-1} &\sim \tau_t(\mathrm{d}x_t|x_{t-1}), \\
y_t|x_t &\sim g_t(y_t|x_t),
\end{aligned}
$$

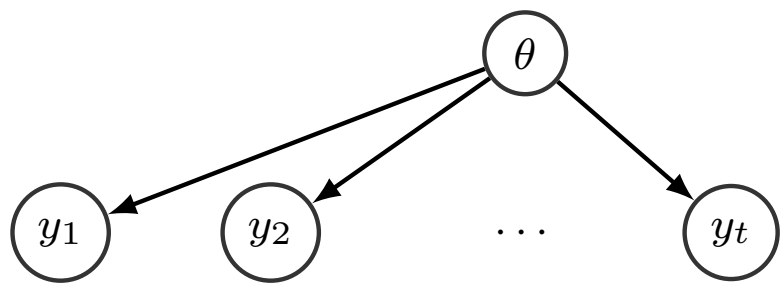
we are interested in *the stochastic filtering problem*: Estimating $\pi_t(x_t|y_{1:t})$ in high-dimensional systems and model misspecification.



Optimization: Given an optimization problem

$$
\theta^\star = \operatorname*{argmin}_{\theta \in \Theta} f(\theta),
$$

where $f(\theta) = \sum_{k=1}^{n} f_k(\theta)$, we are interested in *stochastic optimization* by framing the problem as a sequential inference problem of a matched probabilistic model.



Stochastic filtering problem

Motivation and contributions

- Of critical importance in many fields such as geophysics, object tracking, finance, ecology, aerospace.
- We focus on a class of computational methods called particle filters to solve this problem.
- Particle filters tend to fail solving this problem when
  - State-space models are high-dimensional,
  - Transition models $\tau_t(\mathrm{d}x_t|x_{t-1})$ for $t \geq 1$ are misspecified.

Stochastic filtering problem

Motivation and contributions

- Of critical importance in many fields such as geophysics, object tracking, finance, ecology, aerospace.
- We focus on a class of computational methods called particle filters to solve this problem.
- Particle filters tend to fail solving this problem when
  - State-space models are high-dimensional,
  - Transition models $\tau_t(\mathrm{d}x_t|x_{t-1})$ for $t \geq 1$ are misspecified.
- Our first main contribution is to introduce a novel particle filter which aims at tackling these two difficulties, while keeping the computational tractability of simple particle filters.

Stochastic optimization problem

Motivation and contributions

- Widely popular methods for training models in machine learning when the cost function is defined over a very big number of data points (big-data setting).
- Many algorithms have been proposed, however:
  - Most of them have parameters to tune (e.g. a step-size) and produce unstable behaviour.
  - They are mainly empirical, in the sense that, many parameters do not have intuitive meanings.

Stochastic optimization problem

Motivation and contributions

- Widely popular methods for training models in machine learning when the cost function is defined over a very big number of data points (big-data setting).
- Many algorithms have been proposed, however:
  - Most of them have parameters to tune (e.g. a step-size) and produce unstable behaviour.
  - They are mainly empirical, in the sense that, many parameters do not have intuitive meanings.
- We will show that, for a certain class of problems, we can build probabilistic models which are matched to the cost function.
  - A probabilistic interpretation may provide automatic parameter tuning with an intuitive meaning (e.g. a covariance matrix).

Stochastic optimization problem

Motivation and contributions

- Widely popular methods for training models in machine learning when the cost function is defined over a very big number of data points (big-data setting).
- Many algorithms have been proposed, however:
  - Most of them have parameters to tune (e.g. a step-size) and produce unstable behaviour.
  - They are mainly empirical, in the sense that, many parameters do not have intuitive meanings.
- We will show that, for a certain class of problems, we can build probabilistic models which are matched to the cost function.
  - A probabilistic interpretation may provide automatic parameter tuning with an intuitive meaning (e.g. a covariance matrix).
- As a byproduct, we will obtain a sampling-based probabilistic optimization method, algorithmically similar to a particle filter.

Stochastic filtering: Nudged particle filter

State-space models

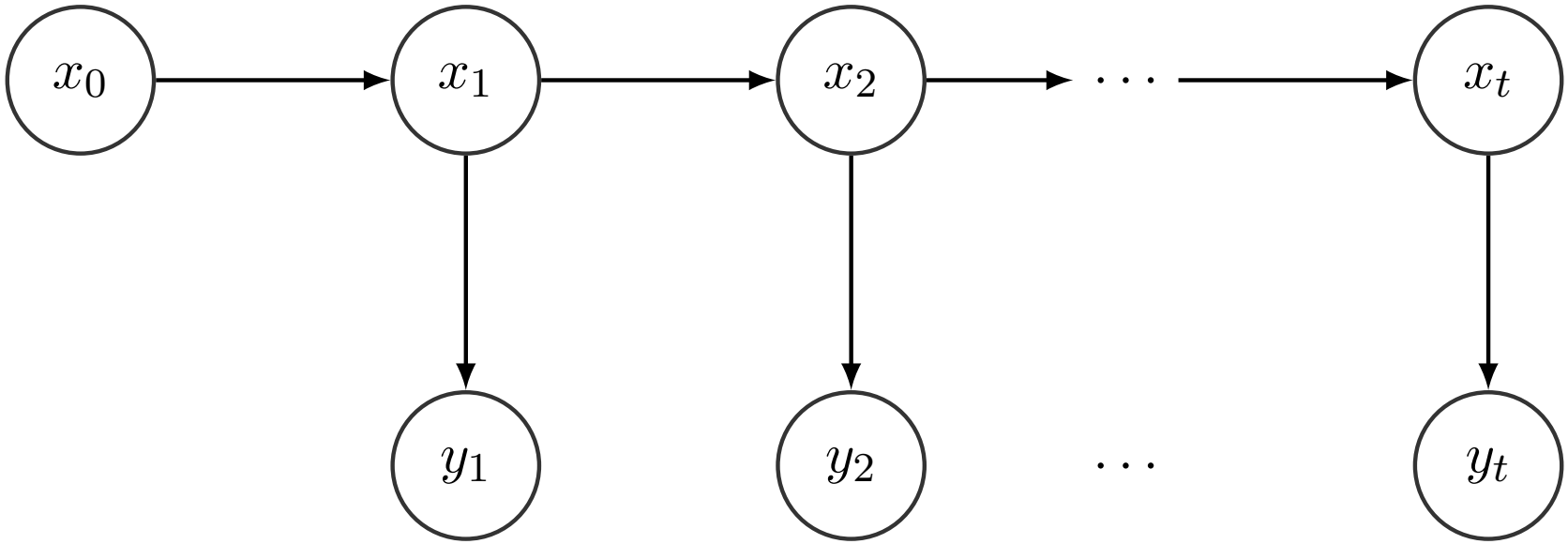


Figure: The conditional independence structure of a state-space model.

$(x_t)_{t\geq 0}$: *hidden* signal process, $(y_t)_{t\geq 1}$ the observation process.

$$
\begin{aligned}
x_0 &\sim \pi_0(\mathrm{d}x_0), && \text{(prior distribution)} \\
x_t|x_{t-1} &\sim \tau_t(\mathrm{d}x_t|x_{t-1}), && \text{(transition model)} \\
y_t|x_t &\sim g_t(y_t|x_t), && \text{(likelihood)}
\end{aligned}
$$

$x_t \in \mathsf{X}$ where X is the state-space. We use: $g_t(x_t) = g_t(y_t|x_t)$.

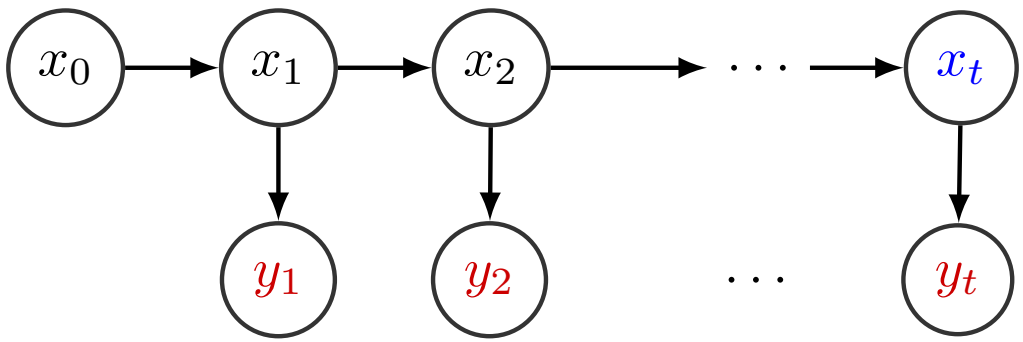
State-space models

problem definition

We are interested in estimating expectations,

$$(\varphi, \pi_t) = \int \varphi(x_t)\pi_t(x_t|y_{1:t})\mathrm{d}x_t = \int \varphi(x_t)\pi_t(\mathrm{d}x_t),$$

sequentially as new data arrives. This problem is known as *the filtering problem*.



Algorithm:

Predict

$$\xi_t(\mathrm{d}x_t) = \int \pi_{t-1}(\mathrm{d}x_{t-1})\tau_t(\mathrm{d}x_t|x_{t-1})$$

Update

$$\pi_t(\mathrm{d}x_t) = \xi_t(\mathrm{d}x_t)\frac{g_t(y_t|x_t)}{p(y_t|y_{1:t-1})}.$$

Bootstrap particle filter

A general algorithm to estimate expectations of any test function $\varphi(x_t)$ given $y_{1:t}$. Assume we are given $\{x_{t-1}^{(i)}\}_{i=1}^N$ for time $t-1$.

- Sampling: draw
$$\bar{x}_t^{(i)} \sim \tau_t(\mathrm{d}x_t | x_{t-1}^{(i)})$$
independently for every $i = 1, \ldots, N$.
- Weighting: compute
$$w_t^{(i)} = g_t(\bar{x}_t^{(i)}) / \bar{Z}_t^N$$
for every $i = 1, \ldots, N$, where $\bar{Z}_t^N = \sum_{i=1}^N g_t(\bar{x}_t^{(i)})$.
- Resampling: draw independently,
$$x_t^{(i)} \sim \tilde{\pi}_t(\mathrm{d}x) := \sum_i w_t^{(i)} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x) \quad \text{for } i = 1, ..., N.$$

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N.$$

Bootstrap particle filter

For any t the estimation of (φ, π_t) is given by,

$$(\varphi, \pi_t) = \int \varphi(x_t)\pi_t(\mathrm{d}x_t) \approx \int \varphi(x_t)\pi_t^N(\mathrm{d}x_t) = \frac{1}{N}\sum_{i=1}^{N} \varphi(x_t^{(i)}) = (\varphi, \pi_t^N).$$

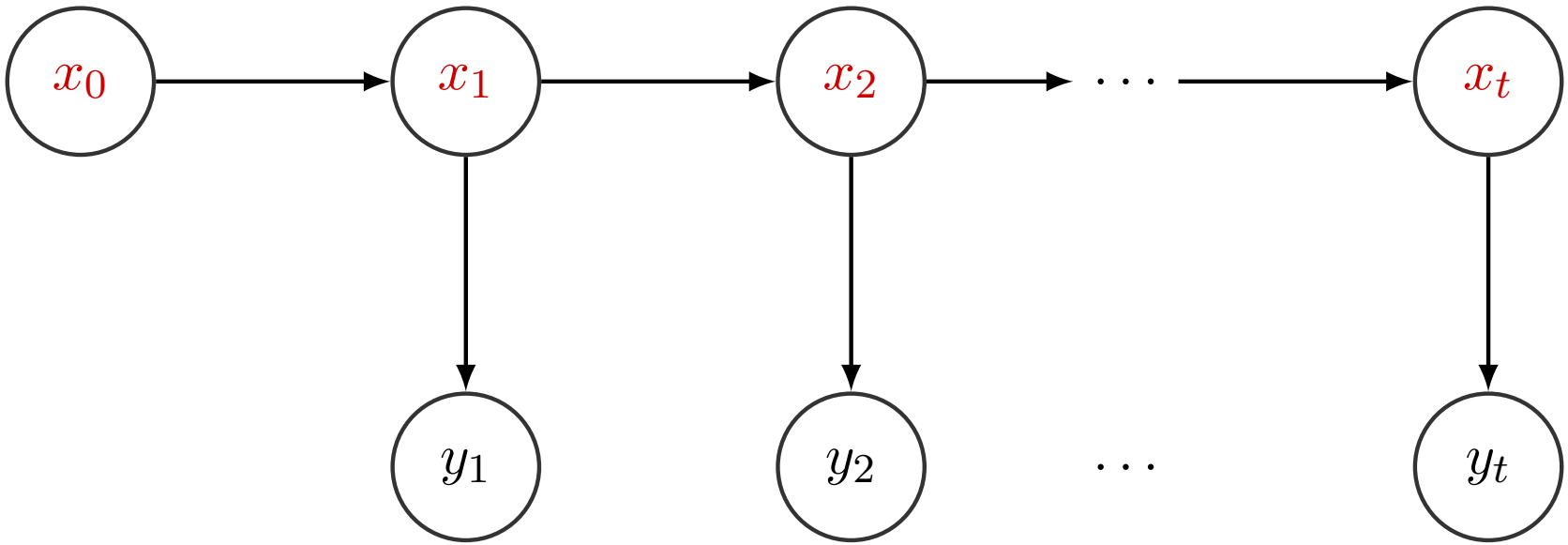
Theorem. (Del Moral and Miclo 2000) *Under suitable assumptions, one can prove that for bounded test functions* $\|\varphi\|_\infty = \sup_{x\in \mathsf{X}} |\varphi(x)| < \infty$,

$$\|(\varphi, \pi_t) - (\varphi, \pi_t^N)\|_p \leq \frac{c_t \|\varphi\|_\infty}{\sqrt{N}}$$

where $c_t < \infty$ *is a constant independent of* N.

Bootstrap particle filter

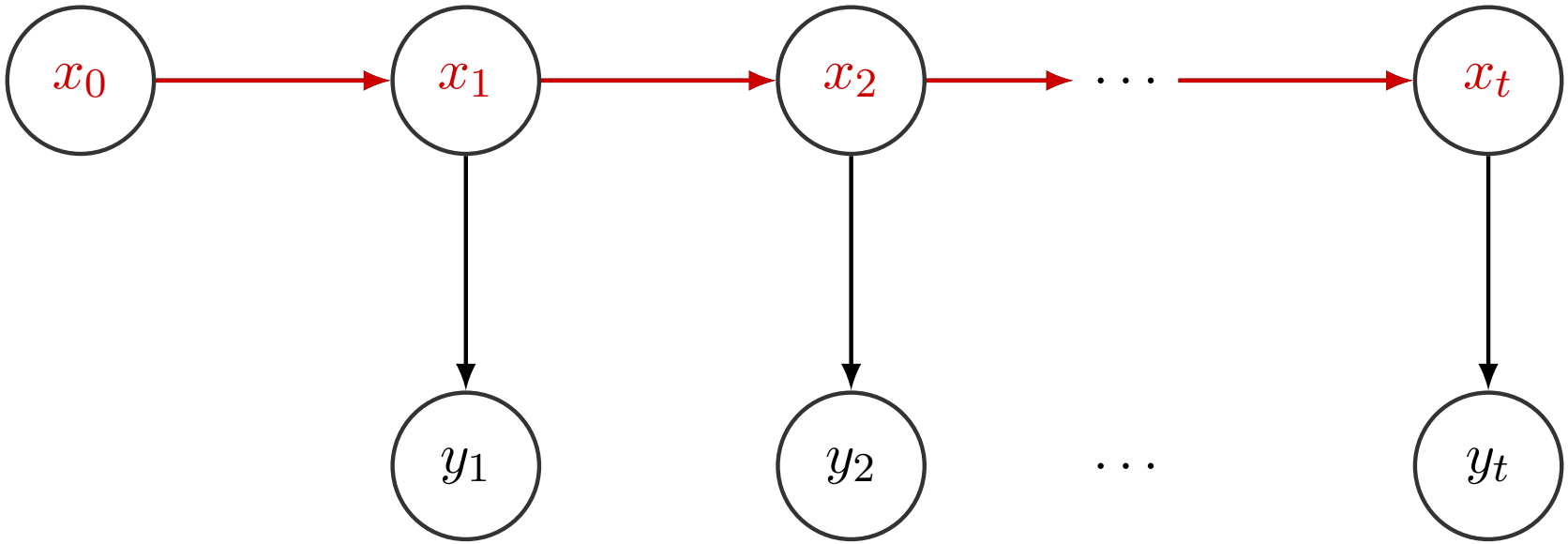
practical problems



- Bootstrap PF does not perform well when the signal process is high-dimensional.

Bootstrap particle filter

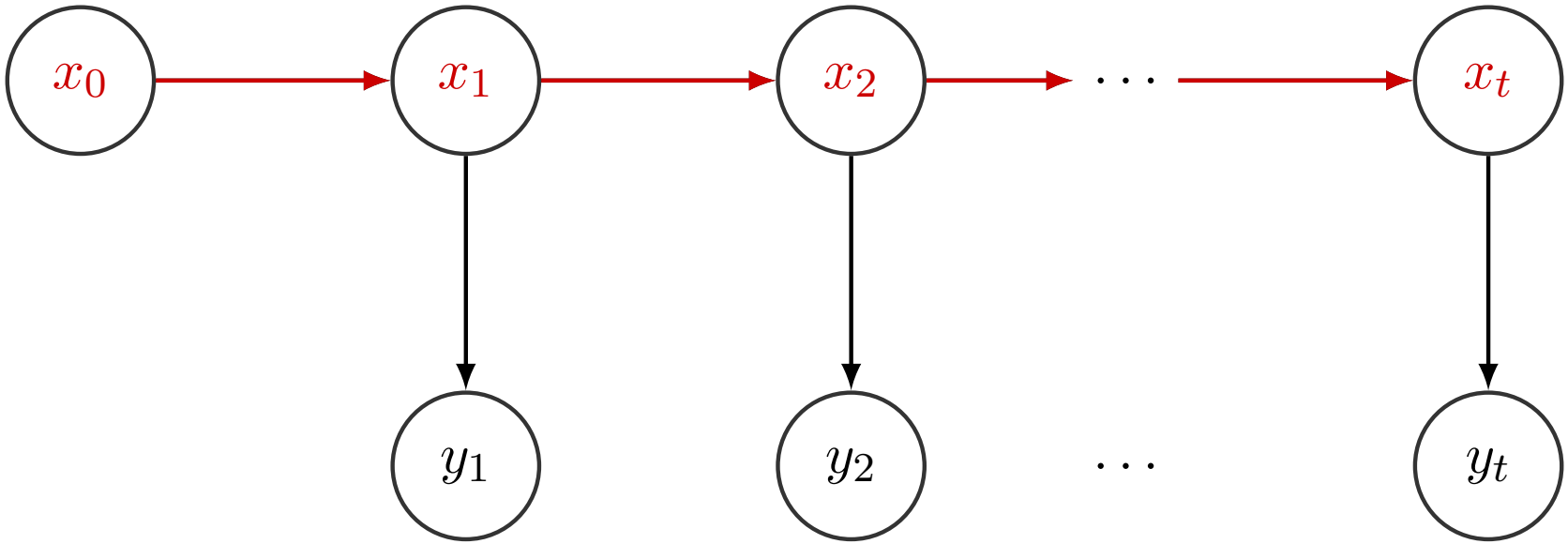
practical problems



- Bootstrap PF does not perform well when the signal process is high-dimensional.
- Bootstrap PF is not robust to model misspecification.

Bootstrap particle filter

practical problems



- Bootstrap PF does not perform well when the signal process is high-dimensional.
- Bootstrap PF is not robust to model misspecification.

One remedy is to use general and *better* proposal distributions.

General particle filter

A general algorithm to estimate expectations of any test function $\varphi(x_t)$ given $y_{1:t}$.

- Generate the initial particle system $\{x_0^{(i)}\}_{i=1}^N$ by drawing N times independently from the prior π_0.
- For $t \geq 1$,
  - Sampling: draw

$$\bar{x}_t^{(i)} \sim q_t(\mathrm{d}x_t | x_{1:t-1}^{(i)}, y_t)$$

    independently for every $i = 1, \ldots, N$.
  - Weighting: compute

$$w_t^{(i)} \propto g_t(\bar{x}_t^{(i)}) \tau_t(\bar{x}_t^{(i)} | x_{t-1}^{(i)}) / q_t(\bar{x}_t^{(i)} | x_{1:t-1}^{(i)}, y_t)$$

    for every $i = 1, \ldots, N$.
  - Resampling: draw $x_t^{(i)}$, $i = 1, ..., N$ from the discrete distribution $\sum_i w_t^{(i)} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x)$, independently for $i = 1, ..., N$.

General particle filter

A general algorithm to estimate expectations of any test function $\varphi(x_t)$ given $y_{1:t}$.

- Generate the initial particle system $\{x_0^{(i)}\}_{i=1}^N$ by drawing N times independently from the prior π_0.
- For $t \geq 1$,
  - Sampling: draw
$$\bar{x}_t^{(i)} \sim q_t(\mathrm{d}x_t | x_{1:t-1}^{(i)}, y_t)$$
independently for every $i = 1, \ldots, N$.
  - Weighting: compute
$$w_t^{(i)} \propto g_t(\bar{x}_t^{(i)}) \tau_t(\bar{x}_t^{(i)} | x_{t-1}^{(i)}) / q_t(\bar{x}_t^{(i)} | x_{1:t-1}^{(i)}, y_t)$$
for every $i = 1, \ldots, N$.
  - Resampling: draw $x_t^{(i)}$, $i = 1, ..., N$ from the discrete distribution $\sum_i w_t^{(i)} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x)$, independently for $i = 1, ..., N$.

Choosing the proposal carefully can help tackling the problems. But weight computations can get complicated and costly!

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N .$$

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N .$$

- Bootstrap particle filter is a simple and efficient algorithm.

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N .$$

- Bootstrap particle filter is a simple and efficient algorithm.
- It performs poorly in high-dimensional and misspecified settings.

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N .$$

- Bootstrap particle filter is a simple and efficient algorithm.
- It performs poorly in high-dimensional and misspecified settings.
- Using a general proposal may be helpful.
  $\rightarrow$ It is not easy to choose a good proposal.

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N .$$

- Bootstrap particle filter is a simple and efficient algorithm.
- It performs poorly in high-dimensional and misspecified settings.
- Using a general proposal may be helpful.
  $\rightarrow$ It is not easy to choose a good proposal.

To alleviate the problems of the BPF, we propose to use a simple scheme using the nudging idea.

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N .$$

- Bootstrap particle filter is a simple and efficient algorithm.
- It performs poorly in high-dimensional and misspecified settings.
- Using a general proposal may be helpful.
  $\rightarrow$ It is not easy to choose a good proposal.

To alleviate the problems of the BPF, we propose to use a simple scheme using the nudging idea.

Nudging

a practical solution to the problem

Nudging is a practical scheme aimed at solving aforementioned problems, proposed in the data assimilation literature (Van Leeuwen 2009).

Nudging

a practical solution to the problem

Nudging is a practical scheme aimed at solving aforementioned problems, proposed in the data assimilation literature (Van Leeuwen 2009).

- It is usually applied after the sampling step and consists of moving particles to "good" regions in the state-space.

Nudging

a practical solution to the problem

Nudging is a practical scheme aimed at solving aforementioned problems, proposed in the data assimilation literature (Van Leeuwen 2009).

- It is usually applied after the sampling step and consists of moving particles to "good" regions in the state-space.
- In the literature, it is usually (vaguely) defined as *pushing particles towards observations*.

Nudging

a practical solution to the problem

Nudging is a practical scheme aimed at solving aforementioned problems, proposed in the data assimilation literature (Van Leeuwen 2009).

- It is usually applied after the sampling step and consists of moving particles to "good" regions in the state-space.
- In the literature, it is usually (vaguely) defined as *pushing particles towards observations*.
  - Minimizing some distance function between y_t and each $x_t^{(i)}$.
  - Kalman-like steps

Nudging

a practical solution to the problem

Nudging is a practical scheme aimed at solving aforementioned problems, proposed in the data assimilation literature (Van Leeuwen 2009).

- It is usually applied after the sampling step and consists of moving particles to "good" regions in the state-space.
- In the literature, it is usually (vaguely) defined as *pushing particles towards observations*.
  - Minimizing some distance function between y_t and each $x_t^{(i)}$.
  - Kalman-like steps
- Moving particles around the state-space defines a complicated proposal, for which the weight computations are costly.

Nudging

a practical solution to the problem

Nudging is a practical scheme aimed at solving aforementioned problems, proposed in the data assimilation literature (Van Leeuwen 2009).

- It is usually applied after the sampling step and consists of moving particles to "good" regions in the state-space.
- In the literature, it is usually (vaguely) defined as *pushing particles towards observations*.
  - Minimizing some distance function between y_t and each $x_t^{(i)}$.
  - Kalman-like steps
- Moving particles around the state-space defines a complicated proposal, for which the weight computations are costly.
- However, it can be very efficient with a very low number of particles!

Nudging

from a classical particle filtering perspective

- Generate the initial particle system $\{x_0^{(i)}\}_{i=1}^N$ by drawing N times independently from the prior π_0.
- For $t \geq 1$,
  - Sampling: draw $\hat{x}_t^{(i)} \sim \tau_t(\mathrm{d}x_t|x_{t-1}^{(i)})$ independently for every $i = 1, \ldots, N$.
  - Nudging (a deterministic transformation): $\bar{x}_t^{(i)} = \alpha_t^{y_t}(\hat{x}_t^{(i)})$.
  - Weighting: compute

$$w_t^{(i)} \propto g_t(\bar{x}_t^{(i)})\tau_t(\bar{x}_t^{(i)}|x_{t-1}^{(i)})/q_t(\bar{x}_t^{(i)}|x_{1:t-1}^{(i)}, y_t)$$

    for every $i = 1, \ldots, N$.
  - Resampling: draw $x_t^{(i)}$, $i = 1, ..., N$ from the discrete distribution $\sum_i w_t^{(i)} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x)$, independently for $i = 1, ..., N$.

Sampling + nudging effectively induces an *implicit* Markov kernel: $q_t(\bar{x}_t^{(i)}|x_{t-1}^{(i)}, y_t)$, which is not available in closed form and whose evaluations require heavy computations!

Nudged particle filter

definition of nudging operator

We aim at developing **computationally efficient** and well-defined nudging schemes. We start with formalizing the *nudging operator*.

Definition 1

A nudging operator $\alpha_t^{y_t} : \mathsf{X} \to \mathsf{X}$ *associated with the likelihood function* $g_t(x)$ *is a map such that*

$$\text{if} \quad x' = \alpha_t^{y_t}(x) \quad \text{then} \quad g_t(x') \geq g_t(x) \tag{1}$$

for every $x, x' \in \mathsf{X}$.

Nudged particle filter

the algorithm (NuPF) (Akyıldız and Míguez 2017)

- Generate the initial particle system $\{x_0^{(i)}\}_{i=1}^N$ by drawing N times independently from the prior π_0.
- For $t \geq 1$,
  - Sampling: draw $\bar{x}_t^{(i)} \sim \tau_t(x_t|x_{t-1}^{(i)})$ independently for every $i = 1, \ldots, N$.
  - **Nudging**: choose a set of indices $\mathcal{I}_t \subset \{1, \ldots, N\}$, then compute $\tilde{x}_t^{(i)} = \alpha_t^{y_t}(\bar{x}_t^{(i)})$ for every $i \in \mathcal{I}_t$. Keep $\tilde{x}_t^{(i)} = \bar{x}_t^{(i)}$ for every $i \in [N] \backslash \mathcal{I}_t$.
  - Weighting: compute $w_t^{(i)} = g_t(\tilde{x}_t^{(i)})/\tilde{Z}_t^N$ for every $i = 1, \ldots, N$, where $\tilde{Z}_t^N = \sum_{i=1}^N g(\tilde{x}_t^{(i)})$.
  - Resample: draw $x_t^{(i)}$ from $\sum_i w_t^{(i)} \delta_{\tilde{x}_t^{(i)}}(\mathrm{d}x)$ independently for $i = 1, ..., N$.

Nudged particle filter

the algorithm (NuPF) (Akyıldız and Míguez 2017)

- Generate the initial particle system $\{x_0^{(i)}\}_{i=1}^N$ by drawing N times independently from the prior π_0.
- For $t \geq 1$,
  - Sampling: draw $\bar{x}_t^{(i)} \sim \tau_t(x_t|x_{t-1}^{(i)})$ independently for every $i = 1, \ldots, N$.
  - **Nudging**: choose a set of indices $\mathcal{I}_t \subset \{1, \ldots, N\}$, then compute $\tilde{x}_t^{(i)} = \alpha_t^{y_t}(\bar{x}_t^{(i)})$ for every $i \in \mathcal{I}_t$. Keep $\tilde{x}_t^{(i)} = \bar{x}_t^{(i)}$ for every $i \in [N] \backslash \mathcal{I}_t$.
  - Weighting: compute $w_t^{(i)} = g_t(\tilde{x}_t^{(i)})/\tilde{Z}_t^N$ for every $i = 1, \ldots, N$, where $\tilde{Z}_t^N = \sum_{i=1}^N g(\tilde{x}_t^{(i)})$.
  - Resample: draw $x_t^{(i)}$ from $\sum_i w_t^{(i)} \delta_{\tilde{x}_t^{(i)}}(\mathrm{d}x)$ independently for $i = 1, ..., N$.

We do not correct the effect of nudging!

Nudged particle filter

the algorithm (NuPF) (Akyıldız and Míguez 2017)

- Generate the initial particle system $\{x_0^{(i)}\}_{i=1}^N$ by drawing N times independently from the prior π_0.
- For $t \geq 1$,
  - Sampling: draw $\bar{x}_t^{(i)} \sim \tau_t(x_t|x_{t-1}^{(i)})$ independently for every $i = 1, \ldots, N$.
  - **Nudging**: choose a set of indices $\mathcal{I}_t \subset \{1, \ldots, N\}$, then compute $\tilde{x}_t^{(i)} = \alpha_t^{y_t}(\bar{x}_t^{(i)})$ for every $i \in \mathcal{I}_t$. Keep $\tilde{x}_t^{(i)} = \bar{x}_t^{(i)}$ for every $i \in [N] \backslash \mathcal{I}_t$.
  - Weighting: compute $w_t^{(i)} = g_t(\tilde{x}_t^{(i)})/\tilde{Z}_t^N$ for every $i = 1, \ldots, N$, where $\tilde{Z}_t^N = \sum_{i=1}^N g(\tilde{x}_t^{(i)})$.
  - Resample: draw $x_t^{(i)}$ from $\sum_i w_t^{(i)} \delta_{\tilde{x}_t^{(i)}}(\mathrm{d}x)$ independently for $i = 1, ..., N$.

We do not correct the effect of nudging!

Next: what is the operator $\alpha_t^{y_t}$ and how is it implemented?

Nudging step

- Implementation of nudging step has two crucial parts:
  - Choosing particles to be nudged (constructing $\mathcal{I}_t$).
  - Pushing particles towards high-likelihood regions.

Nudging step

- Implementation of nudging step has two crucial parts:
  - Choosing particles to be nudged (constructing $\mathcal{I}_t$).
  - Pushing particles towards high-likelihood regions.
- Choosing particles:
  - We choose a random subset of the index set!
    - Choose $M = |\mathcal{I}_t|$ particles randomly at once from $[N] = \{1, \ldots, N\}$ (batch).
    - Choose each particle to be nudged independently with probability M/N (independent).

Nudging step

- Implementation of nudging step has two crucial parts:
  - Choosing particles to be nudged (constructing $\mathcal{I}_t$).
  - Pushing particles towards high-likelihood regions.
- Choosing particles:
  - We choose a random subset of the index set!
    - Choose $M = |\mathcal{I}_t|$ particles randomly at once from $[N] = \{1, \ldots, N\}$ (batch).
    - Choose each particle to be nudged independently with probability M/N (independent).
- Pushing particles (nudging):
  - Gradient step with respect to the likelihood.
  - Random search to find a direction that increases the likelihood.
  - Model specific nudging.

Nudging step

- Implementation of nudging step has two crucial parts:
  - Choosing particles to be nudged (constructing $\mathcal{I}_t$).
  - Pushing particles towards high-likelihood regions.
- Choosing particles:
  - We choose a random subset of the index set!
    - Choose $M = |\mathcal{I}_t|$ particles randomly at once from $[N] = \{1, \ldots, N\}$ (batch).
    - Choose each particle to be nudged independently with probability M/N (independent).
- Pushing particles (nudging):
  - Gradient step with respect to the likelihood.
  - Random search to find a direction that increases the likelihood.
  - Model specific nudging.

We will only deal with the gradient step in this talk.

Nudging step

Gradient nudging step

- Choose $\mathcal{I}_t$.
- For every $i \in \mathcal{I}_t$,

$$\tilde{x}_t^{(i)} = \bar{x}_t^{(i)} + \gamma \nabla_{x_t} g_t(\bar{x}_t^{(i)})$$

where $\nabla_x g_t(x)$ denotes the vector of partial derivatives of g_t with respect to the state variables, i.e.,

$$\nabla_{x_t} g_t = \begin{bmatrix} \frac{\partial g_t}{\partial x_{1,t}} \\ \frac{\partial g_t}{\partial x_{2,t}} \\ \vdots \\ \frac{\partial g_t}{\partial x_{d_x,t}} \end{bmatrix} \quad \text{for} \quad x_t = \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{d_x,t} \end{bmatrix} \in \mathsf{X}.$$

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{nudging}} \tilde{\xi}_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N .$$

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{nudging}} \tilde{\xi}_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N .$$

- We have seen that performing nudging without correcting at the weighting step introduces bias and this is a problem.

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{nudging}} \tilde{\xi}_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N .$$

- We have seen that performing nudging without correcting at the weighting step introduces bias and this is a problem.

Next, we answer the questions:

- Does the algorithm still converge? If so, what is the rate?

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{nudging}} \tilde{\xi}_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N .$$

- We have seen that performing nudging without correcting at the weighting step introduces bias and this is a problem.

Next, we answer the questions:

- Does the algorithm still converge? If so, what is the rate?
- Why would it be robust to model misspecification?

Analysis of the nudged particle filter – convergence

A general result

Assumption 1. *The likelihood function is positive and bounded, i.e.,*

$$g_t(x_t) > 0 \quad \text{and} \quad \|g_t\|_\infty = \sup_{x_t \in \mathsf{X}} |g_t(x_t)| < \infty$$

for $t = 1, \ldots, T$.

Theorem 1. *Let* $y_{1:T}$ *be an arbitrary but fixed sequence of observations, with* $T < \infty$, *and choose any* $M \leq \sqrt{N}$ *and any map* $\alpha_t^{y_t} : \mathsf{X} \to \mathsf{X}$. *If Assumption 1 is satisfied and* $|\mathcal{I}_t| = M$, *then*

$$\|(\varphi, \pi_t^N) - (\varphi, \pi_t)\|_p \leq \frac{c_{t,p}\|\varphi\|_\infty}{\sqrt{N}} \tag{2}$$

for every $t = 1, 2, ..., T$, *any* $\varphi \in B(\mathsf{X})$, *any* $p \geq 1$ *and some constant* $c_t < \infty$ *independent of* N.

Analysis of the nudged particle filter – convergence

A general result

Assumption 1. *The likelihood function is positive and bounded, i.e.,*

$$g_t(x_t) > 0 \quad \text{and} \quad \|g_t\|_\infty = \sup_{x_t \in \mathsf{X}} |g_t(x_t)| < \infty$$

for $t = 1, \ldots, T$.

Theorem 1. *Let* $y_{1:T}$ *be an arbitrary but fixed sequence of observations, with* $T < \infty$, *and choose any* $M \leq \sqrt{N}$ *and any map* $\alpha_t^{y_t} : \mathsf{X} \to \mathsf{X}$. *If Assumption 1 is satisfied and* $|\mathcal{I}_t| = M$, *then*

$$\|(\varphi, \pi_t^N) - (\varphi, \pi_t)\|_p \leq \frac{c_{t,p}\|\varphi\|_\infty}{\sqrt{N}} \tag{2}$$

for every $t = 1, 2, ..., T$, *any* $\varphi \in B(\mathsf{X})$, *any* $p \geq 1$ *and some constant* $c_t < \infty$ *independent of* N.

Analysis of the nudged particle filter – model mismatch

Nudging as an observation dependent implicit model

We can interpret the NuPF as a BPF for an alternative model.

Analysis of the nudged particle filter – model mismatch

Nudging as an observation dependent implicit model

We can interpret the NuPF as a BPF for an alternative model.

Let us assume $y_{1:T}$ to be fixed and construct the alternative dynamical model $\mathcal{M}_1 = \{\pi_0, \tilde{\tau}_t^{y_t}, g_t^{y_t}\}$, where

$$\tilde{\tau}_t^{y_t}(\mathrm{d}x_t|x_{t-1}) := (1-\varepsilon_M)\tau_t(\mathrm{d}x_t|x_{t-1}) + \varepsilon_M \int \delta_{\alpha_t^{y_t}(\bar{x}_t)}(\mathrm{d}x_t)\tau_t(\mathrm{d}\bar{x}_t|x_{t-1}),$$

where $\epsilon_M = \frac{M}{N}$. The kernel $\tilde{\tau}_t^{y_t}$ jointly represents the Markov transition τ_t and independent nudging.

Note that this is for deterministic nudging operators. Different types of nudging can translate into different implicit models.

Analysis of the nudged particle filter – model mismatch

Nudging as an observation dependent implicit model

In other words, the NuPF = the BPF for the following observation dependent dynamical model:

$$
\begin{aligned}
x_0 &\sim \tau_0(\mathrm{d}x), \\
x_t|x_{t-1} &\sim \tilde{\tau}_t^{y_t}(\mathrm{d}x_t|x_{t-1}), \\
y_t|x_t &\sim g_t(y_t|x_t)
\end{aligned}
$$

Analysis of the nudged particle filter – model mismatch

Nudging as an observation dependent implicit model

In other words, the NuPF = the BPF for the following observation dependent dynamical model:

$$
\begin{aligned}
x_0 &\sim \tau_0(\mathrm{d}x), \\
x_t|x_{t-1} &\sim \tilde{\tau}_t^{y_t}(\mathrm{d}x_t|x_{t-1}), \\
y_t|x_t &\sim g_t(y_t|x_t)
\end{aligned}
$$

Dependence of transition kernel to the observation: Transition kernels are "adapted to the data".

- We have shown that the NuPF has the same convergence rate as the BPF.

- We have shown that the NuPF has the same convergence rate as the BPF.
- The NuPF can be interpreted as a BPF for an implicit, observation-dependent model, which potentially explains its robustness against model mismatch.

- We have shown that the NuPF has the same convergence rate as the BPF.
- The NuPF can be interpreted as a BPF for an implicit, observation-dependent model, which potentially explains its robustness against model mismatch.

Next: Simulations and experiments.

Simulations - I

Lorenz 63 model

Lorenz 63 model is given by the following three-dimensional stochastic differential equation,

$$\begin{aligned}
\mathrm{d}x_1 &= -\mathsf{s}(x_1 - x_2)\mathrm{d}s + \mathrm{d}w_1, \\
\mathrm{d}x_2 &= (\mathsf{r}x_1 - x_2 - x_1x_3)\mathrm{d}s + \mathrm{d}w_2, \\
\mathrm{d}x_3 &= (x_1x_2 - \mathsf{b}x_3)\mathrm{d}s + \mathrm{d}w_3,
\end{aligned}$$

where $\{w_i(s)\}_{s\in(0,\infty)}$ for $i = 1, 2, 3$ are 1-dimensional independent Wiener processes and $(\mathsf{s}, \mathsf{r}, \mathsf{b}) \in \mathbb{R}$ are fixed model parameters.

Simulations - I

Lorenz 63 model - simulation and generating observations

We use the Euler-Maruyama scheme with $\mathsf{T} > 0$ and obtain the system of difference equations,

$$
\begin{aligned}
x_{1,t} &= x_{1,t-1} - \mathsf{T}\mathsf{s}(x_{1,t-1} - x_{2,t-1}) + \sqrt{\mathsf{T}}u_{1,t} \\
x_{2,t} &= x_{2,t-1} + \mathsf{T}(\mathsf{r}x_{1,t-1} - x_{2,t-1} - x_{1,t-1}x_{3,t-1}) + \sqrt{\mathsf{T}}u_{2,t} \\
x_{3,t} &= x_{3,t-1} + \mathsf{T}(x_{1,t-1}x_{2,t-1} - \mathsf{b}x_{3,t-1}) + \sqrt{\mathsf{T}}u_{3,t}
\end{aligned}
$$

where $\{u_{i,t}\}_{t\in\mathbb{N}}$, $i = 1, 2, 3$ are i.i.d. $\mathcal{N}(0, 1)$. We assume that we can only observe the variable $x_{1,t}$ every $t_s = 40$ discrete time steps and contaminated by additive noise:

$$
y_n = k_o x_{1,nt_s} + v_n, \quad n = 1, 2, ...,
$$

where $\{v_n\}_{n\in\mathbb{N}}$ is a sequence of i.i.d. $\mathcal{N}(0, 1)$ and the scale parameter k_o is assumed known.

Simulations - I

Lorenz 63 model - model mismatch

We simulate the system with

$$(\mathsf{s}, \mathsf{r}, \mathsf{b}) = \left(10, 28, \frac{8}{3}\right).$$

Simulations - I

Lorenz 63 model - model mismatch

We simulate the system with

$$(\mathsf{s}, \mathsf{r}, \mathsf{b}) = \left(10, 28, \frac{8}{3}\right).$$

However, when we run the BPF and the NuPF, we set

$$(\mathsf{s}, \mathsf{r}, \mathsf{b}) = \left(10, 28, \frac{8}{3} + \epsilon\right),$$

where $\epsilon = 0.75$.

Simulations - I

Lorenz 63 model - model mismatch

We simulate the system with

$$(\mathsf{s}, \mathsf{r}, \mathsf{b}) = \left(10, 28, \frac{8}{3}\right).$$

However, when we run the BPF and the NuPF, we set

$$(\mathsf{s}, \mathsf{r}, \mathsf{b}) = \left(10, 28, \frac{8}{3} + \epsilon\right),$$

where $\epsilon = 0.75$.

Model uncertainty: Our knowledge about the model is imperfect!

Simulations - I

Lorenz 63 model

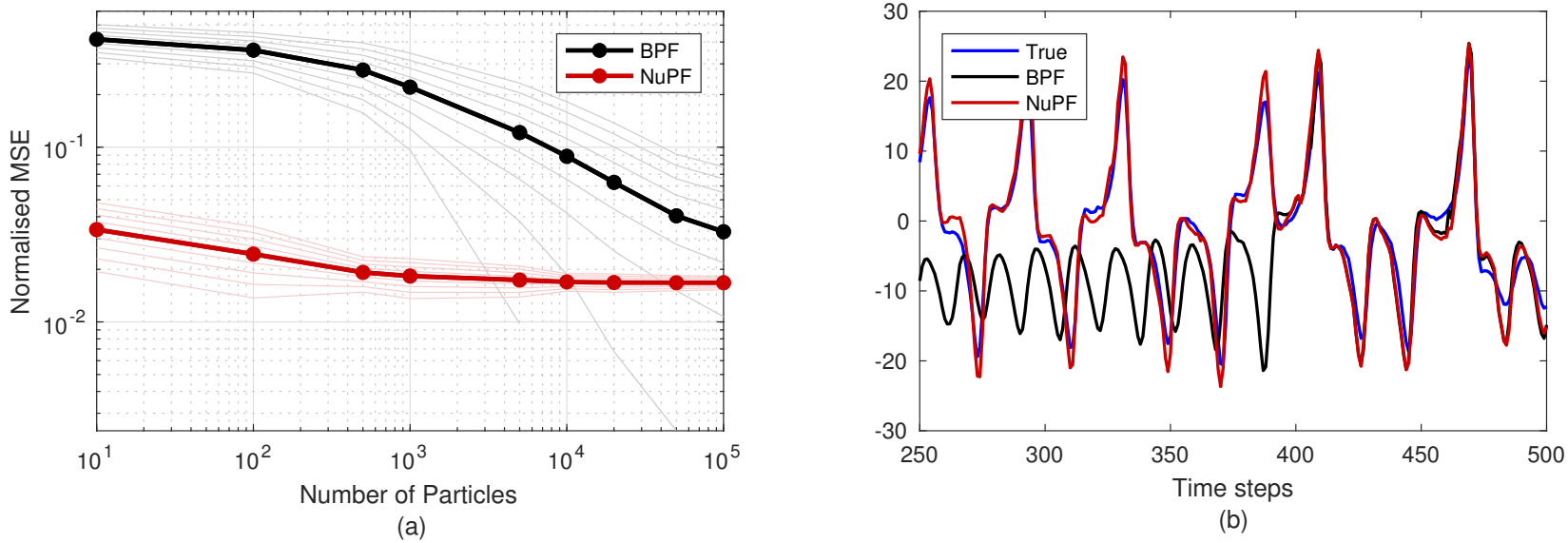


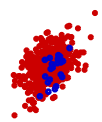
Figure: (a) The results obtained with 1,000 Monte Carlo runs for each $N \in \{10, 100, 500, 1K, 5K, 10K, 20K, 50K, 100K\}$. The dashed lines indicate 1 standard deviation. (b) A sample path from estimation of an *unobserved* dimension (second dimension) in a run where $N = 500$.

Simulations - I

Lorenz 63 model - BPF simulation

Simulations - I

Lorenz 63 model - NuPF simulation: Watch out for the blue particles!



Simulations - II

Lorenz 96 model

We consider Lorenz 96 model which is defined as follows,

$$\mathrm{d}x_i = ((x_{i+1} - x_{i-2})x_{i-1} - x_i + F)\mathrm{d}s + \mathrm{d}w_i$$

where $(w_i(s))_{s\in(0,\infty)}$ are Wiener processes for $i = 1, \ldots, d$.

- We set $F = 8$ which generates chaotic dynamics.

Simulations - II

Lorenz 96 model

We consider Lorenz 96 model which is defined as follows,

$$\mathrm{d}x_i = ((x_{i+1} - x_{i-2})x_{i-1} - x_i + F)\mathrm{d}s + \mathrm{d}w_i$$

where $(w_i(s))_{s\in(0,\infty)}$ are Wiener processes for $i = 1, \ldots, d$.

- We set $F = 8$ which generates chaotic dynamics.
- Circular structure: $x_{-1} = x_{d-1}$, $x_0 = x_d$, and $x_{d+1} = x_1$.

Simulations - II

Lorenz 96 model

We consider Lorenz 96 model which is defined as follows,

$$\mathrm{d}x_i = ((x_{i+1} - x_{i-2})x_{i-1} - x_i + F)\mathrm{d}s + \mathrm{d}w_i$$

where $(w_i(s))_{s\in(0,\infty)}$ are Wiener processes for $i = 1, \ldots, d$.

- We set $F = 8$ which generates chaotic dynamics.
- Circular structure: $x_{-1} = x_{d-1}$, $x_0 = x_d$, and $x_{d+1} = x_1$.

Discretization:

$$x_{i,t} = x_{i,t-1} + \mathsf{T}((x_{i+1,t-1} - x_{i-2,t-1})x_{i-1,t-1} - x_{i,t-1} + F) + \sqrt{\mathsf{T}}u_{i,t}$$

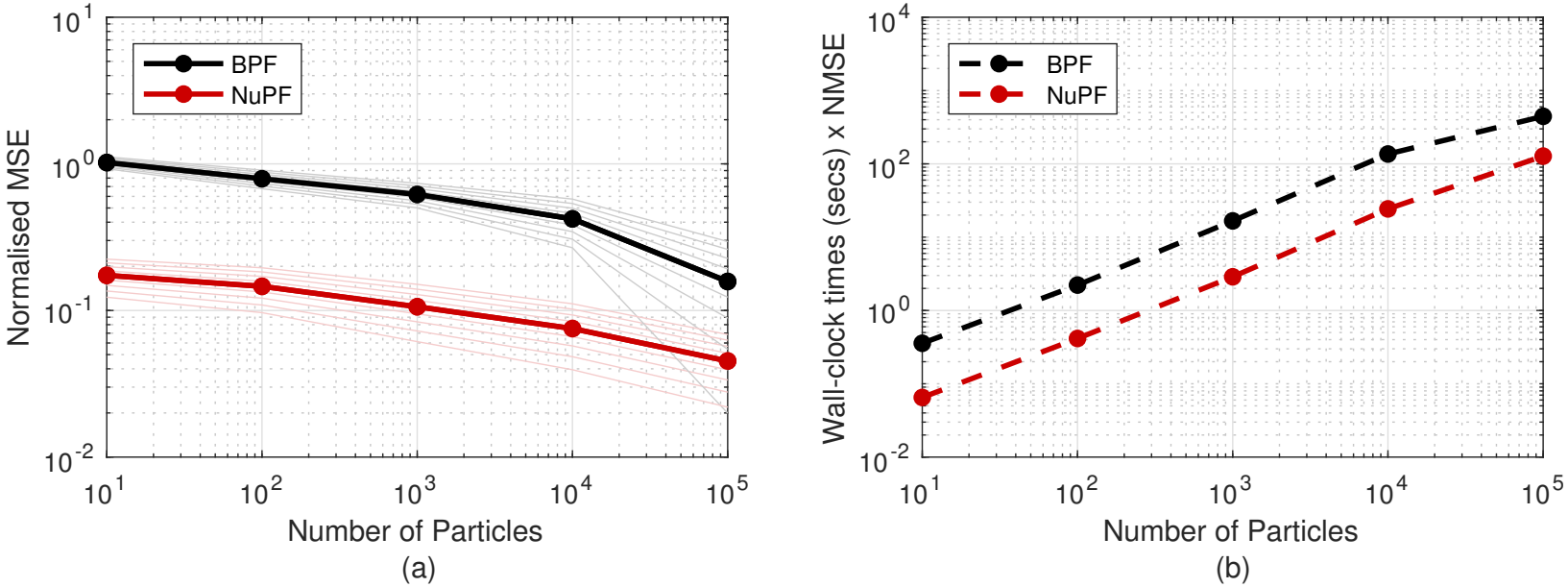
where $u_{i,t}$ are i.i.d $\mathcal{N}(0, 1)$. We observe half of the state variables,

$$y_{k,n} = x_{(2k-1),t_s n} + v_n,$$

where $t_s = 10$ and $(v_n)_{n\in\mathbb{N}}$ are $\mathcal{N}(0, 1)$ i.i.d.

Simulations - II

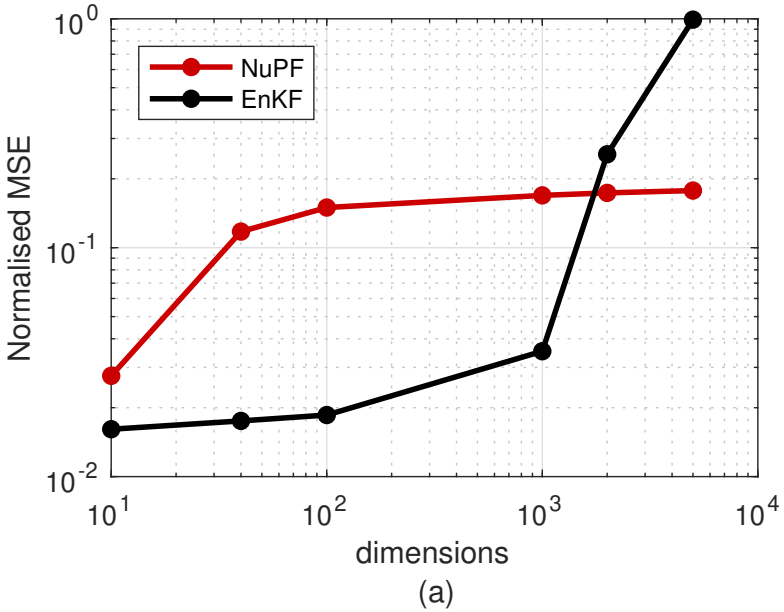
Lorenz 96 model - for fixed dimension $d = 40$

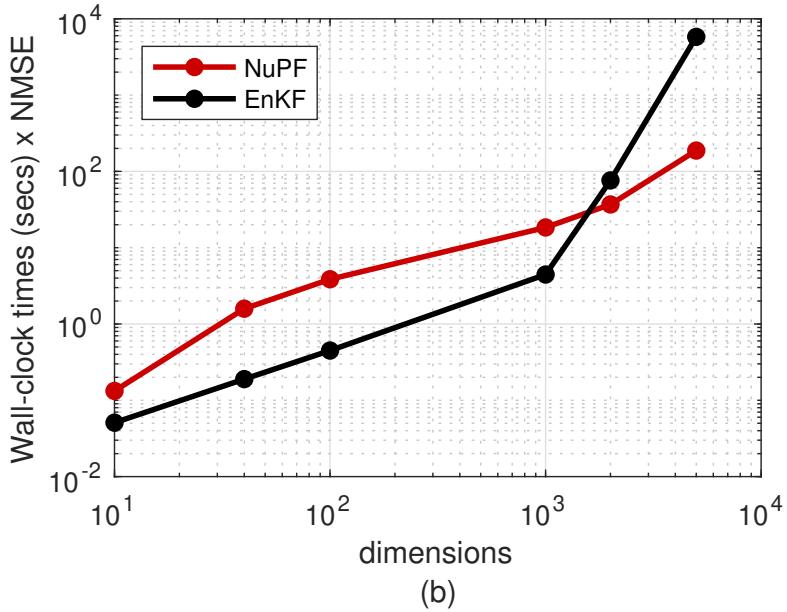


Results are obtained with 1024 Monte Carlo runs for each N.

Simulations - II

Lorenz 96 model - for varying dimensions





Dimensions $d = [10, 40, 100, 1000, 2000, 5000]$ and $N = 500$ kept fixed.
We have run the experiments for 1000 Monte Carlo runs.

Conclusions:

Conclusions:

- We have proposed a particle filter which

Conclusions:

- We have proposed a particle filter which
  - can operate under model mismatch and high-dimensional settings,

Conclusions:

- We have proposed a particle filter which
  - can operate under model mismatch and high-dimensional settings,
  - as computationally efficient as the bootstrap particle filter for most cases,

Conclusions:

- We have proposed a particle filter which
  - can operate under model mismatch and high-dimensional settings,
  - as computationally efficient as the bootstrap particle filter for most cases,
  - has the same convergence rate as the BPF.

Conclusions:

- We have proposed a particle filter which
  - can operate under model mismatch and high-dimensional settings,
  - as computationally efficient as the bootstrap particle filter for most cases,
  - has the same convergence rate as the BPF.

Next: Stochastic optimization.

Stochastic optimization as Bayesian inference

Stochastic optimization

We aim at solving optimization problems of the form

$$\min_{\theta \in \Theta} f(\theta) \quad \text{where} \quad f(\theta) = \sum_{i=1}^{n} f_i(\theta),$$

where $\Theta \subset \mathbb{R}^d$ is the d-dimensional compact search space and n is very large.

Stochastic optimization

We aim at solving optimization problems of the form

$$\min_{\theta \in \Theta} f(\theta) \quad \text{where} \quad f(\theta) = \sum_{i=1}^{n} f_i(\theta),$$

where $\Theta \subset \mathbb{R}^d$ is the d-dimensional compact search space and n is very large.

- Widely encountered in statistics, computer science, machine learning.
- The standard method: Stochastic gradient descent. At iteration t, sample a mini-batch $\mathcal{I}_t \subset \{1, \ldots, n\}$, then take,

$$\theta_t = \text{Proj}_\Theta \left(\theta_{t-1} - \gamma_t \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \nabla f_i(\theta_{t-1}) \right).$$

Stochastic optimization

We aim at solving optimization problems of the form

$$\min_{\theta \in \Theta} f(\theta) \quad \text{where} \quad f(\theta) = \sum_{i=1}^{n} f_i(\theta),$$

where $\Theta \subset \mathbb{R}^d$ is the d-dimensional compact search space and n is very large.

- In this work, we develop a probabilistic view of the problem.

Stochastic optimization

We aim at solving optimization problems of the form

$$\min_{\theta \in \Theta} f(\theta) \quad \text{where} \quad f(\theta) = \sum_{i=1}^{n} f_i(\theta),$$

where $\Theta \subset \mathbb{R}^d$ is the d-dimensional compact search space and n is very large.

- In this work, we develop a probabilistic view of the problem.
- This leads to the use of probabilistic methods.

Stochastic optimization

We aim at solving optimization problems of the form

$$\min_{\theta \in \Theta} f(\theta) \quad \text{where} \quad f(\theta) = \sum_{i=1}^{n} f_i(\theta),$$

where $\Theta \subset \mathbb{R}^d$ is the d-dimensional compact search space and n is very large.

- In this work, we develop a probabilistic view of the problem.
- This leads to the use of probabilistic methods.
- Advantages:
  - Probabilistic methods (such as Kalman or extended Kalman based algorithms) produce more stable behaviour than standard optimizers.

Stochastic optimization

We aim at solving optimization problems of the form

$$\min_{\theta \in \Theta} f(\theta) \quad \text{where} \quad f(\theta) = \sum_{i=1}^{n} f_i(\theta),$$

where $\Theta \subset \mathbb{R}^d$ is the d-dimensional compact search space and n is very large.

- In this work, we develop a probabilistic view of the problem.
- This leads to the use of probabilistic methods.
- Advantages:
  - Probabilistic methods (such as Kalman or extended Kalman based algorithms) produce more stable behaviour than standard optimizers.
  - We can use the probabilistic view to utilize standard numerical methods to solve the problem, e.g., sampling algorithms.

Stochastic optimization as Bayesian inference

Given a cost function $f(\theta) = \sum_{i=1}^{n} f_i(\theta)$, assume that we construct a sequence of non-overlapping subsets (index sets) $(\mathcal{I}_t)_{1 \leq t \leq T}$ of $[n] = \{1, \ldots, n\}$ where $|\mathcal{I}_t| = K$, $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$, and $\bigcup_t \mathcal{I}_t = [n]$.

Then we construct

$$G_t(\theta) = \exp\left(- \sum_{i \in \mathcal{I}_t} f_i(\theta) \right),$$

for $t = 1, \ldots, T$.

Stochastic optimization as Bayesian inference

Given a cost function $f(\theta) = \sum_{i=1}^{n} f_i(\theta)$, assume that we construct a sequence of non-overlapping subsets (index sets) $(\mathcal{I}_t)_{1 \leq t \leq T}$ of $[n] = \{1, \ldots, n\}$ where $|\mathcal{I}_t| = K$, $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$, and $\bigcup_t \mathcal{I}_t = [n]$.

Then we construct

$$G_t(\theta) = \exp\left(-\sum_{i \in \mathcal{I}_t} f_i(\theta)\right),$$

for $t = 1, \ldots, T$.

The functions $(G_t)_{1 \leq t \leq T}$ are termed *potential functions* and can be thought of as *likelihoods*, defined compactly for mini-batches.

Stochastic optimization as Bayesian inference

Given a prior measure π_0, define the recursion,

$$\pi_t(\mathrm{d}\theta) = \pi_{t-1}(\mathrm{d}\theta)\frac{G_t(\theta)}{\int_\Theta G_t(\theta)\pi_{t-1}(\mathrm{d}\theta)}.$$

for $1 \leq t \leq T$.

Stochastic optimization as Bayesian inference

Given a prior measure π_0, define the recursion,

$$\pi_t(\mathrm{d}\theta) = \pi_{t-1}(\mathrm{d}\theta)\frac{G_t(\theta)}{\int_\Theta G_t(\theta)\pi_{t-1}(\mathrm{d}\theta)}.$$

for $1 \leq t \leq T$.

This recursion can be seen as a *sequential Bayes update*.

Stochastic optimization as Bayesian inference

Given a prior measure π_0, define the recursion,

$$\pi_t(\mathrm{d}\theta) = \pi_{t-1}(\mathrm{d}\theta)\frac{G_t(\theta)}{\int_\Theta G_t(\theta)\pi_{t-1}(\mathrm{d}\theta)}.$$

for $1 \leq t \leq T$.

This recursion can be seen as a *sequential Bayes update*.

The key observation:

$$\frac{\mathrm{d}\pi_T}{\mathrm{d}\pi_0}(\theta) \propto \prod_{t=1}^{T} G_t(\theta).$$

Assumption. We assume that the functions G_t are bounded.

Stochastic optimization as Bayesian inference

As a result

$$\operatorname*{argmax}_{\theta\in\Theta} \frac{\mathrm{d}\pi_T}{\mathrm{d}\pi_0}(\theta) = \operatorname*{argmin}_{\theta\in\Theta} \sum_{i=1}^{n} f_i(\theta)$$

since

$$\frac{\mathrm{d}\pi_T}{\mathrm{d}\pi_0}(\theta) \propto \prod_{t=1}^{T} G_t(\theta) \quad \text{and} \quad G_t(\theta) = \exp\left(-\sum_{i\in\mathcal{I}_t} f_i(\theta)\right).$$

Consequently, when π_0 is a uniform distribution on compact Θ

$$\operatorname*{argmax}_{\theta\in\Theta} \pi_T(\theta) = \operatorname*{argmin}_{\theta\in\Theta} \sum_{i=1}^{n} f_i(\theta).$$

The Gaussian case

Sequential Bayes as optimization (Akyıldız, Elvira, and Míguez 2018).

Consider the following model,

$$\pi_0(\theta) = \mathcal{N}(\theta; \theta_0, V_0),$$
$$G_t(\theta) = \mathcal{N}(y_t; x_t^\top \theta, \lambda),$$

then, $\pi_t(\theta) = \mathcal{N}(\theta; \theta_t, V_t)$ where

$$\theta_t = \theta_{t-1} + \frac{V_{t-1} x_t (y_t - x_t^\top \theta_{t-1})}{\lambda + x_t^\top V_{t-1} x_t} \quad (3)$$

and,

$$V_t = V_{t-1} - \frac{V_{t-1} x_t x_t^\top V_{t-1}}{\lambda + x_t^\top V_{t-1} x_t}.$$

Consider the following incremental proximal method recursion for

$$f_t(\theta) = \frac{1}{2}(y_t - x_t^\top \theta)^2$$

which is given by,

$$\theta_t = \mathrm{prox}_{\lambda, f_t}(\theta_{t-1}),$$
$$= \operatorname*{argmin}_\theta \frac{1}{2}(y_t - x_t^\top \theta)^2 + \frac{\lambda}{2}\|\theta - \theta_{t-1}\|^2_{2, V^{-1}},$$

which results in

$$\theta_t = \theta_{t-1} + \frac{V x_t (y_t - x_t^\top \theta_{t-1})}{\lambda + x_t^\top V x_t}. \quad (4)$$

In this special case, sequential Bayesian inference corresponds to the incremental proximal method with a **variable metric**.

The Gaussian case

Sequential Bayes as optimization (Akyıldız, Elvira, and Míguez 2018).

- This interpretation can be extended to the nonlinear case, where one can use extended Kalman updates for nonlinear optimization.
  - Produces much more numerically stable and robust (e.g. insensitive to initial parameters) behaviour compared to usual schemes.
  - Automatic adaptation of the parameters of the algorithm (e.g., as a covariance matrix).
- Further work on *incremental proximal gradient methods* shows that this is an interesting research direction (Akyildiz et al. 2019).

$$\operatorname*{argmax}_{\theta \in \Theta} \pi_T(\theta) = \operatorname*{argmin}_{\theta \in \Theta} \sum_{i=1}^{n} f_i(\theta).$$

$$\operatorname*{argmax}_{\theta \in \Theta} \pi_T(\theta) = \operatorname*{argmin}_{\theta \in \Theta} \sum_{i=1}^{n} f_i(\theta).$$

- As we have discussed, the probabilistic interpretation can be useful to obtain numerically stable and robust optimization methods.

$$\operatorname*{argmax}_{\theta\in\Theta} \pi_T(\theta) = \operatorname*{argmin}_{\theta\in\Theta} \sum_{i=1}^{n} f_i(\theta).$$

- As we have discussed, the probabilistic interpretation can be useful to obtain numerically stable and robust optimization methods.
- We can also use the interpretation to obtain sampling methods to solve the general problem.

$$\operatorname*{argmax}_{\theta\in\Theta} \pi_T(\theta) = \operatorname*{argmin}_{\theta\in\Theta} \sum_{i=1}^{n} f_i(\theta).$$

- As we have discussed, the probabilistic interpretation can be useful to obtain numerically stable and robust optimization methods.
- We can also use the interpretation to obtain sampling methods to solve the general problem.
- The recursion,

$$\pi_t(\mathrm{d}\theta) = \pi_{t-1}(\mathrm{d}\theta)\frac{G_t(\theta)}{\int_\Theta G_t(\theta)\pi_{t-1}(\mathrm{d}\theta)}$$

  suggests a *sequential* sampling method.

$$\operatorname*{argmax}_{\theta\in\Theta} \pi_T(\theta) = \operatorname*{argmin}_{\theta\in\Theta} \sum_{i=1}^{n} f_i(\theta).$$

- As we have discussed, the probabilistic interpretation can be useful to obtain numerically stable and robust optimization methods.
- We can also use the interpretation to obtain sampling methods to solve the general problem.
- The recursion,

$$\pi_t(\mathrm{d}\theta) = \pi_{t-1}(\mathrm{d}\theta)\frac{G_t(\theta)}{\int_\Theta G_t(\theta)\pi_{t-1}(\mathrm{d}\theta)}$$

  suggests a *sequential* sampling method.

Next: We are going to develop a sequential Bayesian inference method based on sampling.

A sampler for stochastic optimization

We want to develop a sampler to simulate

$$\pi_t(\mathrm{d}\theta) = \pi_{t-1}(\mathrm{d}\theta)\frac{G_t(\theta)}{\int_\Theta G_t(\theta)\pi_{t-1}(\mathrm{d}\theta)}.$$

The main difficulty is that this is a static problem, which means samples will degenerate in a couple of iterations.

- A naive sampling method: Given $\{\theta_{t-1}^{(i)}\}_{i=1}^N$,

$$\theta_t^{(i)} \sim \sum_{i=1}^N w_t^{(i)} \delta_{\theta_{t-1}^{(i)}}(\mathrm{d}\theta), \quad \text{where} \quad w_t^{(i)} \propto G_t(\theta_{t-1}^{(i)}).$$

- Suffers from sample impoverishment as we do not generate new samples: we eventually end up with one sample.

A sampler for stochastic optimization

We need a way to *shake* the particles, without introducing too much error.

- Use a jittering kernel (Crisan and Míguez 2014):

$$\kappa(\mathrm{d}\theta|\theta') = (1-\epsilon_N)\delta_{\theta'}(\mathrm{d}\theta) + \epsilon_N \tau(\mathrm{d}\theta|\theta'), \tag{5}$$

  to sample new particles $\theta_t^{(i)} \sim \kappa(\cdot|\theta_{t-1}^{(i)})$.
- We usually choose $\epsilon_N \leq \frac{1}{\sqrt{N}}$.
- τ can be simple, i.e., multivariate Gaussian or multivariate t distribution.

A sampler for stochastic optimization

The sampler:

- Sample $\theta_0^{(i)} \sim \pi_0$ for $i = 1, \ldots, N$.
- For $t \geq 1$:
  - Jitter by generating samples

$$\hat{\theta}_t^{(i)} \sim \kappa(\mathrm{d}\theta | \theta_{t-1}^{(i)}) \qquad \text{for} \quad i = 1, \ldots, N.$$

  - Compute weights,

$$w_t^{(i)} = \frac{G_t(\hat{\theta}_t^{(i)})}{\sum_{i=1}^N G_t(\hat{\theta}_t^{(i)})} \quad \text{for} \quad i = 1, \ldots, N.$$

  - Resample by drawing N i.i.d. samples,

$$\theta_t^{(i)} \sim \hat{\pi}_t^N(\mathrm{d}\theta) := \sum_{i=1}^N w_t^{(i)} \delta_{\hat{\theta}_t^{(i)}}(\mathrm{d}\theta), \qquad i = 1, \ldots, N.$$

A sampler for stochastic optimization

Given samples, how to identify the maxima?

A sampler for stochastic optimization

Given samples, how to identify the maxima?

- Compute a kernel density estimator $\mathsf{p}_t^N(\theta)$:

$$\mathsf{p}_t^N(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathsf{k}_h(\theta - \theta_t^{(i)}).$$

- Choose the empirical maximum:

$$\theta_t^{\star,N} = \operatorname*{argmax}_{i \in \{1,\ldots,N\}} \mathsf{p}_t^N(\theta_t^{(i)}).$$

  This is an $\mathcal{O}(N^2)$ operation.

- Under suitable regularity conditions

$$\lim_{N \to \infty} \pi_t(\theta_t^{\star,N}) = \pi_t(\theta_t^{\star}),$$

  where $\theta_t^{\star} \in \operatorname{argmax}_{\theta \in \Theta} \pi_t(\theta)$.

However...

However...

- The $\mathcal{O}(N^2)$ cost of the kernel density estimator suggests that we should not be in a setup where we need very large N.

However...

- The $\mathcal{O}(N^2)$ cost of the kernel density estimator suggests that we should not be in a setup where we need very large N.
- With small N, however, this sampler can take very long time to move to a global minimum.

However...

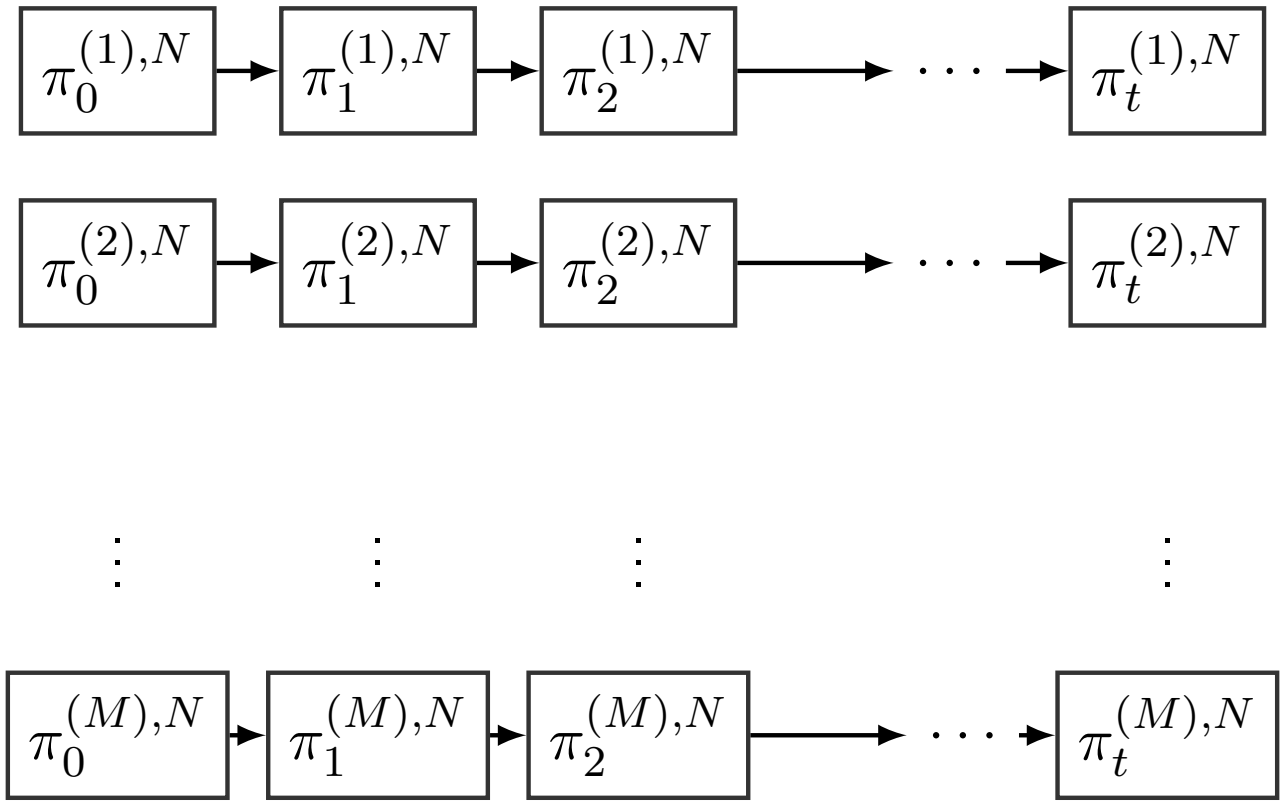
- The $\mathcal{O}(N^2)$ cost of the kernel density estimator suggests that we should not be in a setup where we need very large N.
- With small N, however, this sampler can take very long time to move to a global minimum.
- Again, with small N, there might be practical problems with representing multiple modes.

However...

- The $\mathcal{O}(N^2)$ cost of the kernel density estimator suggests that we should not be in a setup where we need very large N.
- With small N, however, this sampler can take very long time to move to a global minimum.
- Again, with small N, there might be practical problems with representing multiple modes.
- Idea: Use M independent, parallel samplers with small N and select the *best* one to obtain a global estimator with low cost.

A parallel SMC sampler for stochastic optimization

(Akyildiz, Crisan, and Míguez 2018)



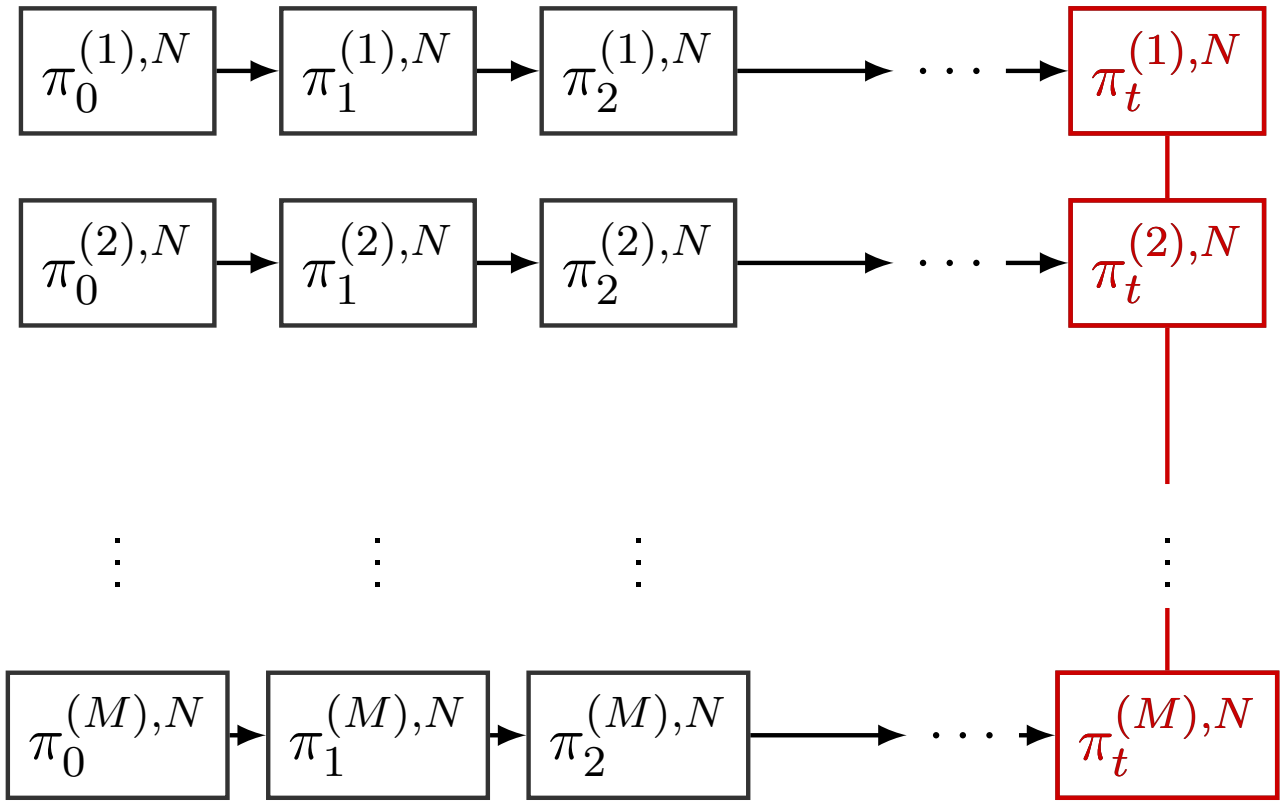
At iteration t, compute the marginal likelihood,

$$Z_{1:t}^{(m),N} = Z_{1:t-1}^{(m),N} \times Z_t^{(m),N} \quad \text{where} \quad Z_t^{(m),N} = \frac{1}{N} \sum_{i=1}^{N} G_t^{(m)}(\hat{\theta}_t^{(i,m)}).$$

for $m = 1, \ldots, M$.

A parallel SMC sampler for stochastic optimization

(Akyildiz, Crisan, and Míguez 2018)



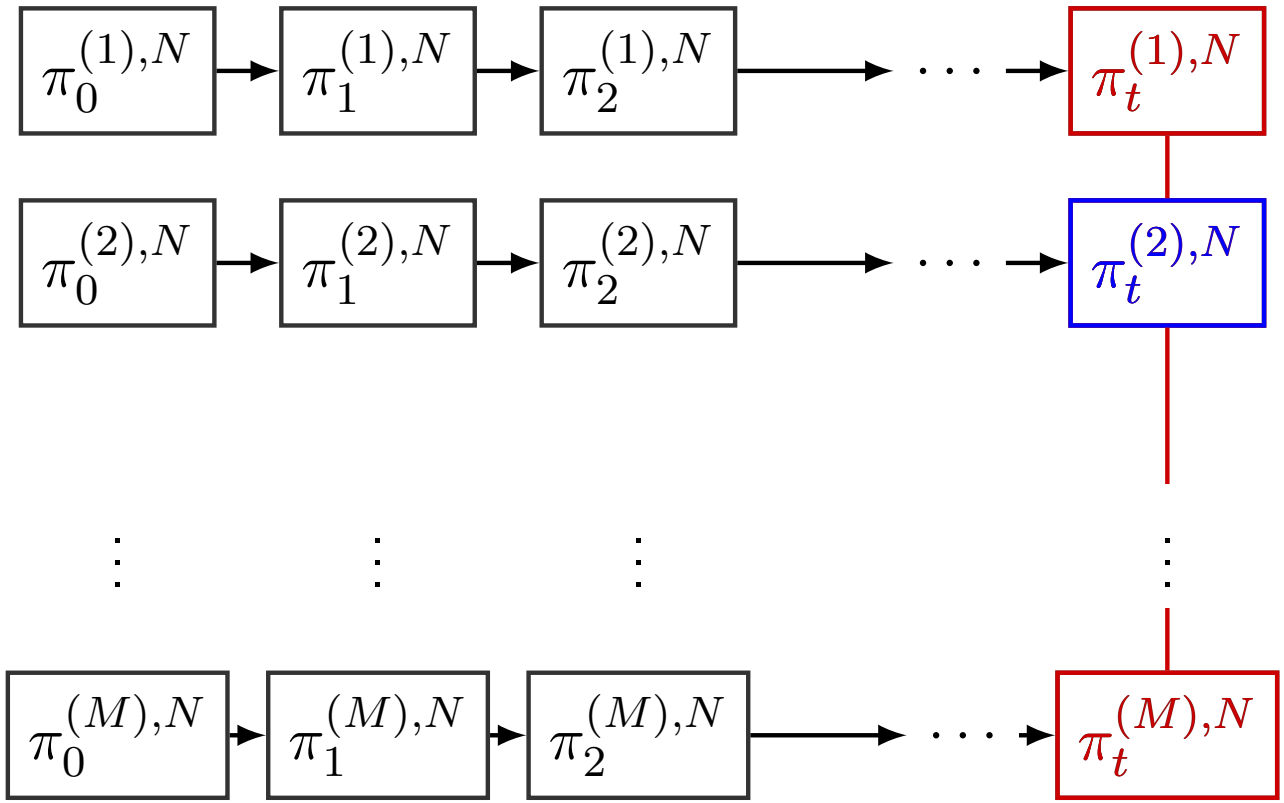
At iteration t, compute the marginal likelihood,

$$Z_{1:t}^{(m),N} = Z_{1:t-1}^{(m),N} \times Z_t^{(m),N} \quad \text{where} \quad Z_t^{(m),N} = \frac{1}{N} \sum_{i=1}^{N} G_t^{(m)}(\hat{\theta}_t^{(i,m)}).$$

for $m = 1, \ldots, M$.

A parallel SMC sampler for stochastic optimization

(Akyildiz, Crisan, and Míguez 2018)



At iteration t, compute the marginal likelihood,

$$Z_{1:t}^{(m),N} = Z_{1:t-1}^{(m),N} \times Z_t^{(m),N} \quad \text{where} \quad Z_t^{(m),N} = \frac{1}{N}\sum_{i=1}^{N} G_t^{(m)}(\hat{\theta}_t^{(i,m)}).$$

for $m = 1, \ldots, M$. Choose: $m_t^\star = \text{argmax}_{m\in\{1,\ldots,M\}} Z_{1:t}^{(m),N}$.

- We have presented a general SMC sampler to solve a stochastic optimization problem.

- We have presented a general SMC sampler to solve a stochastic optimization problem.
- We have the following theoretical guarantee for a single sampler

$$\lim_{N \to \infty} \pi_t(\theta_t^{\star,N}) = \pi_t(\theta_t^\star),$$

where $\theta_t^\star \in \operatorname{argmax}_{\theta \in \Theta} \pi_t(\theta)$.

Next: Simulations.

Simulations

Extended Kalman update as nonlinear proximal optimization

In order to compare the proximal methods and extended Kalman methods for nonlinear regression , we formulate

$$\min_{\theta \in \mathbb{R}^d} f(\theta) \quad \text{where} \quad f(\theta) = \sum_{k=1}^{n} f_k(\theta)$$

where $f_k(\theta) = (y_k - g_k(\theta))^2$ with

$$g_k(\theta) = \frac{1}{1 + \exp(-\alpha - \beta^\top x_k)}$$

where $x_k \in \mathbb{R}^{d-1}$ are inputs and $\theta = (\alpha, \beta) \in \mathbb{R}^d$. We set $d = 21$ and run the EKF and an approximate nonlinear incremental proximal method.

Simulations

Extended Kalman update as nonlinear proximal optimization

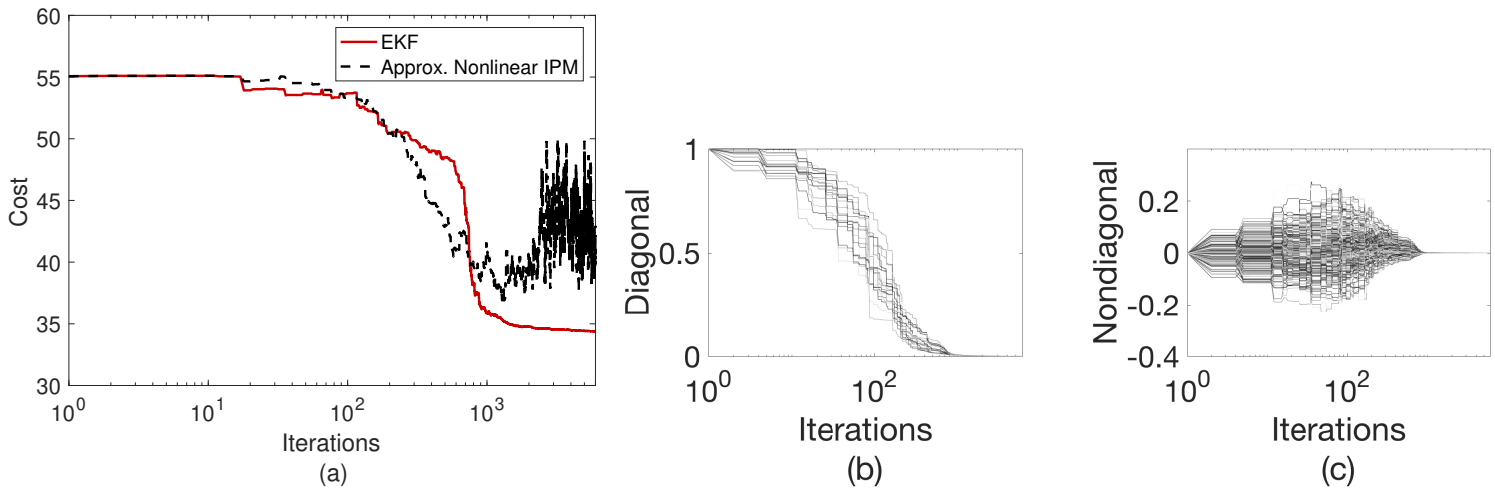


Figure: Results on fitting a sigmoid function using EKF and approximate nonlinear IPM. From (a), it can be seen that the approximate nonlinear IPM proceeds towards the minimum but suffers from instability while the EKF proceeds in a stable way. From (b)-(c), it can be seen that the entries of the diagonal and nondiagonal parts of the covariance matrix V_k converge to zero which is the reason why the EKF does not suffer from instability.

Simulations

a cost function with multiple global minima

In this experiment, we tackle the problem

$$\min_{\theta \in \mathbb{R}^2} f(\theta),$$

where

$$f(\theta) = \sum_{i=1}^{n} f_i(\theta) \qquad \text{and} \qquad f_i(\theta) = -\frac{1}{\lambda} \log \left(\sum_{k=1}^{4} \mathcal{N}(\theta; m_{i,k}, R) \right),$$

with $\lambda = 10$, $R = rI$ with $r = 0.2$. We choose the means $m_{i,k}$ randomly, namely $m_{i,k} \sim \mathcal{N}(m_{i,k}; m_k, \sigma^2)$ where,

$$m_1 = [4, 4]^\top, \quad m_2 = [-4, -4]^\top, \quad m_3 = [-4, 4]^\top, \quad m_4 = [4, -4]^\top,$$

and $\sigma^2 = 0.5$. We have $n = 1,000$. Note that each $f_i(\theta)$ models a mini-batch in this scenario and we choose $K = 1$ in our algorithm.

Simulations

a cost function with multiple global minima

- Uniform prior measure $\pi_0(\theta) = \mathcal{U}([-a, a] \times [-a, a])$ with $a = 50$.
- We run $M = 100$ samplers, each with each $N = 50$ particles, yielding a total number of particles $MN = 5,000$.
- We choose a Gaussian jittering scheme; specifically, the jittering kernel is defined as

$$\kappa(\mathrm{d}\theta|\theta') = (1 - \epsilon_N)\delta_{\theta'}(\mathrm{d}\theta) + \epsilon_N \mathcal{N}(\theta; \theta', \sigma_j^2)\mathrm{d}\theta, \qquad (6)$$

where $\epsilon_N \leq 1/\sqrt{N}$ and $\sigma_j^2 = 0.5$.

Simulations

a cost function with multiple global minima

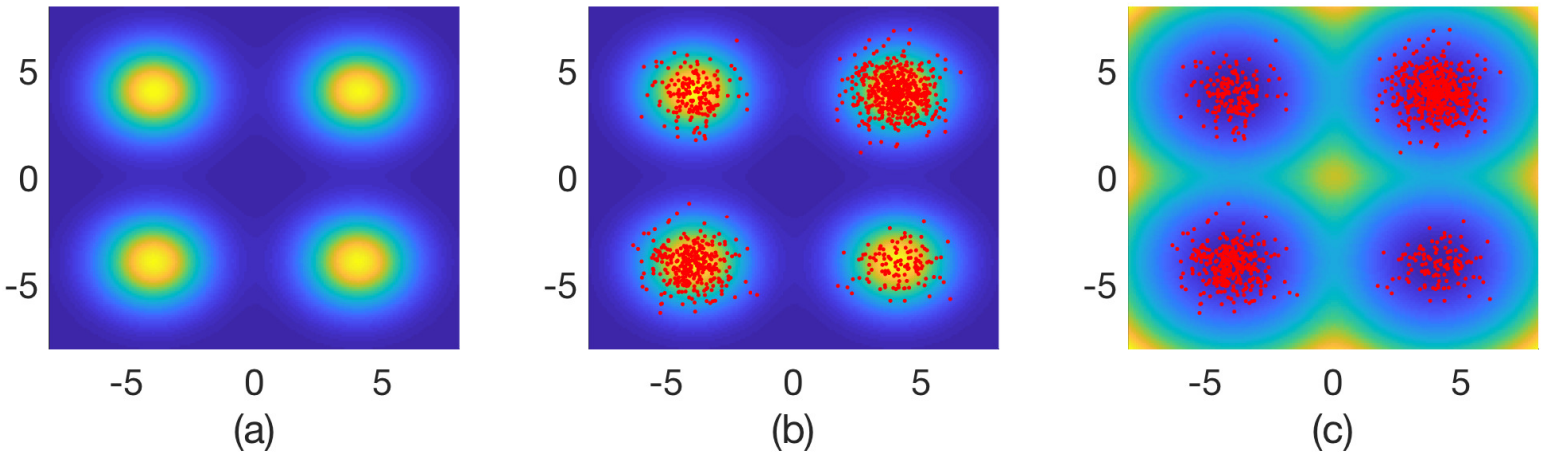


Figure: An illustration of the performance of the proposed algorithm for a cost function with four global minima. (a) The plot of $\pi_T(\theta) \propto \exp(-f(\theta))$. The blue regions indicate low values. It can be seen that there are four global maxima. (b) Samples drawn by the PSMCO at a single time instant. (c) The plot of the samples together with the actual cost function $f(\theta)$.

Simulations

a nonconvex global optimization problem

In this experiment, we address the problem,

$$\min_{\theta \in \mathbb{R}^2} f(\theta) := \sum_{i=1}^{n} (y_i - g_i(\theta))^2, \tag{7}$$

where

$$g_i(\theta) = \frac{1}{1 + \exp(-\theta_1 - \theta_2 x_i)}, \tag{8}$$

with $x_i \in \mathbb{R}$, $f_i(\theta) = (y_i - g_i(\theta))^2$ and $\theta = [\theta_1, \theta_2]^\top$. The function g_i is called as the sigmoid function.

Simulations

a nonconvex global optimization problem

- We have $n = 100,000$. We choose $M = 25$ and $MN = 1,000$, leading to $N = 40$ particles for every sampler. The mini-batch size is $K = 100$.
- The jittering kernel κ is defined in the same way as before, where the Gaussian pdf has a variance chosen as the ratio of the dataset size L to the mini-batch size K, i.e., $\sigma_j^2 = n/K$, which yields a rather large variance $\sigma_j^2 = 1000$.
- We use a Gaussian kernel density with bandwidth $h = 1$.

Simulations

a nonconvex global optimization problem

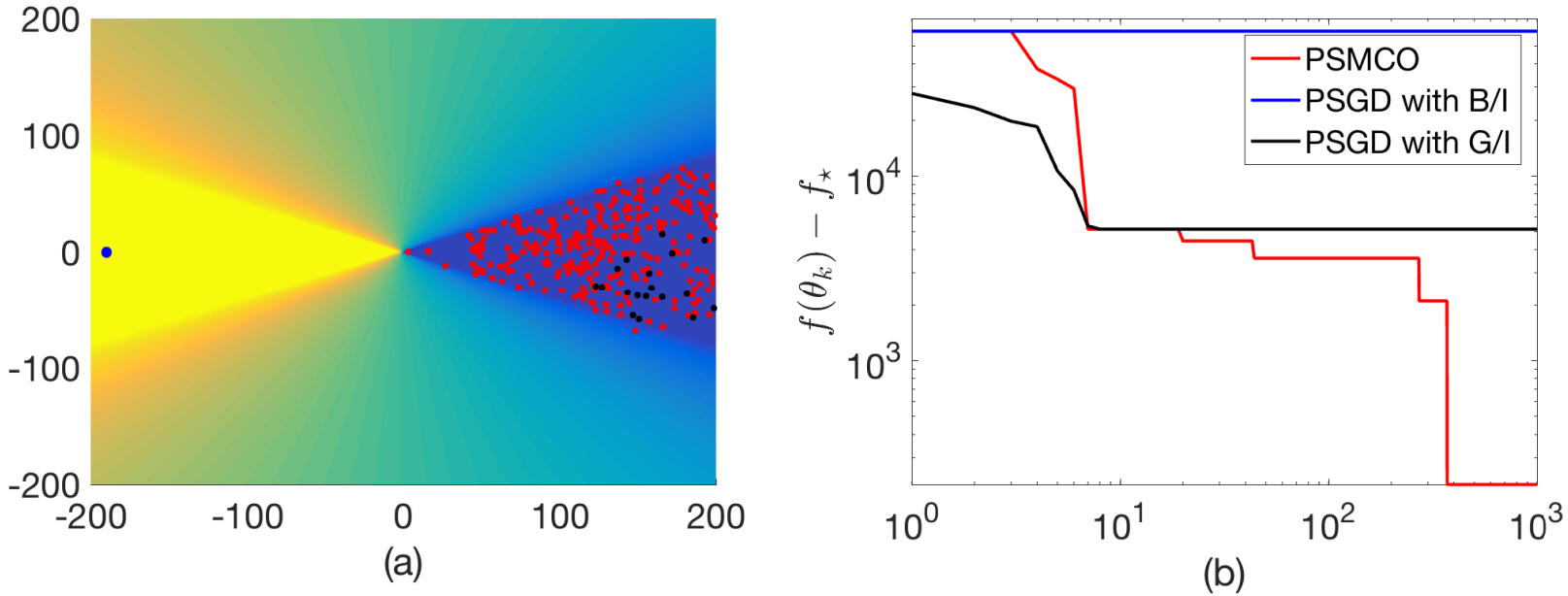


Figure: (a) The cost function and a snapshot of samples from 50th iteration of PSMCO, PSGD with bad initialization (blue point) and PSGD with good initialization (black points). (b) The minimization performance of each algorithm.

Conclusions:

- We have shown that a probabilistic view of the optimization problem may help us to define principled new optimization methods.

Conclusions:

- We have shown that a probabilistic view of the optimization problem may help us to define principled new optimization methods.
- It can also help us to propose new optimizers using inference methods: a global *stochastic* zeroth-order optimization scheme based on sequential Monte Carlo.

Conclusions:

- We have shown that a probabilistic view of the optimization problem may help us to define principled new optimization methods.
- It can also help us to propose new optimizers using inference methods: a global *stochastic* zeroth-order optimization scheme based on sequential Monte Carlo.
- Probabilistic solutions to the optimization problem also enable us to quantify uncertainty.

Next: Future work for filtering and optimization.

Future work on stochastic filtering

- As the nudging increases the efficiency of filters significantly, exploration of the idea within other particles schemes & parameter estimation methods is a natural direction.
- Understanding why nudging improves performance significantly in high-dimensional systems.
- A detailed analysis of model misspecification and the described alternative model.
- Exploration of the nudging in nonlinear Kalman-type filters.

Future work on stochastic optimization

- Developing new Kalman-based stochastic optimization schemes, e.g., (Akyildiz et al. 2019).
  - Variants of proximal algorithms, which is a very rich family, can be cast into a probabilistic framework.
- Using other approximate inference methods to solve the probabilistic recursion.

Thanks!

References I

- Akyildiz, Ömer Deniz, Dan Crisan, and Joaquín Míguez (2018). "Parallel sequential Monte Carlo for stochastic optimization". In: *arXiv preprint arXiv:1811.09469*.
- Akyıldız, Ömer Deniz, Víctor Elvira, and Joaquín Míguez (2018). "The incremental proximal method: A probabilistic perspective". In: *Proc. of IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4279–4283.
- Akyıldız, Ömer Deniz and Joaquín Míguez (2017). "Nudging the particle filter". In: *arXiv preprint arXiv:1708.07801*.
- Akyildiz, Ömer Deniz et al. (2019). "A probabilistic incremental proximal gradient method". In: *arXiv preprint arXiv:1812.01655*.
- Andrieu, Christophe, Arnaud Doucet, and Roman Holenstein (2010). "Particle Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3, pp. 269–342.

References II

- Crisan, Dan and Joaquin Miguez (2018). "Nested particle filters for online parameter estimation in discrete-time state-space Markov models". In: *Bernoulli* 24.4A, pp. 3039–3086.
- Crisan, Dan, Joaquín Míguez, et al. (2014). "Particle-kernel estimation of the filter density in state-space models". In: *Bernoulli* 20.4, pp. 1879–1929.
- Del Moral, Pierre and Laurent Miclo (2000). "Branching and interacting particle systems approximations of Feynman-Kac formulae with applications to non-linear filtering". In: *Séminaire de probabilités XXXIV*. Springer, pp. 1–145.
- Van Leeuwen, Peter Jan (2009). "Particle filtering in geophysical systems". In: *Monthly Weather Review* 137.12, pp. 4089–4114.

—backup slides—

Analysis of the nudged particle filter

A result specific to gradient steps

Assumption 2. *The gradient of the likelihood is bounded. In particular, there are constants* $G_t < \infty$ *such that*

$$\|\nabla_x g_t(x)\|_2 \leq G_t < \infty \quad \text{for } x \in \mathsf{X}, t = 1, 2, \ldots, T.$$

Lemma 1. *Choose the number of nudged particles,* $M > 0$*, and a sequence of step-sizes,* $\gamma_t > 0$*, in such a way that* $\sup_{1 \leq t \leq T} \gamma_t M \leq \sqrt{N}$ *for some* $T < 0$*. If Assumption 2 holds and* φ *is a Lipschitz test function, then the error introduced by the batch gradient nudging step with* $|\mathcal{I}_t| = M$ *can be bounded as,*

$$\left\| (\varphi, \xi_t^N) - (\varphi, \tilde{\xi}_t^N) \right\|_p \leq \frac{L G_t}{\sqrt{N}},$$

where L *is the Lipschitz constant of* φ*, for every* $t = 1, \ldots, T$*.*

Analysis of the nudged particle filter

Convergence using gradient steps

Theorem 2. *Let $y_{1:T}$ be an arbitrary but fixed sequence of observations, with $T < \infty$, and choose a sequence of step sizes $\gamma_t > 0$ and an integer M such that*

$$\sup_{1 \leq t \leq T} \gamma_t M \leq \sqrt{N}.$$

Let π_t^N denote the filter approximation obtained with a NuPF with batch gradient nudging. If Assumptions 1 and 2 are satisfied and $|\mathcal{I}_t| = M$, then

$$\|(\varphi, \pi_t^N) - (\varphi, \pi_t)\|_p \leq \frac{c_{t,p}\|\varphi\|_\infty}{\sqrt{N}} \tag{9}$$

for every $t = 1, 2, ..., T$, any bounded Lipschitz function φ, some constant $c_{t,p} < \infty$ independent of N for any integer $p \geq 1$.

Model inference of a stochastic volatility model

We consider the stochastic volatility model,

$$x_0 \sim \mathcal{N}\left(x_0; \mu, \frac{\sigma_v^2}{1-\phi^2}\right), \tag{10}$$

$$x_t | x_{t-1} \sim \mathcal{N}(x_t; \mu + \phi(x_{t-1} - \mu), \sigma_v^2), \tag{11}$$

$$y_t | x_t \sim \mathcal{N}(y_t; 0, \exp(x_t)), \tag{12}$$

where $\mu \in \mathbb{R}$, $\sigma_v \in \mathbb{R}_+$, and $\phi \in (-1, 1)$.

Model inference of a stochastic volatility model

We consider the stochastic volatility model,

$$x_0 \sim \mathcal{N}\left(x_0; \mu, \frac{\sigma_v^2}{1-\phi^2}\right), \tag{10}$$

$$x_t | x_{t-1} \sim \mathcal{N}(x_t; \mu + \phi(x_{t-1} - \mu), \sigma_v^2), \tag{11}$$

$$y_t | x_t \sim \mathcal{N}(y_t; 0, \exp(x_t)), \tag{12}$$

where $\mu \in \mathbb{R}$, $\sigma_v \in \mathbb{R}_+$, and $\phi \in (-1, 1)$.

- The sequence $(x_t)_{1 \leq t \leq T}$ is called as the log-volatility.

Model inference of a stochastic volatility model

We consider the stochastic volatility model,

$$x_0 \sim \mathcal{N}\left(x_0; \mu, \frac{\sigma_v^2}{1-\phi^2}\right), \tag{10}$$

$$x_t | x_{t-1} \sim \mathcal{N}(x_t; \mu + \phi(x_{t-1} - \mu), \sigma_v^2), \tag{11}$$

$$y_t | x_t \sim \mathcal{N}(y_t; 0, \exp(x_t)), \tag{12}$$

where $\mu \in \mathbb{R}$, $\sigma_v \in \mathbb{R}_+$, and $\phi \in (-1, 1)$.

- The sequence $(x_t)_{1 \le t \le T}$ is called as the log-volatility.
- $(y_t)_{1 \le t \le T}$ are observations, which are log-returns.

Given the historical price sequence $s_0, \ldots, s_T$, we set,

$$y_t = 100 \log(s_t / s_{t-1}), \quad \text{for } 1 \le t \le T.$$

Model inference of a stochastic volatility model

We consider the stochastic volatility model,

$$x_0 \sim \mathcal{N}\left(x_0; \mu, \frac{\sigma_v^2}{1-\phi^2}\right), \tag{10}$$

$$x_t | x_{t-1} \sim \mathcal{N}(x_t; \mu + \phi(x_{t-1} - \mu), \sigma_v^2), \tag{11}$$

$$y_t | x_t \sim \mathcal{N}(y_t; 0, \exp(x_t)), \tag{12}$$

where $\mu \in \mathbb{R}$, $\sigma_v \in \mathbb{R}_+$, and $\phi \in (-1, 1)$.

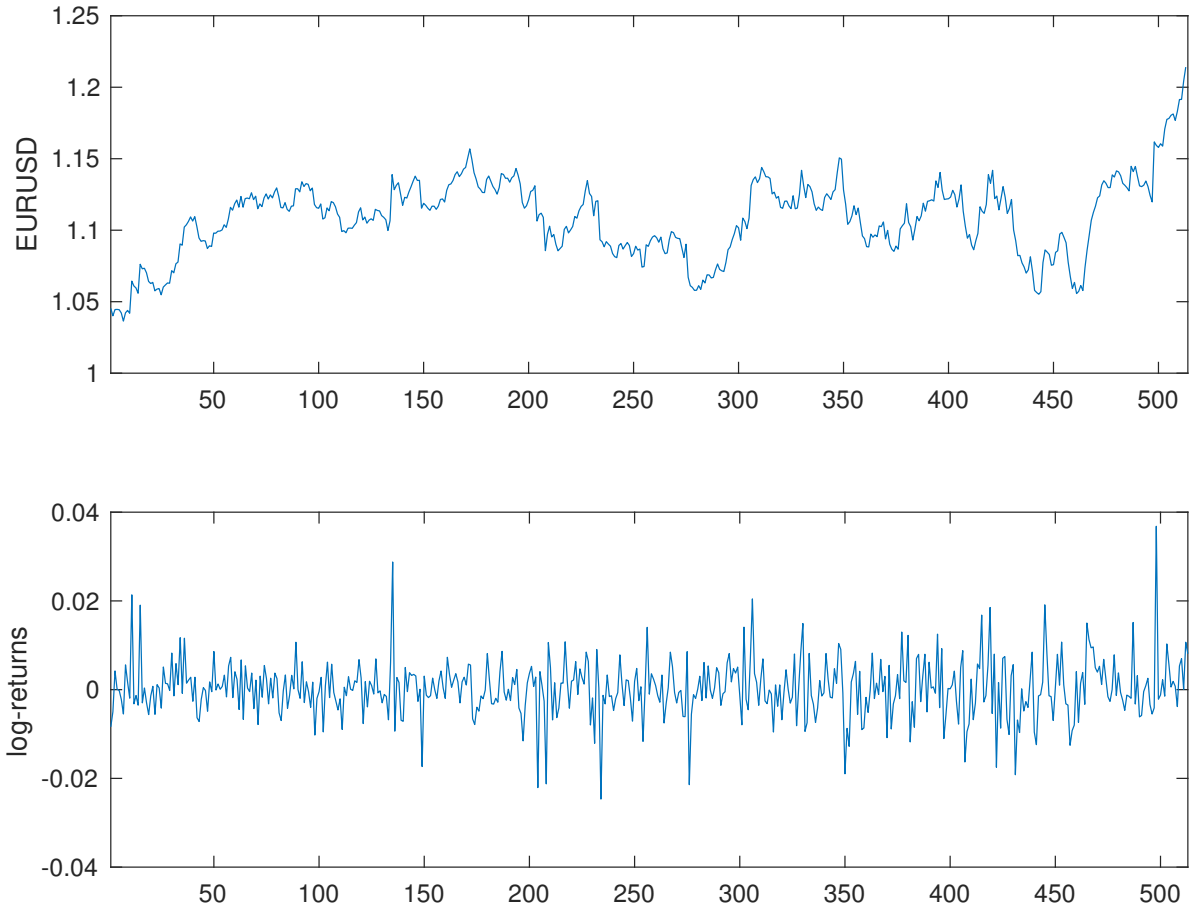
- The sequence $(x_t)_{1 \le t \le T}$ is called as the log-volatility.
- $(y_t)_{1 \le t \le T}$ are observations, which are log-returns.

Given the historical price sequence $s_0, \ldots, s_T$, we set,

$$y_t = 100 \log(s_t / s_{t-1}), \quad \text{for } 1 \le t \le T.$$

Given $y_{1:T}$, we consider inferring the states $x_{1:T}$ and parameters $\theta = (\mu, \sigma_v, \phi)$ jointly using particle Metropolis-Hastings.

Model inference of a stochastic volatility model



Model inference of a stochastic volatility model

Nudging the particle Metropolis-Hastings

Particle Metropolis-Hastings (MH) (Andrieu, Doucet, and Holenstein 2010) is an MH method for sampling from the posterior distribution of parameters of state-space models.

- MH methods simulate a Markov chain from the target distribution, accepting each proposed sample with a certain acceptance probability.

Model inference of a stochastic volatility model

Nudging the particle Metropolis-Hastings

Particle Metropolis-Hastings (MH) (Andrieu, Doucet, and Holenstein 2010) is an MH method for sampling from the posterior distribution of parameters of state-space models.

- MH methods simulate a Markov chain from the target distribution, accepting each proposed sample with a certain acceptance probability.
- Acceptance probabilities computed using the model.

Model inference of a stochastic volatility model

Nudging the particle Metropolis-Hastings

Particle Metropolis-Hastings (MH) (Andrieu, Doucet, and Holenstein 2010) is an MH method for sampling from the posterior distribution of parameters of state-space models.

- MH methods simulate a Markov chain from the target distribution, accepting each proposed sample with a certain acceptance probability.
- Acceptance probabilities computed using the model.
- Particle MH uses an inner BPF routine to estimate acceptance probabilities.

Model inference of a stochastic volatility model

Nudging the particle Metropolis-Hastings

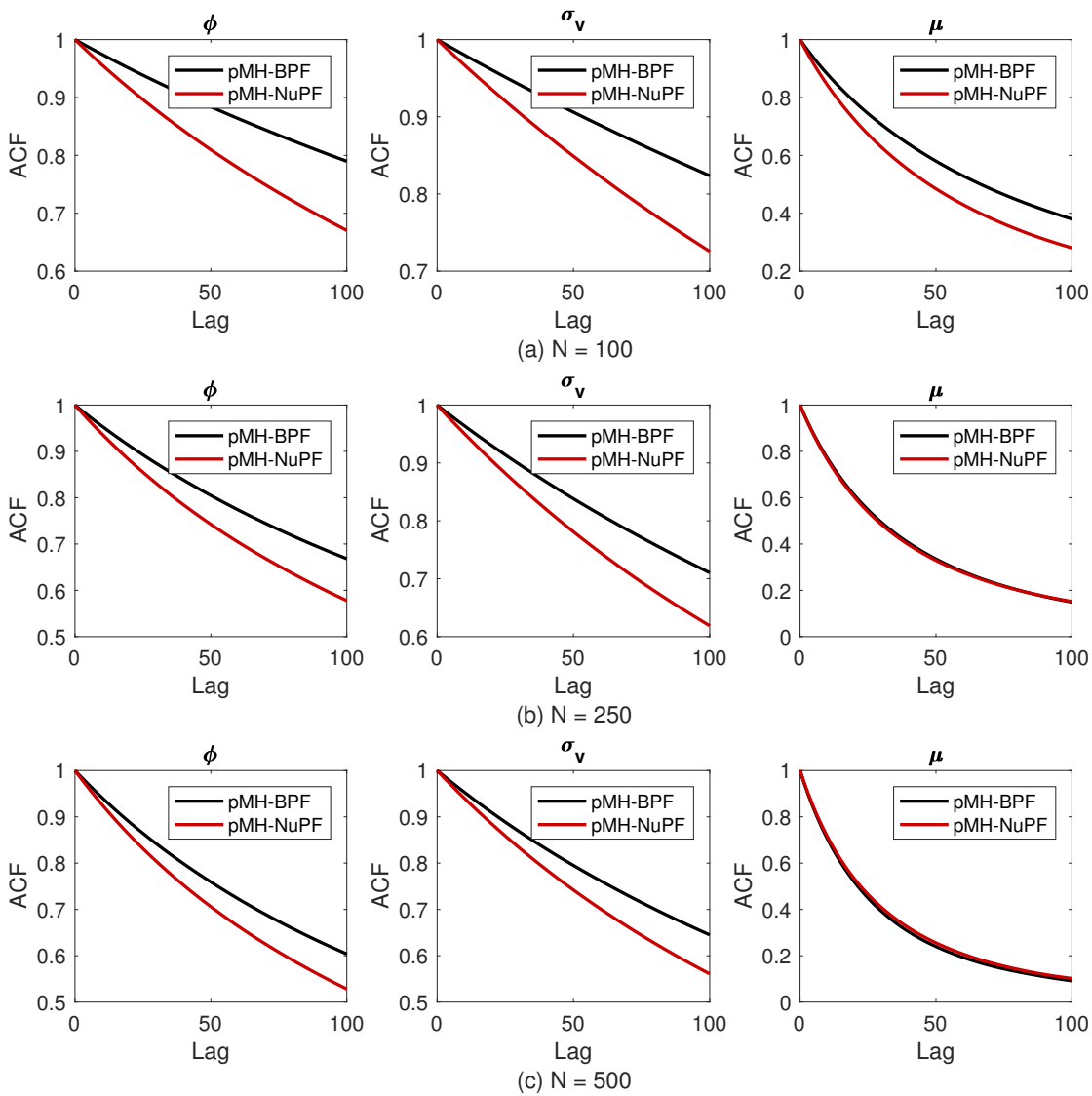
Particle Metropolis-Hastings (MH) (Andrieu, Doucet, and Holenstein 2010) is an MH method for sampling from the posterior distribution of parameters of state-space models.

- MH methods simulate a Markov chain from the target distribution, accepting each proposed sample with a certain acceptance probability.
- Acceptance probabilities computed using the model.
- Particle MH uses an inner BPF routine to estimate acceptance probabilities.

We replace the inner BPF with the NuPF.

Model inference of a stochastic volatility model

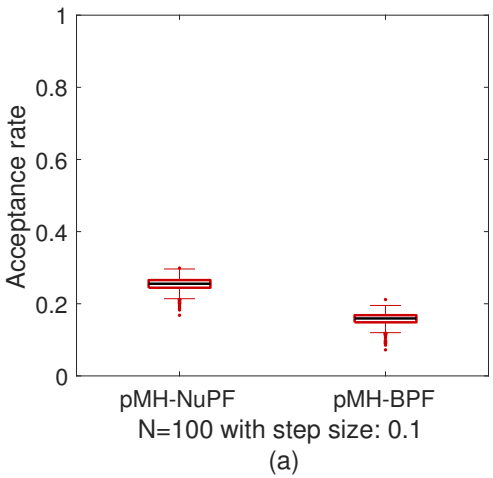
Nudging the particle Metropolis-Hastings

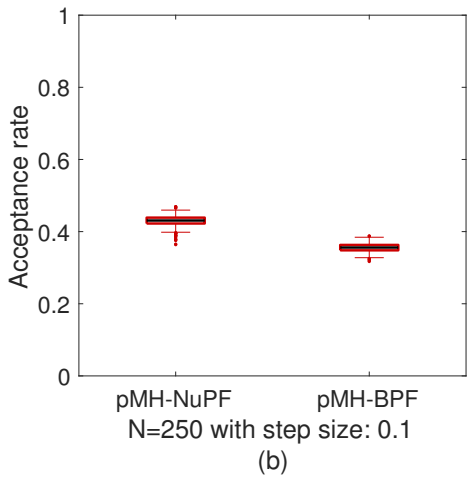


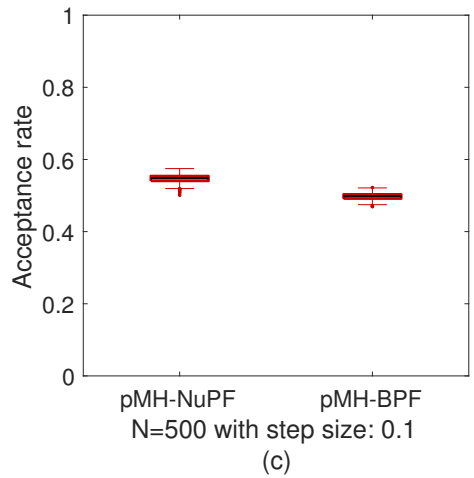
Curves are averaged over 1000 Monte Carlo runs.

Model inference of a stochastic volatility model

Nudging the particle Metropolis-Hastings







Simulations - II

Object tracking

The object of interest is following a model,

$$x_t = Ax_{t-1} + BL(x_{t-1} - x_{\text{target}}) + v_t.$$

- $x_t \in \mathbb{R}^4$ consists of two position and two velocity variables.
- x_{target} is the deterministic, pre-chosen target state.
- The object follows a policy and the policy matrix $L \in \mathbb{R}^{2\times 4}$ is found by solving the Riccati equation.

Simulations - II

Object tracking

The object of interest is following a model,

$$x_t = Ax_{t-1} + BL(x_{t-1} - x_{\text{target}}) + v_t.$$

- $x_t \in \mathbb{R}^4$ consists of two position and two velocity variables.
- x_{target} is the deterministic, pre-chosen target state.
- The object follows a policy and the policy matrix $L \in \mathbb{R}^{2\times 4}$ is found by solving the Riccati equation.

The filters, on the other hand, are not informed about the true model and the policy it follows, they assume,

$$x_t = Ax_{t-1} + v_t.$$

Simulations - II

Object tracking

The observation model,

$$y_{t,i} = 10 \log_{10} \left(\frac{P_0}{\|r_t - s_i\|^2} + \eta \right) + w_{t,i}$$

where

- $r_t = [x_{1,t}, x_{2,t}]^\top$ and s_i is the position of the ith sensor,
- $w_{t,i} \sim \mathcal{MT}(\mu, \Sigma, \nu)$ (Multivariate-t) with $\mu = 0$, $\Sigma = I$, $\nu = 1.01$ (explosive noise!),
- Sensor parameters, we set $\eta = 10^{-9}$ and $P_0 = 1$.

Simulations - II

Object tracking

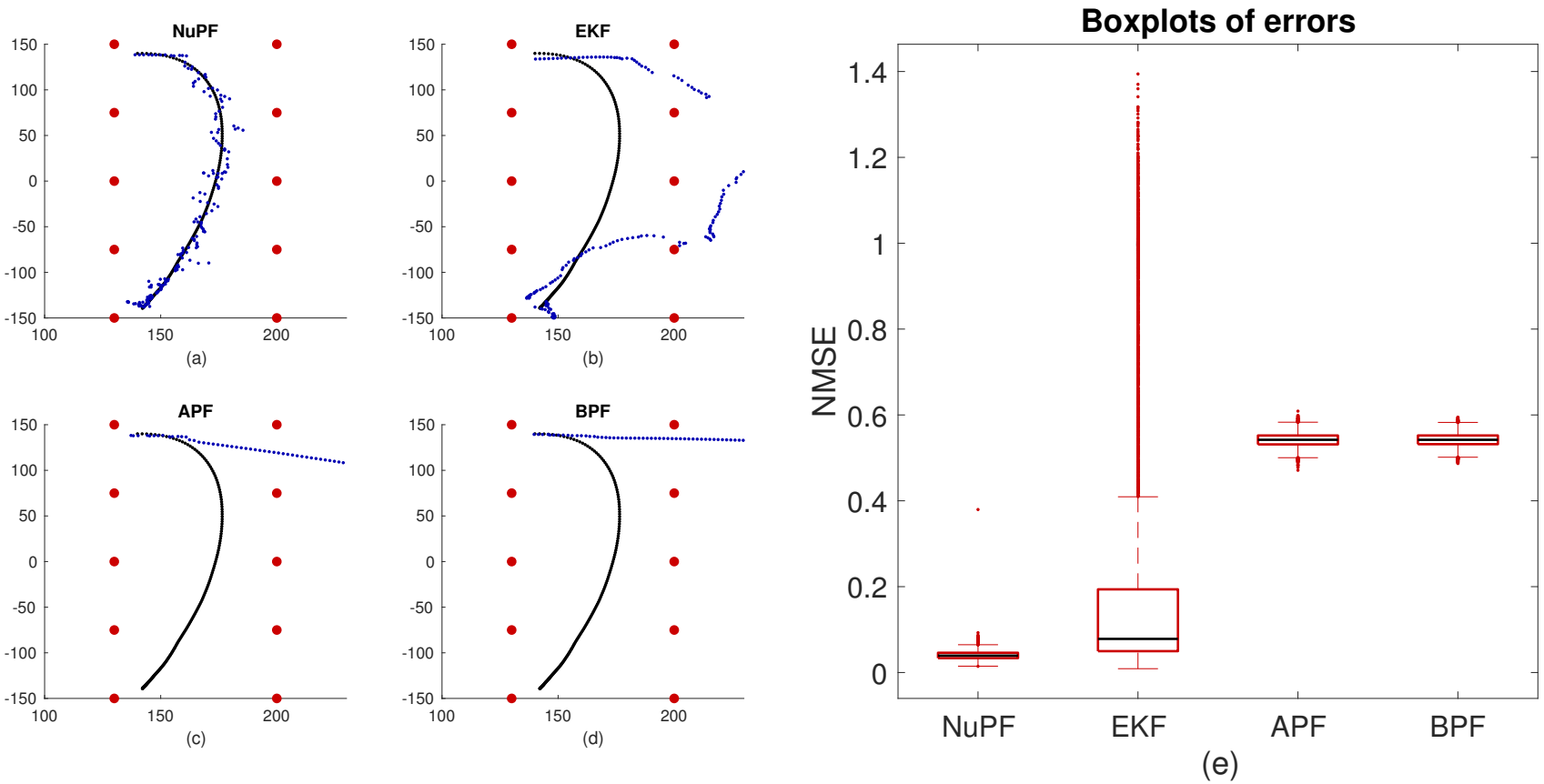


Figure: EKF: Extended Kalman filter, APF: Auxiliary PF. $N = 500$.

Model inference of a stochastic volatility model

Nudging the nested particle filter

Nested particle filter (Crisan and Miguez 2018) is a method for joint inference of parameters and states.

- Two layers of particle filters:

Model inference of a stochastic volatility model

Nudging the nested particle filter

Nested particle filter (Crisan and Miguez 2018) is a method for joint inference of parameters and states.

- Two layers of particle filters:
  - The outer layer is a particle filter on the parameter space.

Model inference of a stochastic volatility model

Nudging the nested particle filter

Nested particle filter (Crisan and Miguez 2018) is a method for joint inference of parameters and states.

- Two layers of particle filters:
  - The outer layer is a particle filter on the parameter space.
  - The inner layer consists of usual bootstrap PFs on the state-space.

Model inference of a stochastic volatility model

Nudging the nested particle filter

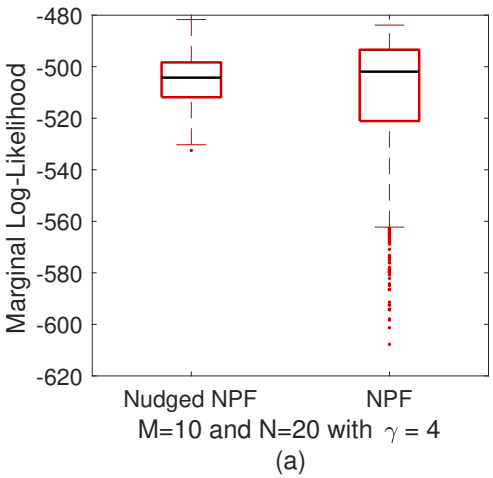
Nested particle filter (Crisan and Miguez 2018) is a method for joint inference of parameters and states.

- Two layers of particle filters:
  - The outer layer is a particle filter on the parameter space.
  - The inner layer consists of usual bootstrap PFs on the state-space.

We replace inner BPFs with NuPFs.

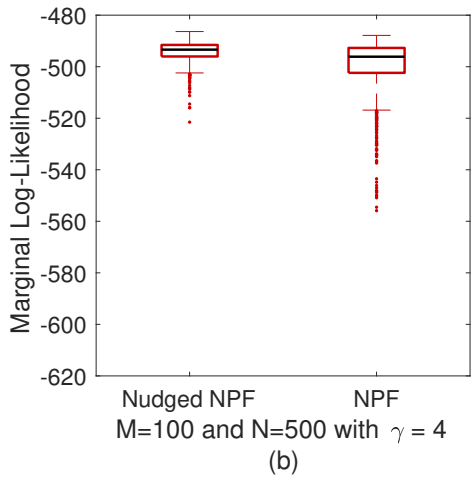
Model inference of a stochastic volatility model

Nudging the nested particle filter



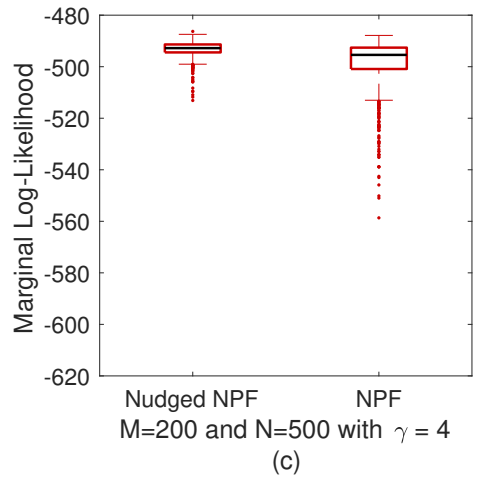
M=10 and N=20 with $\gamma = 4$

(a)



M=100 and N=500 with $\gamma = 4$

(b)



M=200 and N=500 with $\gamma = 4$

(c)

Analysis

a basic L_p result

Theorem 1. Under suitable assumptions on G_t, we obtain

$$\left\|(\varphi, \pi_t) - \left(\varphi, \pi_t^N\right)\right\|_p \leq \frac{c_{t,p}\|\varphi\|_\infty}{\sqrt{N}}$$

for $t = 1, \ldots, T$ and for any $p \geq 1$ where $c_{t,p} > 0$ is a constant independent of N.

Analysis

a basic L_p result

Theorem 1. Under suitable assumptions on G_t, we obtain

$$\left\|(\varphi, \pi_t) - \left(\varphi, \pi_t^N\right)\right\|_p \leq \frac{c_{t,p}\|\varphi\|_\infty}{\sqrt{N}}$$

for $t = 1, \ldots, T$ and for any $p \geq 1$ where $c_{t,p} > 0$ is a constant independent of N.

- However, this result is about expectations w.r.t π_t^N. We want a result about our maximum estimator!

Analysis

A kernel density estimator can be written as an expectation.

Analysis

A kernel density estimator can be written as an expectation.

Let k_h be a kernel with bandwidth h, e.g., a Gaussian kernel. We denote the KDE as

$$\begin{aligned}\mathsf{p}_t^{N,h}(\theta) &= \frac{1}{N}\sum_{i=1}^{N} \mathsf{k}_h(\theta - \theta_t^{(i)}), \\ &= (\mathsf{k}_h^\theta, \pi_t^N),\end{aligned}$$

where $\mathsf{k}_h^\theta(\theta') = \mathsf{k}_h(\theta - \theta')$.

Analysis

A kernel density estimator can be written as an expectation.

Let k_h be a kernel with bandwidth h, e.g., a Gaussian kernel. We denote the KDE as

$$\begin{aligned}\mathsf{p}_t^{N,h}(\theta) &= \frac{1}{N}\sum_{i=1}^{N} \mathsf{k}_h(\theta - \theta_t^{(i)}), \\ &= (\mathsf{k}_h^\theta, \pi_t^N),\end{aligned}$$

where $\mathsf{k}_h^\theta(\theta') = \mathsf{k}_h(\theta - \theta')$.

Then it is possible to use Theorem 1 to analyze kernel density estimators.

Analysis

a uniform convergence result

Theorem 2. Under suitable assumptions on π_t and kernel k, choose

$$h = \left\lfloor N^{\frac{1}{2(d+1)}} \right\rfloor^{-1} \tag{13}$$

and denote $\mathsf{p}_t^N(\theta) = \mathsf{p}_t^{h,N}(\theta)$ since $h = h(N)$. Then

$$\sup_{\theta \in \Theta} |\mathsf{p}_t^N(\theta) - \pi_t(\theta)| \leq \frac{U^\varepsilon}{\left\lfloor N^{\frac{1}{2(d+1)}} \right\rfloor^{1-\varepsilon}} \tag{14}$$

where $U^\varepsilon \geq 0$ is an almost surely finite random variable and $0 < \varepsilon < 1$ is a constant, both of which are independent of N and θ. Then

$$\lim_{N \to \infty} \sup_{\theta \in \Theta} |\mathsf{p}_t^N(\theta) - \pi_t(\theta)| = 0 \qquad \text{a.s.} \tag{15}$$

Analysis

convergence to a global maximum of π_t

Theorem 3. Let $\theta_t^{\star,N} = \mathrm{argmax}_{i\in\{1,\dots,N\}}\, \mathsf{p}_t^N(\theta_t^{(i)})$ be an estimate of a global maximum of π_t. Then, under the assumptions of Theorem 2,

$$\lim_{N\to\infty} \mathsf{p}_t^N(\theta_t^{\star,N}) = \pi_t(\theta_t^\star) \qquad \text{a.s.},$$

where $\theta_t^\star \in \mathrm{argmax}_{\theta\in\Theta}\, \pi_t(\theta)$.

Analysis of the nudged particle filter – model mismatch

What is a better model?

Analysis of the nudged particle filter – model mismatch

What is a better model?

Let us assume we are given two probabilistic models, $\mathcal{M}_0$ and $\mathcal{M}_1$. Given a sequence of observations $y_{1:T}$, we say that the model $\mathcal{M}_1$ is better than $\mathcal{M}_0$ if,

$$\mathsf{p}(y_{1:T}|\mathcal{M}_1) \geq \mathsf{p}(y_{1:T}|\mathcal{M}_0).$$

Analysis of the nudged particle filter – model mismatch

What is a better model?

Let us assume we are given two probabilistic models, $\mathcal{M}_0$ and $\mathcal{M}_1$. Given a sequence of observations $y_{1:T}$, we say that the model $\mathcal{M}_1$ is better than $\mathcal{M}_0$ if,

$$\mathsf{p}(y_{1:T}|\mathcal{M}_1) \geq \mathsf{p}(y_{1:T}|\mathcal{M}_0).$$

For an SSM, say for model $\mathcal{M}_0 = \{\pi_0, \tau_t, g_t\}$,

$$\mathsf{p}(y_{1:T}|\mathcal{M}_0) = \int \cdots \int \prod_{t=1}^{T} g_t^{y_t}(x_t)\tau_t(\mathrm{d}x_t|x_{t-1})\pi_0(\mathrm{d}x_0). \qquad (16)$$

We empirically observe that

$$\mathsf{p}(y_{1:T}|\mathcal{M}_1) \geq \mathsf{p}(y_{1:T}|\mathcal{M}_0)$$

where $\mathcal{M}_0$ is the original, assumed SSM.