# A nested filtering scheme for Bayesian model inference and tracking

Sara Pérez Vieites

Department of Signal Theory & Communications,
Universidad Carlos III de Madrid.
spvieites@tsc.uc3m.es

*Supervisor:* Joaquín Míguez

December 5th, 2017

# Index

# Index

# State-of-the-art Methods

- Very-large-scale stochastic dynamic models of real world phenomena, where we estimate
  - dynamic variables and
  - unknown static parameters.

  $\Rightarrow$ Conventional prediction and estimation methods are not well suited.
- State-of-the-art algorithms:
  1. Bayesian methods (SMC$^2$ [Chopin et al, 2012], pMCMC [Andrieu et al, 2010])
     - approximate the posterior distribution of the unknown variables and parameters, but
     - they are non-recursive.
  2. Maximum likelihood estimators
     - provide point estimates only
     - recursive, but no measure of uncertainty

# Contributions

- A purely recursive generic nested filtering scheme is introduced.
  1. The nested particle filter [Crisan and Miguez, 2016] is a particular case, which infers the posterior distribution of variables and parameters
     - using a bank of particle filters.
     - it is computationally costly.
  2. Therefore, a new class of nested hybrid filters is proposed, using
     - a bank of Gaussian filters such as EKF or EnKF, and
     - reducing running times without significant changes in accuracy.
- A stochastic two-scale Lorenz 96 model is used in computer simulations.

  $\Rightarrow$ It is a benchmark system for meteorology.

# Index

# State-space model

- The dynamical behaviour of the system state can be modeled by sets of nonlinear stochastic differential equations (SDEs), like:

$$d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})dt + dW \tag{1}$$

where $\boldsymbol{f}(\tilde{\boldsymbol{x}})$ is a nonlinear function parametrized by $\boldsymbol{\theta}$ and $dW$ is a brownian process.

# State-space model

- The value of the system state can be approximated applying any sort of discretization method as

$$\tilde{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_{k-1} + h\bar{\boldsymbol{f}}_\sigma(\tilde{\boldsymbol{x}}_{k-1}, \boldsymbol{\theta}, \boldsymbol{v}_k) \tag{2}$$

  where
  - $\bar{\boldsymbol{f}}_\sigma(\tilde{\boldsymbol{x}}_{k-1}, \boldsymbol{\theta}, \boldsymbol{v}_k)$ is an estimate of the vector of time derivatives
  - $h > 0$ is a time-discretisation step with $k = 0, 1, \ldots$ and
  - $\sigma \geq 0$ is a parameter that controls the power of the perturbations, that needs to be small enough to preserve the underlying dynamics of the system

# Observations

States, $\tilde{\boldsymbol{x}}_k$, and unknown parameters, $\boldsymbol{\theta}$, are estimated from a sequence of observation vectors, modelled as

$$\tilde{\boldsymbol{y}}_{kT} = \boldsymbol{g}_{\sigma_o}(\tilde{\boldsymbol{x}}_{kT}, \boldsymbol{\theta}, \tilde{\boldsymbol{u}}_{kT}), \quad k = 1, 2, \ldots, \quad T \geq 1, \tag{3}$$

where

- $\boldsymbol{g}_{\sigma_o} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$, being the observation vectors of dimension $d_y \leq d_x$,
- $T$ is the discrete observation period and
- $\tilde{\boldsymbol{u}}_k$ is a sequence of zero-mean independent vectors representing observational noise, where power is scaled by a known factor $\sigma_o > 0$.

# Dynamical Model

- As we only consider observations every $T$ discrete time steps, it is convenient to rewrite the dynamic model in the same time scale. We work with the pair of random sequences $\boldsymbol{x}_n := \tilde{\boldsymbol{x}}_{nT}$ and $\boldsymbol{y}_n := \tilde{\boldsymbol{y}}_{nT}$, as

$$\boldsymbol{y}_n = \boldsymbol{g}_{\sigma_o}(\boldsymbol{x}_n, \boldsymbol{\theta}, \boldsymbol{u}_n), \tag{4}$$

$$\boldsymbol{x}_n = \bar{\boldsymbol{F}}_{T,\sigma}(\boldsymbol{x}_{n-1}, \boldsymbol{\theta}, \boldsymbol{v}_n), \quad n = 1, 2, \ldots \tag{5}$$

- Here $\bar{\boldsymbol{F}}$ represents the transformation from $\boldsymbol{x}_{n-1} := \tilde{\boldsymbol{x}}_{(n-1)T}$ to $\boldsymbol{x}_n := \tilde{\boldsymbol{x}}_{nT}$ in $T$ steps as

$$\begin{aligned}
\tilde{\boldsymbol{x}}_{(n-1)T} &= \boldsymbol{x}_{n-1} \\
\tilde{\boldsymbol{x}}_{(n-1)T+1} &= \tilde{\boldsymbol{x}}_{(n-1)T} + \bar{\boldsymbol{f}}_{\sigma}(\tilde{\boldsymbol{x}}_{(n-1)T}, \boldsymbol{\theta}, \tilde{\boldsymbol{v}}_{(n-1)T+1}) \\
&\vdots \\
\tilde{\boldsymbol{x}}_{nT-1} &= \tilde{\boldsymbol{x}}_{nT-2} + \bar{\boldsymbol{f}}_{\sigma}(\tilde{\boldsymbol{x}}_{nT-2}, \boldsymbol{\theta}, \tilde{\boldsymbol{v}}_{nT-1}) \\
\boldsymbol{x}_n &= \tilde{\boldsymbol{x}}_{nT-1} + \bar{\boldsymbol{f}}_{\sigma}(\tilde{\boldsymbol{x}}_{nT-1}, \boldsymbol{\theta}, \tilde{\boldsymbol{v}}_{nT})
\end{aligned} \tag{6}$$

# Objective

- The aim of this work is to estimate the parameters $\boldsymbol{\theta}$ and its posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$.
- We combine Monte Carlo with other auxiliary filters for that purpose.

# Nested filtering

- The following (naive) algorithm yields a weighted Monte Carlo approximation of $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n})d\boldsymbol{\theta}$ at each time $n$:
  1. Draw $N$ i.i.d. samples $\boldsymbol{\theta}_n^i$, $i = 1, 2, \ldots, N$, from $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n-1})$.
  2. Compute importance weights,

$$\tilde{w}_n^i = p(\boldsymbol{y}_n|\boldsymbol{\theta}_n^i, \boldsymbol{y}_{1:n-1}), \quad i = 1, \ldots, N, \tag{7}$$

     and normalise them. We obtain the IS estimate $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n})d\boldsymbol{\theta} = \sum_{i=1}^N w_n^i \delta_{\boldsymbol{\theta}_n^i}(d\boldsymbol{\theta})$.

- This weighted Monte Carlo approximation is not practical because
  - it is not possible to draw from $p(\theta|\boldsymbol{y}_{1:n})$, at least exactly, and
  - the likelihood $u_n(\theta_n^i) := p(\boldsymbol{y}_n|\boldsymbol{\theta}_n^i, \boldsymbol{y}_{1:n-1})$ cannot be evaluated exactly either.
- We can obtain an estimate of $u_n(\theta)$ if we approximate first the predictive measure $p(\boldsymbol{x}_n|\theta, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$ since it can be rewritten as

$$u_n(\theta) = \int p(\boldsymbol{y}_n|\boldsymbol{x}_n, \theta)p(\boldsymbol{x}_n|\theta, \boldsymbol{y}_{1:n-1})d(\boldsymbol{x}_n) \tag{8}$$

# Nested filtering

- The following (naive) algorithm yields a weighted Monte Carlo approximation of $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n})d\boldsymbol{\theta}$ at each time $n$:
  1. Draw $N$ i.i.d. samples $\boldsymbol{\theta}_n^i$, $i = 1, 2, \ldots, N$, from $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n-1})$.
  2. Compute importance weights,

$$\tilde{w}_n^i = p(\boldsymbol{y}_n|\boldsymbol{\theta}_n^i, \boldsymbol{y}_{1:n-1}), \quad i = 1, \ldots, N, \tag{7}$$

     and normalise them. We obtain the IS estimate $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n})d\boldsymbol{\theta} = \sum_{i=1}^N w_n^i \delta_{\boldsymbol{\theta}_n^i}(d\boldsymbol{\theta})$.
- This weighted Monte Carlo approximation is not practical because
  - it is not possible to draw from $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n})$, at least exactly, and
  - the likelihood $u_n(\boldsymbol{\theta}_n^i) := p(\boldsymbol{y}_n|\boldsymbol{\theta}_n^i, \boldsymbol{y}_{1:n-1})$ cannot be evaluated exactly either.
- We can obtain an estimate of $u_n(\theta)$ if we approximate first the predictive measure $p(x_n|\theta, y_{1:n-1})dx_n$ since it can be rewritten as

$$u_n(\theta) = \int p(y_n|x_n, \theta)p(x_n|\theta, y_{1:n-1})d(x_n) \tag{8}$$

# Nested filtering

- The following (naive) algorithm yields a weighted Monte Carlo approximation of $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n})d\boldsymbol{\theta}$ at each time $n$:
  1. Draw $N$ i.i.d. samples $\boldsymbol{\theta}_n^i$, $i = 1, 2, \ldots, N$, from $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n-1})$.
  2. Compute importance weights,

$$\tilde{w}_n^i = p(\boldsymbol{y}_n|\boldsymbol{\theta}_n^i, \boldsymbol{y}_{1:n-1}), \quad i = 1, \ldots, N, \tag{7}$$

     and normalise them. We obtain the IS estimate $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n})d\boldsymbol{\theta} = \sum_{i=1}^N w_n^i \delta_{\boldsymbol{\theta}_n^i}(d\boldsymbol{\theta})$.
- This weighted Monte Carlo approximation is not practical because
  - it is not possible to draw from $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n})$, at least exactly, and
  - the likelihood $u_n(\boldsymbol{\theta}_n^i) := p(\boldsymbol{y}_n|\boldsymbol{\theta}_n^i, \boldsymbol{y}_{1:n-1})$ cannot be evaluated exactly either.
- We can obtain an estimate of $u_n(\boldsymbol{\theta})$ if we approximate first the predictive measure $p(\boldsymbol{x}_n|\boldsymbol{\theta}, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$ since it can be rewritten as

$$u_n(\boldsymbol{\theta}) = \int p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}) p(\boldsymbol{x}_n|\boldsymbol{\theta}, \boldsymbol{y}_{1:n-1}) d(\boldsymbol{x}_n) \tag{8}$$

# General Nested Filter

## Algorithm

1. **Initialisation**

   *Draw* $\boldsymbol{\theta}_0^{(i)}, i = 1, \ldots, N$, *i.i.d. samples from* $p(\boldsymbol{\theta})d\boldsymbol{\theta}$.

2. **Recursive step**

   2.1 *For* $i = 1, \ldots, N$:

   2.1.1 *Draw* $\bar{\boldsymbol{\theta}}_n^{(i)}$ *from* $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{n-1}^i)$.

   2.1.2 *Approximate* $p(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$.

   2.1.3 *Use* $p(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$ *to compute* $\hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *and let* $w_n^i \propto \hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *be the normalised weight of* $\bar{\boldsymbol{\theta}}_n^i$.

   2.2 *Resample to obtain the set* $\{\boldsymbol{\theta}_{i=1}^N\}$ *and the approximation* $\hat{p}(d\boldsymbol{\theta}|\boldsymbol{y}_{1:n}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_n^i}(d\boldsymbol{\theta})$.

# General Nested Filter

## Algorithm

1. **Initialisation**

   *Draw* $\boldsymbol{\theta}_0^{(i)}, i = 1, \ldots, N$, *i.i.d. samples from* $p(\boldsymbol{\theta})d\boldsymbol{\theta}$.

2. **Recursive step**

   2.1 *For* $i = 1, \ldots, N$:

   2.1.1 *Draw* $\bar{\boldsymbol{\theta}}_n^{(i)}$ *from* $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{n-1}^i)$.

   2.1.2 *Approximate* $p(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$.

   2.1.3 *Use* $p(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$ *to compute* $\hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *and let* $w_n^i \propto \hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *be the normalised weight of* $\bar{\boldsymbol{\theta}}_n^i$.

   2.2 *Resample to obtain the set* $\{\boldsymbol{\theta}_{i=1}^N\}$ *and the approximation* $\hat{p}(d\boldsymbol{\theta}|\boldsymbol{y}_{1:n}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_n^i}(d\boldsymbol{\theta})$.

Sampling $\bar{\boldsymbol{\theta}}_n^{(i)}$

# General Nested Filter

## Algorithm

1. **Initialisation**

   *Draw* $\boldsymbol{\theta}_0^{(i)}, i = 1, \ldots, N$, *i.i.d. samples from* $p(\boldsymbol{\theta})d\boldsymbol{\theta}$.

2. **Recursive step**

   2.1 *For* $i = 1, \ldots, N$:

   2.1.1 *Draw* $\bar{\boldsymbol{\theta}}_n^{(i)}$ *from* $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{n-1}^i)$.

   2.1.2 *Approximate* $p(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$.

   2.1.3 *Use* $p(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$ *to compute* $\hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *and let* $w_n^i \propto \hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *be the normalised weight of* $\bar{\boldsymbol{\theta}}_n^i$.

   2.2 *Resample to obtain the set* $\{\boldsymbol{\theta}_{i=1}^N\}$ *and the approximation* $\hat{p}(d\boldsymbol{\theta}|\boldsymbol{y}_{1:n}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_n^i}(d\boldsymbol{\theta})$.

Sampling $\bar{\boldsymbol{\theta}}_n^{(i)}$

↓ Prediction

$\hat{p}(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$

# General Nested Filter

## Algorithm

1. **Initialisation**

   *Draw* $\boldsymbol{\theta}_0^{(i)}, i = 1, \ldots, N$, *i.i.d. samples from* $p(\boldsymbol{\theta})d\boldsymbol{\theta}$.

2. **Recursive step**

   2.1 *For* $i = 1, \ldots, N$:

   2.1.1 *Draw* $\bar{\boldsymbol{\theta}}_n^{(i)}$ *from* $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{n-1}^i)$.

   2.1.2 *Approximate* $p(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$.

   2.1.3 *Use* $p(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$ *to compute* $\hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *and let* $w_n^i \propto \hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *be the normalised weight of* $\bar{\boldsymbol{\theta}}_n^i$.

   2.2 *Resample to obtain the set* $\{\boldsymbol{\theta}_{i=1}^N\}$ *and the approximation* $\hat{p}(d\boldsymbol{\theta}|\boldsymbol{y}_{1:n}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_n^i}(d\boldsymbol{\theta})$.

Sampling $\bar{\boldsymbol{\theta}}_n^{(i)}$

$\downarrow$ Prediction

$\hat{p}(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n \longrightarrow \hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$

$\downarrow$ Update

$\hat{p}(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n})d\boldsymbol{x}_n$

# General Nested Filter

## Algorithm

1. **Initialisation**

   *Draw* $\boldsymbol{\theta}_0^{(i)}, i = 1, \ldots, N$, *i.i.d. samples from* $p(\boldsymbol{\theta})d\boldsymbol{\theta}$.

2. **Recursive step**

   2.1 *For* $i = 1, \ldots, N$*:*

   2.1.1 *Draw* $\bar{\boldsymbol{\theta}}_n^{(i)}$ *from* $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{n-1}^i)$.

   2.1.2 *Approximate* $p(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$.

   2.1.3 *Use* $p(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n$ *to compute* $\hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *and let* $w_n^i \propto \hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *be the normalised weight of* $\bar{\boldsymbol{\theta}}_n^i$.

   2.2 *Resample to obtain the set* $\{\boldsymbol{\theta}_{i=1}^N\}$ *and the approximation* $\hat{p}(d\boldsymbol{\theta}|\boldsymbol{y}_{1:n}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_n^i}(d\boldsymbol{\theta})$.

Sampling $\bar{\boldsymbol{\theta}}_n^{(i)}$

↓ Prediction

$\hat{p}(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n \longrightarrow \hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$

↓ Update

$\hat{p}(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n})d\boldsymbol{x}_n$

↓ Resampling

$\hat{p}(\boldsymbol{x}_n|\boldsymbol{\theta}_n^i, \boldsymbol{y}_{1:n})d\boldsymbol{x}_n$ and $\hat{p}(d\boldsymbol{\theta}|\boldsymbol{y})$

# NHF via EKF

Let us assume $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0|\bar{\boldsymbol{x}}_0, \bar{\boldsymbol{P}}_0)$, $\boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{v}_k|\boldsymbol{0}, \boldsymbol{Q})$ and $\boldsymbol{u}_k \sim \mathcal{N}(\boldsymbol{u}_k|\boldsymbol{0}, \boldsymbol{R})$. Functions $\bar{\boldsymbol{f}}$ in Eq. (2) and $\boldsymbol{g}$ in Eq. (4) are nonlinear and differentiable, with $\boldsymbol{J}_{\bar{\boldsymbol{f}},\boldsymbol{x},\boldsymbol{\theta}}$ and $\boldsymbol{J}_{\boldsymbol{g},\boldsymbol{x},\boldsymbol{\theta}}$ denoting their respective Jacobian matrices evaluated at the point $\boldsymbol{x}$ and $\boldsymbol{\theta}$.

Sampling $\bar{\boldsymbol{\theta}}_n^{(i)}$

$\downarrow$ Prediction

$\hat{p}(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n \longrightarrow \hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$

$\downarrow$ Update

$\hat{p}(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n})d\boldsymbol{x}_n$

$\downarrow$ Resampling

$\hat{p}(\boldsymbol{x}_n|\boldsymbol{\theta}_n^i, \boldsymbol{y}_{1:n})d\boldsymbol{x}_n$ and $\hat{p}(d\boldsymbol{\theta}|\boldsymbol{y})$

## Algorithm

1. *Initialisation:* draw $N$ i.i.d. particles $\boldsymbol{\theta}_0^i \sim p(\boldsymbol{\theta})d\boldsymbol{\theta}, i = 1, \ldots, N$. Let $\bar{\boldsymbol{x}}_0^i = \bar{\boldsymbol{x}}_0$ and $\bar{\boldsymbol{P}}_0^i = \boldsymbol{P}_0$ for every $i$.
2. *Recursive step:* at time $n$, we have available $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n-1})d\boldsymbol{\theta} \approx \frac{1}{N}\sum_{i=1}^N \delta_{\bar{\boldsymbol{\theta}}_{n-1}^i}(d\boldsymbol{\theta})$ and, for each $i = 1, \ldots, N$, $p(\boldsymbol{x}_n|\boldsymbol{\theta}_{n-1}^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_{n-1} \approx \mathcal{N}(\boldsymbol{x}_{n-1}|\bar{\boldsymbol{x}}_{n-1}^i, \bar{\boldsymbol{P}}_{n-1}^i)d\boldsymbol{x}_n$.

# NHF via EKF

Let us assume $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0|\bar{\boldsymbol{x}}_0, \bar{\boldsymbol{P}}_0)$, $\boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{v}_k|\boldsymbol{0}, \boldsymbol{Q})$ and $\boldsymbol{u}_k \sim \mathcal{N}(\boldsymbol{u}_k|\boldsymbol{0}, \boldsymbol{R})$. Functions $\bar{\boldsymbol{f}}$ in Eq. (2) and $\boldsymbol{g}$ in Eq. (4) are nonlinear and differentiable, with $\boldsymbol{J}_{\bar{\boldsymbol{f}},\boldsymbol{x},\boldsymbol{\theta}}$ and $\boldsymbol{J}_{\boldsymbol{g},\boldsymbol{x},\boldsymbol{\theta}}$ denoting their respective Jacobian matrices evaluated at the point $\boldsymbol{x}$ and $\boldsymbol{\theta}$.

Sampling $\bar{\boldsymbol{\theta}}_n^{(i)}$

$\downarrow$ Prediction

$\hat{p}(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n \longrightarrow \hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$

$\downarrow$ Update

$\hat{p}(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n})d\boldsymbol{x}_n$

$\downarrow$ Resampling

$\hat{p}(\boldsymbol{x}_n|\boldsymbol{\theta}_n^i, \boldsymbol{y}_{1:n})d\boldsymbol{x}_n$ and $\hat{p}(d\boldsymbol{\theta}|\boldsymbol{y})$

## Algorithm

1. *Initialisation: draw $N$ i.i.d. particles $\boldsymbol{\theta}_0^i \sim p(\boldsymbol{\theta})d\boldsymbol{\theta}, i = 1, \ldots, N$. Let $\bar{\boldsymbol{x}}_0^i = \bar{\boldsymbol{x}}_0$ and $\bar{\boldsymbol{P}}_0^i = \boldsymbol{P}_0$ for every $i$.*
2. *Recursive step: at time $n$, we have available $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n-1})d\boldsymbol{\theta} \approx \frac{1}{N}\sum_{i=1}^N \delta_{\bar{\boldsymbol{\theta}}_{n-1}^i}(d\boldsymbol{\theta})$ and, for each $i = 1, \ldots, N$, $p(\boldsymbol{x}_n|\boldsymbol{\theta}_{n-1}^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_{n-1} \approx \mathcal{N}(\boldsymbol{x}_{n-1}|\bar{\boldsymbol{x}}_{n-1}^i, \bar{\boldsymbol{P}}_{n-1}^i)d\boldsymbol{x}_n$.*

Sampling $\bar{\theta}_n^{(i)}$

↓ Prediction

$\hat{p}(x_n|\bar{\theta}_n^i, y_{1:n-1})dx_n \longrightarrow \hat{u}_n(\bar{\theta}_n^i)$

↓ Update

$\hat{p}(x_n|\bar{\theta}_n^i, y_{1:n})dx_n$

↓ Resampling

$\hat{p}(x_n|\theta_n^i, y_{1:n})dx_n$ and $\hat{p}(d\theta|y)$

(a) *Prediction:*

i *Draw* $\bar{\theta}_n^i \sim \kappa_N(d\theta|\theta_{n-1}^i)$, $i = 1, \ldots, N$.

ii *Let* $\breve{x}_0^i = \bar{x}_{n-1}^i$ *and* $\breve{P}_0^i = \bar{P}_{n-1}^i$, $i = 1, \ldots, N$. *Then, for each* $i$ *and* $k = 1, \ldots, T$ *compute*

$$\breve{x}_k^i = \bar{f}_\sigma(\breve{x}_{k-1}^i, \bar{\theta}_n^i, \mathbf{0}), \tag{9}$$

$$\breve{P}_k^i = J_{\bar{f}, \breve{x}_{k-1}^i, \bar{\theta}_n^i} \breve{P}_{k-1}^i J_{\bar{f}, \breve{x}_{k-1}^i, \bar{\theta}_n^i}^\top + \sigma^2 Q \tag{10}$$

iii *Set* $p(x_n|\bar{\theta}_n^i, y_{1:n-1})dx_n = \mathcal{N}(x_n|\hat{x}_n^i, \hat{P}_n^i)dx_n$ *where* $\hat{x}_n^i = \breve{x}_T^i$ *and* $\hat{P}_n^i = \hat{P}_T^i$.

Sampling $\bar{\theta}_n^{(i)}$

$\downarrow$ Prediction

$\hat{p}(x_n|\bar{\theta}_n^i, y_{1:n-1})dx_n \longrightarrow \hat{u}_n(\bar{\theta}_n^i)$

$\downarrow$ Update

$\hat{p}(x_n|\bar{\theta}_n^i, y_{1:n})dx_n$

$\downarrow$ Resampling

$\hat{p}(x_n|\theta_n^i, y_{1:n})dx_n$ and $\hat{p}(d\theta|y)$

(b) *Update:*

i *For* $i = 1, \ldots, N$, *compute*

$$S_n^i = J_{g,\hat{x}_n^i,\bar{\theta}_n^i} \hat{P}_n^i J_{g,\hat{x}_n^i,\bar{\theta}_n^i}^\top + \sigma_o^2 R \quad (11)$$

$$K_n^i = \hat{P}_n^i J_{g,\hat{x}_n^i,\bar{\theta}_n^i}^\top (S_n^i)^{-1} \quad (12)$$

$$\check{x}_n^i = \hat{x}_n^i + K_n^i (y_n - g(\hat{x}_n^i, \bar{\theta}_n^i)) \quad (13)$$

$$\check{P}_n^i = (I_{d_x} - K_n^i J_{g,\hat{x}_n^i,\bar{\theta}_n^i}) \hat{P}_n^i \quad (14)$$

ii *Compute* $\hat{u}(\bar{\theta}_n^i) = \mathcal{N}(y_n|g(\hat{x}_n^i, \bar{\theta}_n^i), S_n^i)$ *and obtain the normalised weights.*

iii *Set the filter approximation* $p(x_n|\bar{\theta}_n^i, y_{1:n})dx_n = \mathcal{N}(x_n|\check{x}_n^i, \check{P}_n^i)dx_n$.

Sampling $\bar{\theta}_n^{(i)}$

$\downarrow$ Prediction

$\hat{p}(x_n|\bar{\theta}_n^i, y_{1:n-1})dx_n \longrightarrow \hat{u}_n(\bar{\theta}_n^i)$

$\downarrow$ Update

$\hat{p}(x_n|\bar{\theta}_n^i, y_{1:n})dx_n$

$\downarrow$ Resampling

$\hat{p}(x_n|\theta_n^i, y_{1:n})dx_n$ and $\hat{p}(d\theta|y)$

(b) *Update:*

i *For* $i = 1, \ldots, N$, *compute*

$$S_n^i = J_{g,\hat{x}_n^i,\bar{\theta}_n^i} \hat{P}_n^i J_{g,\hat{x}_n^i,\bar{\theta}_n^i}^\top + \sigma_o^2 R \quad (11)$$

$$K_n^i = \hat{P}_n^i J_{g,\hat{x}_n^i,\bar{\theta}_n^i}^\top (S_n^i)^{-1} \quad (12)$$

$$\check{x}_n^i = \hat{x}_n^i + K_n^i (y_n - g(\hat{x}_n^i, \bar{\theta}_n^i)) \quad (13)$$

$$\check{P}_n^i = (I_{d_x} - K_n^i J_{g,\hat{x}_n^i,\bar{\theta}_n^i}) \hat{P}_n^i \quad (14)$$

ii *Compute* $\hat{u}(\bar{\theta}_n^i) = \mathcal{N}(y_n|g(\hat{x}_n^i, \bar{\theta}_n^i), S_n^i)$ *and obtain the normalised weights.*

iii *Set the filter approximation* $p(x_n|\bar{\theta}_n^i, y_{1:n})dx_n = \mathcal{N}(x_n|\check{x}_n^i, \check{P}_n^i)dx_n$.

Sampling $\bar{\boldsymbol{\theta}}_n^{(i)}$

$\downarrow$ Prediction

$\hat{p}(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n \longrightarrow \hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$

$\downarrow$ Update

$\hat{p}(\boldsymbol{x}_n|\bar{\boldsymbol{\theta}}_n^i, \boldsymbol{y}_{1:n})d\boldsymbol{x}_n$

$\downarrow$ Resampling

$\hat{p}(\boldsymbol{x}_n|\boldsymbol{\theta}_n^i, \boldsymbol{y}_{1:n})d\boldsymbol{x}_n$ and $\hat{p}(d\boldsymbol{\theta}|\boldsymbol{y})$

(c) *Resampling: draw indices $j_1, \ldots, j_N$ from the multinomial distribution with probabilities $w_n^1, \ldots, w_n^N$, then set*

$$\boldsymbol{\theta}_n^i = \bar{\boldsymbol{\theta}}_n^{j_i}, \quad \bar{\boldsymbol{x}}_n^i = \check{\boldsymbol{x}}_n^{j_i} \quad \text{and} \quad \bar{\boldsymbol{P}}_n^i = \check{\boldsymbol{P}}_n^{j_i} \tag{15}$$

*for $i = 1, \ldots, N$. Hence, $p(\boldsymbol{x}_n|\boldsymbol{\theta}_n^i, \boldsymbol{y}_{1:n})d\boldsymbol{x}_n$ $= \mathcal{N}(\boldsymbol{x}_n|\bar{\boldsymbol{x}}_n^i, \bar{\boldsymbol{P}}_n^i)d\boldsymbol{x}_n$.*

# Computation of point estimates

- The posterior-mean estimators of $\boldsymbol{\theta}$ and $\boldsymbol{x}$ become

$$\hat{\boldsymbol{\theta}}_n = \int \boldsymbol{\theta} p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n}) d\boldsymbol{\theta} \approx \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\theta}_n^i = \hat{\boldsymbol{\theta}}_n^N,$$

$$\boldsymbol{x}_n = \int \boldsymbol{x} p(\boldsymbol{x}|\boldsymbol{y}_{1:n}, \boldsymbol{\theta}) d\boldsymbol{x}_n \approx \frac{1}{N} \sum_{i=1}^{N} \bar{\boldsymbol{x}}_n^i = \boldsymbol{x}_n^N.$$

- The mean square error (MSE) of $\hat{\boldsymbol{\theta}}_n$ is calculated as

$$MSE_n = \int \| \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_n \|^2 p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n}) d\boldsymbol{\theta}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \| \boldsymbol{\theta}_n^i - \hat{\boldsymbol{\theta}}_n^N \|^2$$

# Index

# Preliminaries and notation

For the integral of a function $a(\boldsymbol{\theta})$ w.r.t. a measure $\alpha$, hereafter we use the shorthand

$$(a, \alpha) := \int a(\boldsymbol{\theta})\alpha(d\boldsymbol{\theta}).$$

The sequence of posterior probability measures of the unknown parameters, $\mu_n(d\boldsymbol{\theta}) := p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n})d\boldsymbol{\theta}$, $n \geq 1$, can be constructed recursively starting from a prior $\mu_0$ as

$$\mu_n = u_n \star \mu_{n-1} \quad \text{where} \quad (f, u_n \star \alpha) = \frac{(fu_n, \alpha)}{(u_n, \alpha)}.$$

If, instead of the true likelihood $u_n$, we use another function $\bar{u}_n \neq u_n$ to update the posterior probability measure then we obtain the new sequence of measures

$$\bar{\mu}_0 = \mu_0, \quad \bar{\mu}_n = \bar{u}_n \star \bar{\mu}_{n-1}, \quad n = 1, 2, \ldots$$

# Convergence Theorem

A.1. The estimator $\hat{u}_n(\boldsymbol{\theta})$ is random and can be written as

$$\hat{u}_n(\boldsymbol{\theta}) = \bar{u}_n(\boldsymbol{\theta}) + m_n(\boldsymbol{\theta}),$$

where $m_n(\boldsymbol{\theta})$ is a zero-mean random variable with finite variance. Furthermore, the mean $\bar{u}_n(\boldsymbol{\theta}) = \mathrm{E}\left[\hat{u}_n(\boldsymbol{\theta})\right]$ has the form

$$\bar{u}_n(\boldsymbol{\theta}) = u_n(\boldsymbol{\theta}) + b_n(\boldsymbol{\theta}),$$

where $b_n(\boldsymbol{\theta})$ is a deterministic and bounded bias function.

## Theorem 1

Let the sequence of observations $y_{1:n_o}$ be arbitrary but fixed, with $n_o < \infty$, and choose an arbitrary function $h \in B(D)$. Let $\mu_n^N = \frac{1}{N}\sum_{i=1}^N \delta_{\theta_n^i}$ be the random probability measure in the parameter space generated by the nested filter. If A.1 holds and under regularity conditions, then

$$\left\|(h, \mu_n^N) - (h, \bar{\mu}_n)\right\|_p \leq \frac{c_n \|h\|_\infty}{\sqrt{N}}, \quad \text{for } n = 0, 1, \ldots, n_o,$$

where $\{c_n\}_{0 \leq n \leq n_o}$ is a sequence of constants independent of $N$. □

# Convergence Theorem

A.1. The estimator $\hat{u}_n(\boldsymbol{\theta})$ is random and can be written as

$$\hat{u}_n(\boldsymbol{\theta}) = \bar{u}_n(\boldsymbol{\theta}) + m_n(\boldsymbol{\theta}),$$

where $m_n(\boldsymbol{\theta})$ is a zero-mean random variable with finite variance. Furthermore, the mean $\bar{u}_n(\boldsymbol{\theta}) = \mathrm{E}\left[\hat{u}_n(\boldsymbol{\theta})\right]$ has the form

$$\bar{u}_n(\boldsymbol{\theta}) = u_n(\boldsymbol{\theta}) + b_n(\boldsymbol{\theta}),$$

where $b_n(\boldsymbol{\theta})$ is a deterministic and bounded bias function.

## Theorem 1

Let the sequence of observations $y_{1:n_o}$ be arbitrary but fixed, with $n_o < \infty$, and choose an arbitrary function $h \in B(D)$. Let $\mu_n^N = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_n^i}$ be the random probability measure in the parameter space generated by the nested filter. If A.1 holds and under regularity conditions, then

$$\left\|(h, \mu_n^N) - (h, \bar{\mu}_n)\right\|_p \leq \frac{c_n \|h\|_\infty}{\sqrt{N}}, \quad \text{for } n = 0, 1, \ldots, n_o,$$

where $\{c_n\}_{0 \leq n \leq n_o}$ is a sequence of constants independent of $N$. $\square$

# Index

# A Stochastic Lorenz 96 Model

- The model consists of two sets of dynamic variables, $\boldsymbol{x}(t)$ and $\boldsymbol{z}(t)$, that displays some key features of atmosphere dynamics (including chaotic behaviour) in a relatively simple model of arbitrary dimension. The system of differential equations takes the form

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}_1(\boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{\alpha})dt + dW \\ \dot{\boldsymbol{z}}(t) &= \boldsymbol{f}_2(\boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{\alpha})dt + d\bar{W} \end{aligned} \tag{16}$$

- Let us assume there are $d_x$ slow variables and $L$ fast variables per slow variable. The maps, $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ functions, can be written as

$$\begin{aligned} \boldsymbol{f}_1 &= [f_{11}, \ldots, f_{1d_x}]^\top, \\ \boldsymbol{f}_2 &= [f_{21}, \ldots, f_{2L}]^\top, \end{aligned}$$

where

$$f_{1j}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha}) = -x_{j-1}(x_{j-2} - x_{j+1}) - x_j + F - \frac{HC}{B} \sum_{l=(j-1)L}^{Lj-1} z_l,$$

$$f_{2l}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha}) = -CBz_{l+1}(z_{l+2} - z_{l-1}) - Cz_l + \frac{CF}{B} + \frac{HC}{B} x_{\lfloor \frac{l-1}{L} \rfloor}.$$

# A Stochastic Lorenz 96 Model

- The model consists of two sets of dynamic variables, $\boldsymbol{x}(t)$ and $\boldsymbol{z}(t)$, that displays some key features of atmosphere dynamics (including chaotic behaviour) in a relatively simple model of arbitrary dimension. The system of differential equations takes the form

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}_1(\boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{\alpha})dt + dW \\ \dot{\boldsymbol{z}}(t) &= \boldsymbol{f}_2(\boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{\alpha})dt + d\bar{W} \end{aligned} \tag{16}$$

- Let us assume there are $d_x$ slow variables and $L$ fast variables per slow variable. The maps, $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ functions, can be written as

$$\begin{aligned} \boldsymbol{f}_1 &= \left[f_{11}, \ldots, f_{1d_x}\right]^\top, \\ \boldsymbol{f}_2 &= \left[f_{21}, \ldots, f_{2L}\right]^\top, \end{aligned}$$

where

$$\begin{aligned} f_{1j}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha}) &= -x_{j-1}(x_{j-2} - x_{j+1}) - x_j + F - \frac{HC}{B} \sum_{l=(j-1)L}^{Lj-1} z_l, \\ f_{2l}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha}) &= -CBz_{l+1}(z_{l+2} - z_{l-1}) - Cz_l + \frac{CF}{B} + \frac{HC}{B} x_{\lfloor \frac{l-1}{L} \rfloor}. \end{aligned}$$

# A Stochastic Lorenz 96 Model

- Applying the a discretization method we obtain a discrete-time version of the two-scale Lorenz 96 model

$$\bar{\boldsymbol{x}}_k = \bar{\boldsymbol{x}}_{k-1} + h\bar{\boldsymbol{f}}_{1,\sigma}(\bar{\boldsymbol{x}}_{k-1}, \bar{\boldsymbol{z}}_{k-1}, \alpha, \boldsymbol{v}_k), \tag{17}$$

$$\bar{\boldsymbol{z}}_k = \bar{\boldsymbol{z}}_{k-1} + h\bar{\boldsymbol{f}}_{2,\bar{\sigma}}(\bar{\boldsymbol{x}}_{k-1}, \bar{\boldsymbol{z}}_{k-1}, \alpha, \bar{\boldsymbol{v}}_k) \tag{18}$$

- We assume that the observations are linear but can only be collected from this system once every $T$ discrete time steps and only one out of $K$ slow variables can be observed, having the form

$$\boldsymbol{y}_n = \begin{bmatrix} x_{K,nT} \\ x_{2K,nT} \\ \vdots \\ x_{d_y K,nT} \end{bmatrix} + \boldsymbol{u}_n, \tag{19}$$

# A Stochastic Lorenz 96 Model

- Applying the a discretization method we obtain a discrete-time version of the two-scale Lorenz 96 model

$$\bar{\boldsymbol{x}}_k = \bar{\boldsymbol{x}}_{k-1} + h\bar{\boldsymbol{f}}_{1,\sigma}(\bar{\boldsymbol{x}}_{k-1}, \bar{\boldsymbol{z}}_{k-1}, \alpha, \boldsymbol{v}_k), \tag{17}$$

$$\bar{\boldsymbol{z}}_k = \bar{\boldsymbol{z}}_{k-1} + h\bar{\boldsymbol{f}}_{2,\bar{\sigma}}(\bar{\boldsymbol{x}}_{k-1}, \bar{\boldsymbol{z}}_{k-1}, \alpha, \bar{\boldsymbol{v}}_k) \tag{18}$$

- We assume that the observations are linear but can only be collected from this system once every $T$ discrete time steps and only one out of $K$ slow variables can be observed, having the form

$$\boldsymbol{y}_n = \begin{bmatrix} x_{K,nT} \\ x_{2K,nT} \\ \vdots \\ x_{d_y K,nT} \end{bmatrix} + \boldsymbol{u}_n, \tag{19}$$

# A Stochastic Lorenz 96 Model

- We generate both ground-truth values for the slow variables $\{\boldsymbol{x}_n\}_{n\geq 0}$ and synthetic observations, $\{\boldsymbol{y}_n\}_{n\geq 1}$. In the algorithm, as a forecast model, we use the differential equation

$$\dot{x}_j = f_j(\boldsymbol{x}, \boldsymbol{\theta}) = -x_{j-1}(x_{j-2} - x_{j+1}) - x_j + F - \ell(x_j, \mathrm{a}) \tag{20}$$

  where

  - $\mathrm{a} = [\mathrm{a}_1, \mathrm{a}_2]^\top$ is a (constant) parameter vector,
  - $\boldsymbol{\theta} = [F, \mathrm{a}^\top]^\top$ contains all the parameters and
  - function $\ell(x_j, \mathrm{a})$ is an ansatz, a polynomial in $x_j$ of degree 2, for the coupling term $\frac{HC}{B} \sum_{l=(j-1)L}^{Lj-1} \bar{z}_l$.

- Applying a discretization method we obtain

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + h\bar{\boldsymbol{f}}_\sigma(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{v}_k) \tag{21}$$

# Simulation setup

$$
\begin{aligned}
f_{1j}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha}) &= -x_{j-1}(x_{j-2} - x_{j+1}) - x_j + F - \frac{HC}{B} \sum_{l=(j-1)L}^{Lj-1} z_l, \\
f_{2l}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha}) &= -CBz_{l+1}(z_{l+2} - z_{l-1}) - Cz_l + \frac{CF}{B} + \frac{HC}{B} x_{\lfloor \frac{l-1}{L} \rfloor}. \qquad (22)
\end{aligned}
$$

| | |
|---|---|
| Integration step | $h = 10^{-3}$ |
| Model parameters | $F = 8$, $H = 0.75$, $C = 10$ and $B = 15$ |
| Fast variables | $L = 10$ |
| Observed variables | $K = 2$ |
| Noise scaling factors | $\sigma = \frac{h}{4} = 0.25 \times 10^{-3}$ and $\sigma_o = 4$ |

- The accuracy of the estimation is assessed in terms of empirical MSE per dimension averaged over several independent simulation runs.

$$
MSE_k = \frac{1}{d_x} \parallel \breve{\boldsymbol{x}}_k - \tilde{\boldsymbol{x}}_k \parallel^2 . \qquad (23)
$$

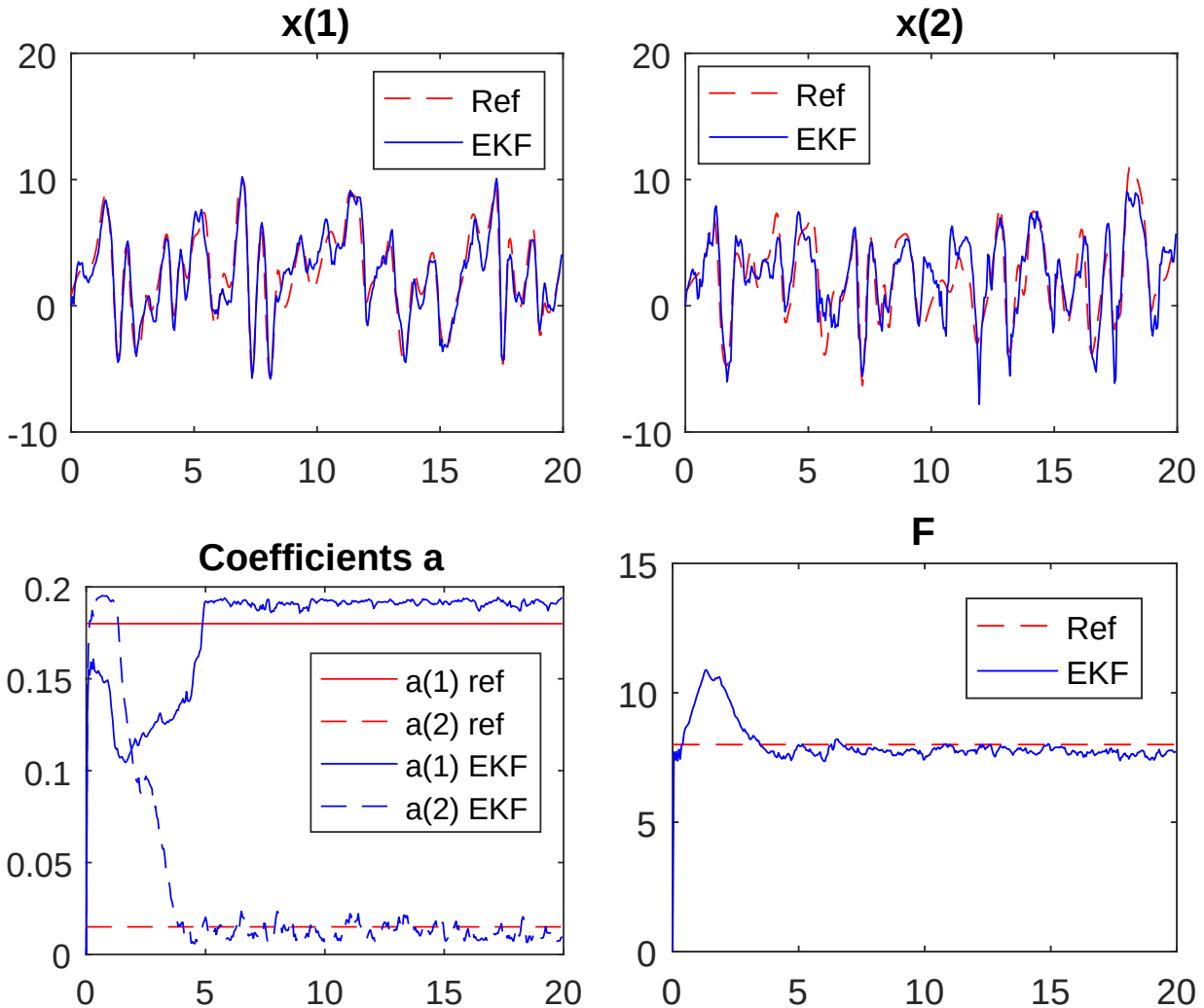
# NPF vs NHF's

- Dimension $d_x = 50$ and gap between observations of $hT = 0.05$ continuous-time units.
- $N = M = 1400$ particles in both layers of NPF.
- $N = 200$ particles for the first layer of NHFs and $M = 50$ for the EnKF's run.

| Algorithm | Running time | MSE |
|---|---|---|
| NPF | 6.85 hours | 1.872 |
| NHF + EKF | 1.196 minutes | 1.653 |
| NHF + EnKF | 11.674 minutes | 0.472 |

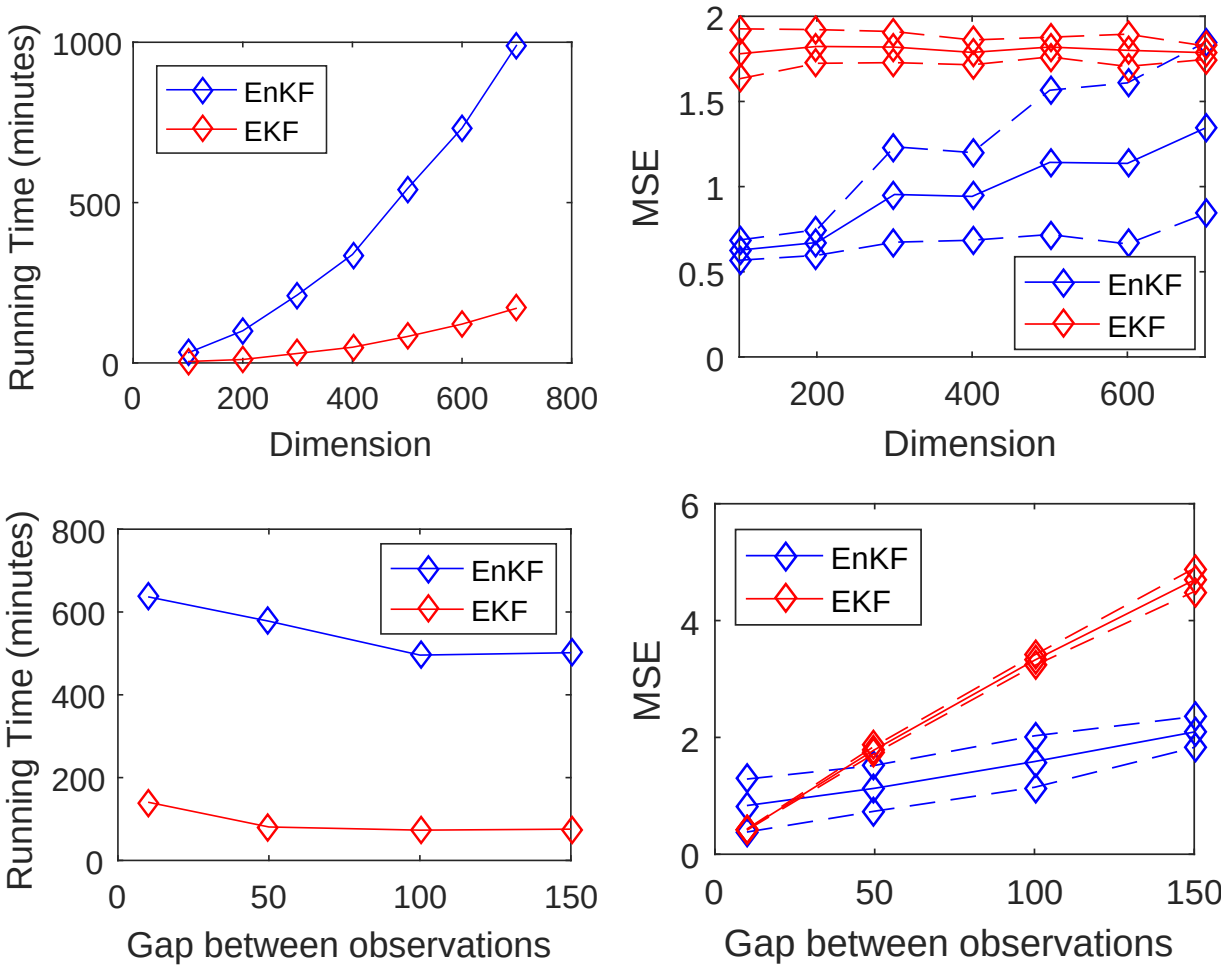
Table: Running times and average MSE (over time and state dimensions) for the NPF and two NHFs, based on the EKF and the EnKF, respectively.

# 4000-dimensional Experiment



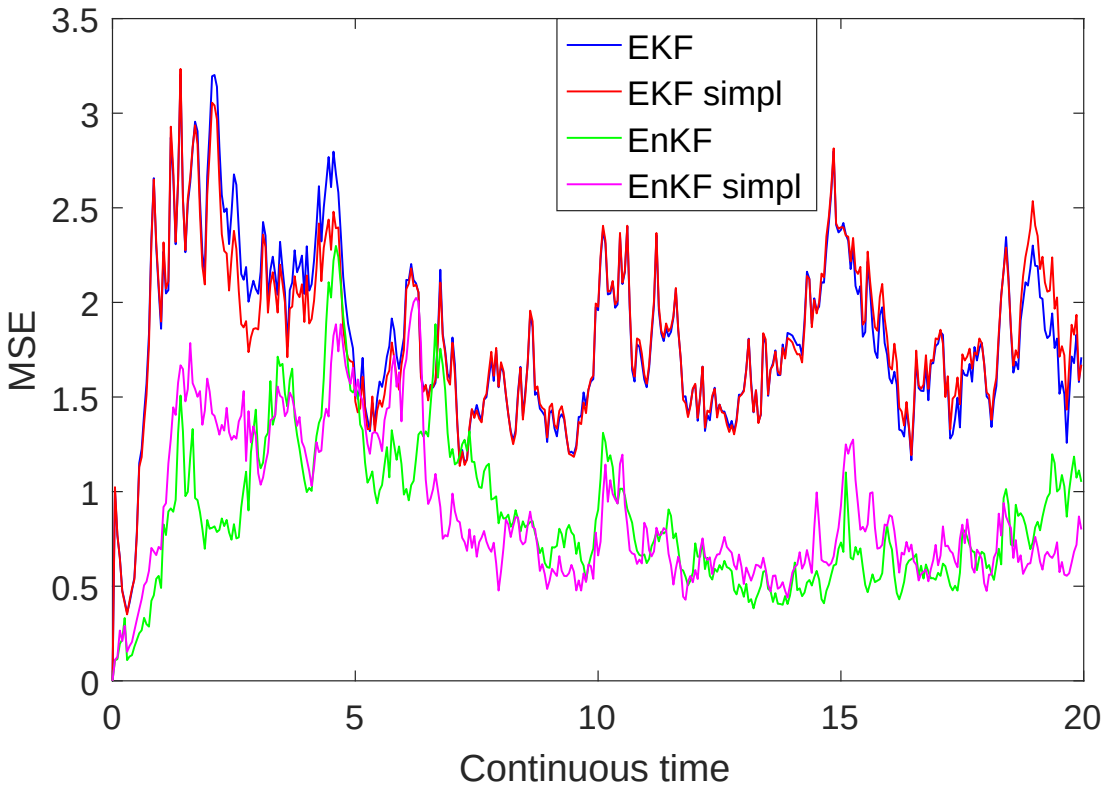

Sequences of state estimates in $x_1$ and $x_2$ and estimates of the parameters $\mathrm{a} = [\mathrm{a}_1, \mathrm{a}_2]^\top$ and $F$ over time in a 4,000-dimensional Lorenz 96 model. Variable $x_1$ is observed (in Gaussian noise), while $x_2$ is unobserved. The reference values are represented in red lines.

# NHF-EKF vs NHF-EnKF




Comparison of the NHF-EKF (red lines) and NHF-EnKF (blue lines) in terms of their running time and their *MSE* as the state dimension $d_x$ increase ($T = 50$ discrete time steps) and as the gap between observations $T$ increases ($d_x = 500$).

# Simplifications Performance




Comparison of the original scheme of NHF using both Gaussian filters (EKF and EnKF) and the inverse simplifications.

# Index

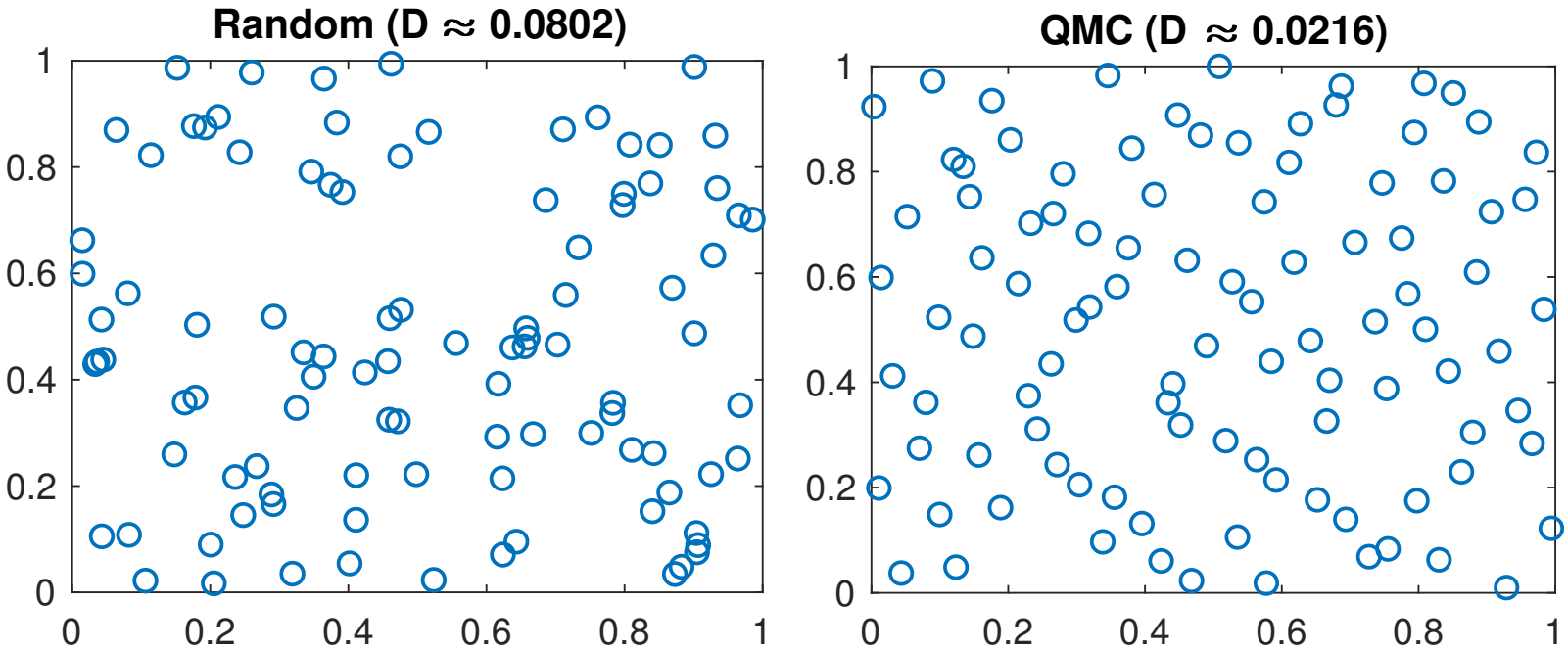
# QMC

- We are going to study an alternative to MC, that works similarly but in a **deterministic way**.
- The main difference with MC is the use of a **low-discrepancy** point set (sub-random or quasi-random) to generate the samples.
- Discrepancy is defined as

$$D(\boldsymbol{u}^{(1:N)}, \mathcal{A}) = \sup_{A \in \mathcal{A}} \left| \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}(\boldsymbol{u}^n \in [\boldsymbol{a}, \boldsymbol{b}]) - \lambda_s(A) \right|$$

where we have $N$ vectors $\boldsymbol{u}^n \in [0,1)^d$, $\lambda_s(A)$ is the volume of $A$ and $\mathcal{A}$ is the family of measurable subsets of $[0,1)^d$.

# QMC




QMC vs MC: $N = 100$ points sampled independently and uniformly in $[0,1)^2$ (left) and QMC sequence in $[0,1)^2$ of the same length (right).

# Discrepancy preserving map

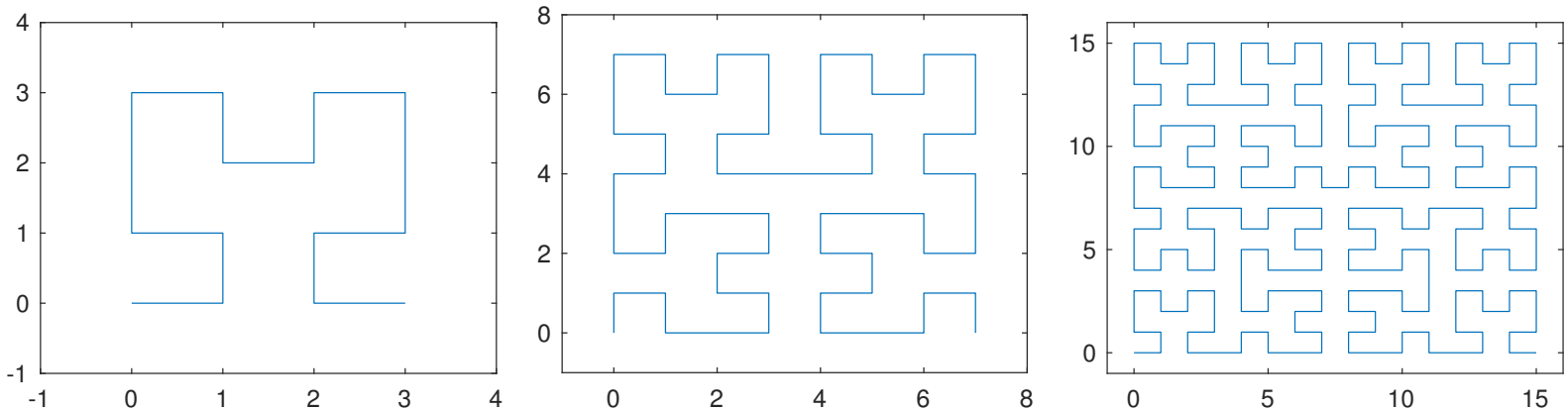
- $\Phi : D_\theta \longrightarrow [0,1)^d$ is a mapping that is discrepancy preserving, where $D_\theta$ is the support of $\boldsymbol{\theta}$.
- Then, $\Phi(\boldsymbol{\theta}) = (\Phi_1(\boldsymbol{\theta}_1), \ldots, \Phi_n(\boldsymbol{\theta}_n))$, where $\Phi_i$'s are continuous and strictly monotone, e.g.,

$$\Phi_i(\boldsymbol{\theta}_i) = \left[1 + exp(-\frac{\boldsymbol{\theta}_i - \underline{\boldsymbol{\theta}}_i}{\bar{\boldsymbol{\theta}}_i - \underline{\boldsymbol{\theta}}_i})\right]^{-1}, \quad i = 1, \ldots, d$$

where $\underline{\boldsymbol{\theta}}_i = \mu_{\boldsymbol{\theta}_{1:N}} - 2\sigma_{\boldsymbol{\theta}_{1:N}}$ and $\bar{\boldsymbol{\theta}}_i = \mu_{\boldsymbol{\theta}_{1:N}} + 2\sigma_{\boldsymbol{\theta}_{1:N}}$.

# Hilbert curve

- The Hilbert curve is a continuous fractal map $H : [0,1) \longrightarrow [0,1)^d$.
- It admits a pseudo-inverse $h : [0,1)^d \longrightarrow [0,1)$, which is used in SQMC to sort the samples $\bar{\theta}_i$.




Hilbert curves of order $m = 4, 8, 16$.

# Sequential QMC

## Algorithm

1. **Initialisation**

   1.1 *Generate a QMC point set* $\boldsymbol{v}_0^{(i)}$ *in* $(0,1]^d$ *and compute* $\bar{\boldsymbol{\theta}}_0^{(i)}$ *from* $\kappa_N(d\boldsymbol{\theta}, \boldsymbol{v}_0^{(i)})$.

   1.2 *Compute the normalised weights* $w_0^i \propto \hat{u}_0(\bar{\boldsymbol{\theta}}_0^i)$.

2. **Recursive step**

   2.1 *Generate a QMC point set* $\boldsymbol{V}_n^{(i)} = [r_n^{(i)}, \boldsymbol{v}_n^{(i)}]$ *in* $(0,1]^{d+1}$.

   2.2 *Hilbert sort: find permutation* $\sigma_{t-1}$ *such that* $h \circ \phi(\boldsymbol{\theta}_n^{\sigma_{t-1}(1)}) \leq \ldots \leq h \circ \phi(\boldsymbol{\theta}_n^{\sigma_{t-1}(N)})$ *and reorder the weights* $w_n^{\sigma_{t-1}(i)}$.

   2.3 *Find permutation* $\tau$ *such that* $r_t^{\tau(1)} \leq \ldots \leq r_t^{\tau(N)}$, *and obtain the set* $\{\boldsymbol{\theta}_{i=1}^N\}$ *applying the inversion method. Compute* $\bar{\boldsymbol{\theta}}_{n+1}^{(i)}$ *from* $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_n^i, \boldsymbol{v}_n^{(i)})$.

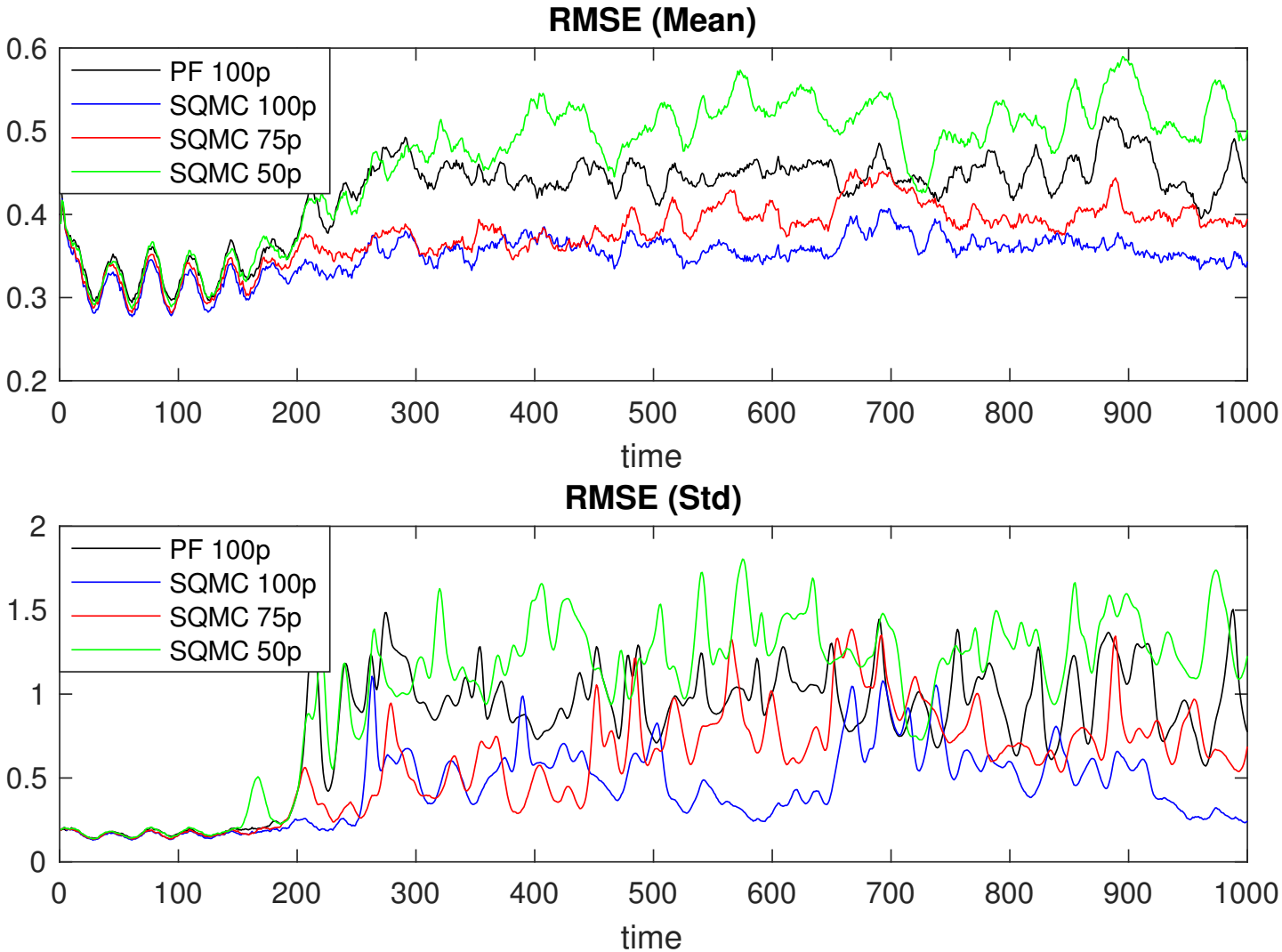
   2.4 *Compute* $\hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *and let* $w_n^i \propto \hat{u}_n(\bar{\boldsymbol{\theta}}_n^i)$ *be the normalised weight of* $\bar{\boldsymbol{\theta}}_n^i$.

# Lorenz 63 model

- In this model the state-space consists in three variables ($x$, $y$ and $z$), each of them with its respective SDE.

$$\begin{aligned} f_x(x, y, z, S) &= -S(x - y), \\ f_y(x, y, z, R) &= Rx - y - xz, \\ f_z(x, y, z, B) &= xy - Bz, \end{aligned}$$

- In order to do some experiments, they are discretized and the parameters are cosidered known ($S = 10$, $R = 28$ and $Q = \frac{8}{3}$).




SQMC vs SMC. Results obtained over 1000 simulation runs of a Lorenz 63 model.

# Index

# Conclusions

- We have introduced a recursive methodology to estimate the static parameters and the dynamic variables.
- The use of Gaussian filters is investigated as they admit fast implementations that can be well suited to high dimensional systems and two of them are simulated.
- Simplifications in the NHF-EKF and NHF-EnKF allowed the implementation of high dimensional systems.
- We have proved, under very general assumptions, that the proposed method converges (with optimal Monte Carlo rates) to a possibly biased version of the posterior distribution of the parameters.

# Future work

- Apply SQMC and other filters in the first layer of the algorithm in order to improve its performance.
- Use this algorithms with real data thanks to our collaboration with MeteoGalicia.