Aalto University
School of Electrical Engineering

Parallel square-root statistical linear regression for state estimation in nonlinear state space models

Fatemeh Yaghoobi

Aalto University, Finland

In the previous talks you learned how GP-based ODE solvers can be reformulated as Bayesian filtering and smoothing problem.

What is the problem and the goal

- Probabilistic state-space models

$$
\begin{aligned}
x_0 &\sim p(x_0) \\
x_k \mid x_{k-1} &\sim p(x_k \mid x_{k-1}), \quad k \geq 1, \\
y_k \mid x_k &\sim p(y_k \mid x_k), \quad k \geq 1
\end{aligned}
$$

- $x_k \in \mathbb{R}^{n_x}$ and $y_k \in \mathbb{R}^{n_y}$ are the state and measurement at time step k,
- $p(x_k \mid x_{k-1})$ is the transition density of the states,
- $p(y_k \mid x_k)$ is the conditional density of the measurements,
- $x_0 \sim p(x_0)$ is the initial distribution at time $k = 0$.
- Find smoothing results given a set of observations, that is, $p(x_k \mid y_{1:n})$, for $0 \leq k \leq n$, in general state-space models (SSMs).

What is the problem and the goal

Probabilistic state-space models:

$$
\begin{aligned}
x_0 &\sim p(x_0) \\
x_k \mid x_{k-1} &\sim p(x_k \mid x_{k-1}), \quad k \geq 1, \\
y_k \mid x_k &\sim p(y_k \mid x_k), \quad k \geq 1
\end{aligned}
$$

- We want to fill in a few thoughts on the following topics:

  - ⋆ **How to deal with the underlying SSM,**
  - ⋆ **How to find an efficient estimation:**
    1. **Reducing computation times –> parallel state estimation**
    2. **Reducing memory requirements and computational complexity –> square-root state estimation**

Elaborate the motivation and the goal

- There are Graphics processing units (GPUs) with a huge number of cores, for example, NVIDIA® GeForce RTX 3080 Ti 12 GB has 10240 CUDA cores whereas intel core i9 12900k has 16 cores
- Parallel Processing: It is possible to perform many calculations simultaneously.
- However, the algorithm itself should be parallel.
- Using shorter word lengths, such as 8-bit or 16-bit, is that they require less memory and computational resources than longer word lengths, such as 32-bit or 64-bit.

The main goal:
Find smoothing results by making the efficient use of available computational resources.

The development of the different formulations to deal with the underlying SSM

Remember what was the problem and the goal

Probabilistic state-space models:

$$
\begin{aligned}
x_0 &\sim p(x_0) \\
x_k \mid x_{k-1} &\sim p(x_k \mid x_{k-1}), \quad k \geq 1, \\
y_k \mid x_k &\sim p(y_k \mid x_k), \quad k \geq 1
\end{aligned}
$$

- We want to fill in a few thoughts on the following topics:

⋆ **How to deal with the underlying SSM,**

⋆ **How to find an efficient estimation:**

1. **Reducing computation times –> parallel state estimation**
2. **Reducing memory requirements and computational complexity –> square-root state estimation**

State Space Models and affine approximation

- Make affine approximations for state-space model

$$x_k \approx F_{k-1} x_{k-1} + c_{k-1} + q_{k-1}, \qquad q_k \sim \mathcal{N}(0, \Lambda_k), \tag{1a}$$

$$y_k \approx H_k x_k + d_k + v_k, \qquad v_k \sim \mathcal{N}(0, \Omega_k). \tag{1b}$$

Linearization parameters:

$$\{F_{0:n-1}, c_{0:n-1}, \Lambda_{0:n-1}, H_{1:n}, d_{1:n}, \Omega_{1:n}\},$$

State Space Models and affine approximation

- Make affine approximations for state-space model

$$x_k \approx F_{k-1} x_{k-1} + c_{k-1} + q_{k-1}, \qquad q_k \sim \mathcal{N}(0, \Lambda_k), \tag{1a}$$
$$y_k \approx H_k x_k + d_k + v_k, \qquad v_k \sim \mathcal{N}(0, \Omega_k). \tag{1b}$$

  Linearization parameters:

$$\{F_{0:n-1}, c_{0:n-1}, \Lambda_{0:n-1}, H_{1:n}, d_{1:n}, \Omega_{1:n}\},$$

- For a nonlinear state-space model with additive noise, parameters can be obtained by statistical linear regression (SLR)

State Space Models and affine approximation

- Make affine approximations for state-space model

$$
\begin{aligned}
x_k &\approx F_{k-1} x_{k-1} + c_{k-1} + q_{k-1}, & q_k &\sim \mathcal{N}(0, \Lambda_k), & \text{(1a)} \\
y_k &\approx H_k x_k + d_k + v_k, & v_k &\sim \mathcal{N}(0, \Omega_k). & \text{(1b)}
\end{aligned}
$$

  Linearization parameters:

$$
\{F_{0:n-1}, c_{0:n-1}, \Lambda_{0:n-1}, H_{1:n}, d_{1:n}, \Omega_{1:n}\},
$$

- For a nonlinear state-space model with additive noise, parameters can be obtained by statistical linear regression (SLR)
- For a general nonlinear state-space, parameters can be obtained by generalized statistical linear regression (GSLR)

State Space Models and affine approximation

- Idea behind SLR and GSLR methods:

State Space Models and affine approximation

- Idea behind SLR and GSLR methods:

  For given $y \mid x \sim p(y \mid x)$ and $p(x)$ with known $\mathbb{E}[x]$ and $\mathbb{V}[x]$, find affine approximation, $y \approx Hx + d + r$, where $r \sim (0, \Omega)$, by minimising the mean square error (MSE) with respect to $p(x)$

State Space Models and affine approximation

- Idea behind SLR and GSLR methods:

  For given $y \mid x \sim p(y \mid x)$ and $p(x)$ with known $\mathbb{E}[x]$ and $\mathbb{V}[x]$, find affine approximation, $y \approx Hx + d + r$, where $r \sim (0, \Omega)$, by minimising the mean square error (MSE) with respect to $p(x)$
- The solution to this problem

$$H = \mathbb{C}[x, y]^\top \mathbb{V}[x]^{-1}, \quad d = \mathbb{E}[y] - H\mathbb{E}[x], \quad \Omega = \mathbb{V}[y] - H\mathbb{V}[x]H^\top.$$

  where $\mathbb{C}[y, x]$ is the cross-covariance matrix of x and y.

State Space Models and affine approximation

- Idea behind SLR and GSLR methods:

  For given $y \mid x \sim p(y \mid x)$ and $p(x)$ with known $\mathbb{E}[x]$ and $\mathbb{V}[x]$, find affine approximation, $y \approx Hx + d + r$, where $r \sim (0, \Omega)$, by minimising the mean square error (MSE) with respect to $p(x)$
- The solution to this problem

$$H = \mathbb{C}[x, y]^\top \mathbb{V}[x]^{-1}, \quad d = \mathbb{E}[y] - H\mathbb{E}[x], \quad \Omega = \mathbb{V}[y] - H\mathbb{V}[x]H^\top.$$

where $\mathbb{C}[y, x]$ is the cross-covariance matrix of x and y.

GENERAL: $\boxed{y \mid x \sim p(y \mid x)}$

$$\mathbb{E}[y] = \mathbb{E}[\mathbb{E}[y \mid x]],$$
$$\mathbb{V}[y] = \mathbb{E}[\mathbb{V}[y \mid x]] + \mathbb{V}[\mathbb{E}[y \mid x]],$$
$$\mathbb{C}[y, x] = \mathbb{C}[\mathbb{E}[y \mid x], x],$$

ADDITIVE: $\boxed{y = g(x) + e,\ e \sim N(0, \Sigma_R)}$

$$\mathbb{E}[y] = \mathbb{E}[g(x)],$$
$$\mathbb{V}[y] = \Sigma_R,$$
$$\mathbb{C}[y, x] = \mathbb{C}[g(x), x].$$

State Space Models and affine approximation

GENERAL: $\boxed{y \mid x \sim p(y \mid x)}$

$$\mathbb{E}[y] = \mathbb{E}[\mathbb{E}[y \mid x]],$$

$$\mathbb{V}[y] = \mathbb{E}[\mathbb{V}[y \mid x]] + \mathbb{V}[\mathbb{E}[y \mid x]],$$

$$\mathbb{C}[y, x] = \mathbb{C}[\mathbb{E}[y \mid x], x],$$

ADDITIVE: $\boxed{y = g(x) + e, \; e \sim N(0, \Sigma_R)}$

$$\mathbb{E}[y] = \mathbb{E}[g(x)],$$

$$\mathbb{V}[y] = \Sigma_R,$$

$$\mathbb{C}[y, x] = \mathbb{C}[g(x), x].$$

State Space Models and affine approximation

GENERAL: $\boxed{y \mid x \sim p(y \mid x)}$	**ADDITIVE:** $\boxed{y = g(x) + e,\ e \sim N(0, \Sigma_R)}$
$\mathbb{E}[y] = \mathbb{E}[\mathbb{E}[y \mid x]],$	$\mathbb{E}[y] = \mathbb{E}[g(x)],$
$\mathbb{V}[y] = \mathbb{E}[\mathbb{V}[y \mid x]] + \mathbb{V}[\mathbb{E}[y \mid x]],$	$\mathbb{V}[y] = \Sigma_R,$
$\mathbb{C}[y, x] = \mathbb{C}[\mathbb{E}[y \mid x], x],$	$\mathbb{C}[y, x] = \mathbb{C}[g(x), x].$

- SLR: we have $y = g(x) + e$ and $p(x)$ with known $\mathbb{E}[x]$ and $\mathbb{V}[x]$.

State Space Models and affine approximation

GENERAL: $\boxed{y \mid x \sim p(y \mid x)}$

$$\mathbb{E}[y] = \mathbb{E}[\mathbb{E}[y \mid x]],$$

$$\mathbb{V}[y] = \mathbb{E}[\mathbb{V}[y \mid x]] + \mathbb{V}[\mathbb{E}[y \mid x]],$$

$$\mathbb{C}[y, x] = \mathbb{C}[\mathbb{E}[y \mid x], x],$$

ADDITIVE: $\boxed{y = g(x) + e,\ e \sim N(0, \Sigma_R)}$

$$\mathbb{E}[y] = \mathbb{E}[g(x)],$$

$$\mathbb{V}[y] = \Sigma_R,$$

$$\mathbb{C}[y, x] = \mathbb{C}[g(x), x].$$

- SLR: we have $y = g(x) + e$ and $p(x)$ with known $\mathbb{E}[x]$ and $\mathbb{V}[x]$.
- GSLR: we have $y \mid x \sim p(y \mid x)$ where the two conditional moments $\mathbb{E}[y \mid x]$ and $\mathbb{V}[y \mid x]$ of y are tractable, and $p(x)$ with known $\mathbb{E}[x]$ and $\mathbb{V}[x]$.

State Space Models and affine approximation

GENERAL: $\boxed{y \mid x \sim p(y \mid x)}$

$$\mathbb{E}[y] = \mathbb{E}[\mathbb{E}[y \mid x]],$$

$$\mathbb{V}[y] = \mathbb{E}[\mathbb{V}[y \mid x]] + \mathbb{V}[\mathbb{E}[y \mid x]],$$

$$\mathbb{C}[y, x] = \mathbb{C}[\mathbb{E}[y \mid x], x],$$

ADDITIVE: $\boxed{y = g(x) + e,\ e \sim N(0, \Sigma_R)}$

$$\mathbb{E}[y] = \mathbb{E}[g(x)],$$

$$\mathbb{V}[y] = \Sigma_R,$$

$$\mathbb{C}[y, x] = \mathbb{C}[g(x), x].$$

- SLR: we have $y = g(x) + e$ and $p(x)$ with known $\mathbb{E}[x]$ and $\mathbb{V}[x]$.
- GSLR: we have $y \mid x \sim p(y \mid x)$ where the two conditional moments $\mathbb{E}[y \mid x]$ and $\mathbb{V}[y \mid x]$ of y are tractable, and $p(x)$ with known $\mathbb{E}[x]$ and $\mathbb{V}[x]$.
- Sigma-point (SP) methods can be used to approximate the integrals .

State Space Models and affine approximation

GENERAL: $\boxed{y \mid x \sim p(y \mid x)}$

$$\mathbb{E}[y] = \mathbb{E}[\mathbb{E}[y \mid x]],$$
$$\mathbb{V}[y] = \mathbb{E}[\mathbb{V}[y \mid x]] + \mathbb{V}[\mathbb{E}[y \mid x]],$$
$$\mathbb{C}[y, x] = \mathbb{C}[\mathbb{E}[y \mid x], x],$$

ADDITIVE: $\boxed{y = g(x) + e,\ e \sim N(0, \Sigma_R)}$

$$\mathbb{E}[y] = \mathbb{E}[g(x)],$$
$$\mathbb{V}[y] = \Sigma_R,$$
$$\mathbb{C}[y, x] = \mathbb{C}[g(x), x].$$

- SLR: we have $y = g(x) + e$ and $\boxed{p(x) \text{ with known } \mathbb{E}[x] \text{ and } \mathbb{V}[x]}$.
- GSLR: we have $y \mid x \sim p(y \mid x)$ where the two conditional moments $\mathbb{E}[y \mid x]$ and $\mathbb{V}[y \mid x]$ of y are tractable, and $\boxed{p(x) \text{ with known } \mathbb{E}[x] \text{ and } \mathbb{V}[x]}$.
- Objective function:
  $\boxed{\text{minimising the mean square error (MSE) with respect to } p(x)}$

- Affine approximation

$$x_k \approx F_{k-1}^{(i)} x_{k-1} + c_{k-1}^{(i)} + q_{k-1}^{(i)}, \qquad q_k^{(i)} \sim \mathcal{N}(0, \Lambda_k^{(i)}), \tag{2a}$$

$$y_k \approx H_k^{(i)} x_k + d_k^{(i)} + v_k^{(i)}, \qquad v_k^{(i)} \sim \mathcal{N}(0, \Omega_k^{(i)}). \tag{2b}$$

define the parameters appearing at iteration i as:

$$\Gamma_{1:n}^{(i)} = \{F_{0:n-1}^{(i)}, c_{0:n-1}^{(i)}, \Lambda_{0:n-1}^{(i)}, H_{1:n}^{(i)}, d_{1:n}^{(i)}, \Omega_{1:n}^{(i)}\}, \tag{3}$$

- SLR with respect to the current approximate filtering/smoothing density (IPLF/IPLS) - **Additive case**- **(García-Fernández et al., 2017)**
- GSLR with respect to the current approximate filtering/smoothing density - **non-additive case** - **(Tronarp et al., 2018)**

The development of the parallel methods

Remember what was the problem and the goal

Linearized state-space models:

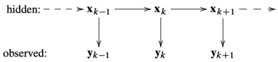
$$
\begin{aligned}
x_k &\approx F_{k-1}x_{k-1} + c_{k-1} + q_{k-1}, & q_k &\sim \mathcal{N}(0, \Lambda_k), \\
y_k &\approx H_k x_k + d_k + v_k, & v_k &\sim \mathcal{N}(0, \Omega_k).
\end{aligned}
$$

- We want to fill in a few thoughts on the following topics:

  - ⋆ **How to deal with the underlying SSM,**
  - ⋆ **How to find an efficient estimation:**
    1. **Reducing computation times –> parallel state estimation**
    2. **Reducing memory requirements and computational complexity –> square-root state estimation**

Parallel-in-time Bayesian filtering and smoothing

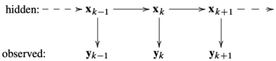
- Filtering and smoothing methods are sequential.
- Its computation takes $O(n)$ time. {n is the number of observations}



Question: Is there a way to compute the estimation faster?

Parallel-in-time Bayesian filtering and smoothing

- Filtering and smoothing methods are sequential.
- Its computation takes $O(n)$ time. {n is the number of observations}

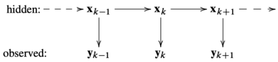


Question: Is there a way to compute the estimation faster?

Yes, Särkkä & García-Fernández presented a tractable parallel framework for filtering and smoothing distributions in a general case and in a linear Gaussian SSM

Parallel-in-time Bayesian filtering and smoothing

- Filtering and smoothing methods are sequential.
- Its computation takes $O(n)$ time. {n is the number of observations}

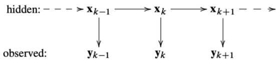


Question: Is there a way to compute the estimation faster?

Yes, Särkkä & García-Fernández presented a tractable parallel framework for filtering and smoothing distributions in a general case and in a linear Gaussian SSM

- The proposed temporally **parallel** algorithm

Parallel-in-time Bayesian filtering and smoothing



- Filtering and smoothing methods are sequential.
- Its computation takes $O(n)$ time. {n is the number of observations}

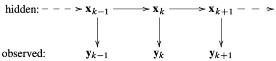
  Question: Is there a way to compute the estimation faster?

  Yes, Särkkä & García-Fernández presented a tractable parallel framework for filtering and smoothing distributions in a general case and in a linear Gaussian SSM

- The proposed temporally **parallel** algorithm

⋆ allows to benefit from parallelization capabilities of GPUs and TPUs.

Parallel-in-time Bayesian filtering and smoothing



- Filtering and smoothing methods are sequential.
- Its computation takes $O(n)$ time. {n is the number of observations}

  Question: Is there a way to compute the estimation faster?

  Yes, Särkkä & García-Fernández presented a tractable parallel framework for filtering and smoothing distributions in a general case and in a linear Gaussian SSM
- The proposed temporally **parallel** algorithm
  - ⋆ allows to benefit from parallelization capabilities of GPUs and TPUs.
  - ⋆ converts sequential $O(n)$ algorithms to $O(\log n)$ parallel algorithms.

- **Definition:** Given set of elements $\{a_k\}_{1 \leq k \leq n}$, and a binary associative operator $\otimes$, the algorithm computes $(a_1 \otimes a_2 \otimes \ldots \otimes a_K)_{1 \leq K \leq n}$ with $O(\log n)$ span-complexity.

- **Definition:** Given set of elements $\{a_k\}_{1 \leq k \leq n}$, and a binary associative operator $\otimes$, the algorithm computes $(a_1 \otimes a_2 \otimes \ldots \otimes a_K)_{1 \leq K \leq n}$ with $O(\log n)$ span-complexity.

  Associative operator $\otimes$
  $a \otimes (b \otimes c) = (a \otimes b) \otimes c$.

- **Definition:** Given set of elements $\{a_k\}_{1 \leq k \leq n}$, and a binary associative operator $\otimes$, the algorithm computes $(a_1 \otimes a_2 \otimes \ldots \otimes a_K)_{1 \leq K \leq n}$ with $O(\log n)$ span-complexity.

  Associative operator $\otimes$
  $a \otimes (b \otimes c) = (a \otimes b) \otimes c$.
- **Example:** Cumulative sum of $n = 8$ numbers

T	3	1	7	0	4	1	6	3

- **Definition:** Given set of elements $\{a_k\}_{1\leq k\leq n}$, and a binary associative operator $\otimes$, the algorithm computes $(a_1 \otimes a_2 \otimes \ldots \otimes a_K)_{1\leq K\leq n}$ with $O(\log n)$ span-complexity.

  Associative operator $\otimes$
  $a \otimes (b \otimes c) = (a \otimes b) \otimes c$.
- **Example:** Cumulative sum of $n = 8$ numbers

T	3	1	7	0	4	1	6	3

- Scan on a CPU, exactly n adds

- **Definition:** Given set of elements $\{a_k\}_{1 \leq k \leq n}$, and a binary associative operator $\otimes$, the algorithm computes $(a_1 \otimes a_2 \otimes \ldots \otimes a_K)_{1 \leq K \leq n}$ with $O(\log n)$ span-complexity.

  Associative operator $\otimes$
  $a \otimes (b \otimes c) = (a \otimes b) \otimes c$.

- **Example:** Cumulative sum of $n = 8$ numbers

T	3	1	7	0	4	1	6	3

- Scan on a CPU, exactly n adds
- Scan on a GPU, $2\ \log n$ steps

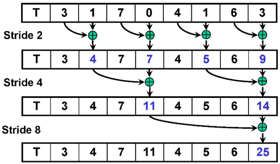
Parallel-in-time Bayesian filtering and smoothing - parallel scan method

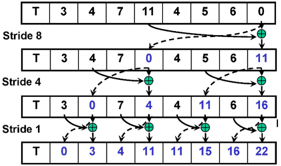
- **Definition:** Given set of elements $\{a_k\}_{1 \le k \le n}$, and a binary associative operator $\otimes$, the algorithm computes $(a_1 \otimes a_2 \otimes \ldots \otimes a_K)_{1 \le K \le n}$ with $O(\log n)$ span-complexity.

  Associative operator $\otimes$
  $a \otimes (b \otimes c) = (a \otimes b) \otimes c$.
- **Example:** Cumulative sum of $n = 8$ numbers

T	3	1	7	0	4	1	6	3

- Scan on a CPU, exactly n adds
- Scan on a GPU, $2 \log n$ steps





Parallel-in-time Bayesian filtering using parallel scan method

- Linear state space model:

$$
\begin{aligned}
p(x_k \mid x_{k-1}) &= N(x_k; F_{k-1} x_{k-1} + c_{k-1}, \Lambda_{k-1}), \\
p(y_k \mid x_k) &= N(y_k; H_k x_k + d_k, \Omega_k)
\end{aligned}
$$

Parallel-in-time Bayesian filtering using parallel scan method

- Linear state space model:

$$p(x_k \mid x_{k-1}) = N(x_k; F_{k-1}x_{k-1} + c_{k-1}, \Lambda_{k-1}),$$
$$p(y_k \mid x_k) = N(y_k; H_k x_k + d_k, \Omega_k)$$

- Parameterize the element $a_k = (p(x_k \mid y_k, x_{k-1}), p(y_k \mid x_{k-1}))$ as a set $\{A_k, b_k, C_k, \eta_k, J_k\}$ such that:

$$p(x_k \mid y_k, x_{k-1}) = N(x_k; A_k x_{k-1} + b_k, C_k), \quad p(y_k \mid x_{k-1}) \propto N_I(x_{k-1}; \eta_k, J_k),$$

Parallel-in-time Bayesian filtering using parallel scan method

- Linear state space model:

$$p(x_k \mid x_{k-1}) = N(x_k; F_{k-1}x_{k-1} + c_{k-1}, \Lambda_{k-1}),$$
$$p(y_k \mid x_k) = N(y_k; H_k x_k + d_k, \Omega_k)$$

- Parameterize the element $a_k = (p(x_k \mid y_k, x_{k-1}), p(y_k \mid x_{k-1}))$ as a set $\{A_k, b_k, C_k, \eta_k, J_k\}$ such that:

$$p(x_k \mid y_k, x_{k-1}) = N(x_k; A_k x_{k-1} + b_k, C_k), \quad p(y_k \mid x_{k-1}) \propto N_I(x_{k-1}; \eta_k, J_k),$$

- The result of the associative filtering operator: $\{A_i, b_i, C_i, \eta_i, J_i\} \otimes \{A_j, b_j, C_j, \eta_j, J_k\} := \{A_{ij}, b_{ij}, C_{ij}, \eta_{ij}, J_{ij}\}$

Parallel-in-time Bayesian filtering using parallel scan method

Initialization parameters: $\mathrm{a_k}$

$$\mathbf{1 < k \leq n}$$

$$S_k = H_k \Lambda_{k-1} H_k^\top + \Omega_k,,$$

$$K_k = \Lambda_{k-1} H_k{}^\top S_k^{-1},$$

$$A_k = (I_{n_x} - K_k H_k) F_{k-1},$$

$$b_k = c_{k-1} + K_k (y_k - H_k c_{k-1} - d_{k-1}),$$

$$C_k = (I_{n_x} - K_k H_k) \Lambda_{k-1},$$

$$J_k = (H_k F_{k-1})^\top S_k^{-1} H_k F_{k-1},$$

$$\eta_k = (H_k F_{k-1})^\top S_k^{-1} H_k (y_k - H_k c_{k-1} - d_k),$$

Combination results: $\otimes$

$$A_{ij} = A_j (I_{n_x} + C_i J_j)^{-1} A_i,$$

$$b_{ij} = A_j (I_{n_x} + C_i J_j)^{-1} (b_i + C_i \eta_j) + b_j,$$

$$C_{ij} = A_j (I_{n_x} + C_i J_j)^{-1} C_i A_j^\top + C_j,$$

$$J_{ij} = A_i^\top (I_{n_x} + J_j C_i)^{-1} J_j A_i + J_i$$

$$\eta_{ij} = A_i^\top (I_{n_x} + J_j C_i)^{-1} (\eta_j - J_j b_i) + \eta_i.$$

Parallel-in-time Bayesian filtering using parallel scan method

Initialization parameters: $\mathrm{a_k}$	**Combination results:** $\otimes$
$\mathbf{1 < k \leq n}$	
$S_k = H_k \Lambda_{k-1} H_k^\top + \Omega_k,,$	
$K_k = \Lambda_{k-1} {H_k}^\top S_k^{-1},$	
$A_k = (I_{n_x} - K_k H_k) F_{k-1},$	$A_{ij} = A_j (I_{n_x} + C_i J_j)^{-1} A_i,$
$b_k = c_{k-1} + K_k (y_k - H_k c_{k-1} - d_{k-1}),$	$b_{ij} = A_j (I_{n_x} + C_i J_j)^{-1} (b_i + C_i \eta_j) + b_j,$
$C_k = (I_{n_x} - K_k H_k) \Lambda_{k-1},$	$C_{ij} = A_j (I_{n_x} + C_i J_j)^{-1} C_i A_j^\top + C_j,$
$J_k = (H_k F_{k-1})^\top S_k^{-1} H_k F_{k-1},$	$J_{ij} = A_i^\top (I_{n_x} + J_j C_i)^{-1} J_j A_i + J_i$
$\eta_k = (H_k F_{k-1})^\top S_k^{-1} H_k (y_k - H_k c_{k-1} - d_k),$	$\eta_{ij} = A_i^\top (I_{n_x} + J_j C_i)^{-1} (\eta_j - J_j b_i) + \eta_i.$

- Recovering the filtering distribution at step k: $a_1 \otimes \cdots \otimes a_k = \binom{p(x_k \mid y_{1:k})}{p(y_{1:k})}$

Parallel-in-time Bayesian smoothing using parallel scan method

- Parameterize the element $a_k = p(x_k \mid y_{1:k}, x_{k+1})$ as a set $\{E_k, g_k, L_k\}$ such that

$$p(x_k \mid y_{1:k}, x_{k+1}) = N(x_k; E_k x_{k+1} + g_k, L_k)$$

Parallel-in-time Bayesian smoothing using parallel scan method

- Parameterize the element $a_k = p(x_k \mid y_{1:k}, x_{k+1})$ as a set $\{E_k, g_k, L_k\}$ such that

$$p(x_k \mid y_{1:k}, x_{k+1}) = N(x_k; E_k x_{k+1} + g_k, L_k)$$

- The result of the associative smoothing operator:

$$\{E_i, g_i, L_i\} \otimes \{E_j, g_j, L_j\} := \{E_{ij}, g_{ij}, L_{ij}\}$$

Parallel-in-time Bayesian smoothing using parallel scan method

- Parameterize the element $a_k = p(x_k \mid y_{1:k}, x_{k+1})$ as a set $\{E_k, g_k, L_k\}$ such that
$$p(x_k \mid y_{1:k}, x_{k+1}) = N(x_k; E_k x_{k+1} + g_k, L_k)$$
- The result of the associative smoothing operator:
$$\{E_i, g_i, L_i\} \otimes \{E_j, g_j, L_j\} := \{E_{ij}, g_{ij}, L_{ij}\}$$

Initialization parameters: $\mathbf{a_k}$	**Combination results: $\otimes$**
$E_k = P_k F_k^\top (F_k P_k^f F_k^\top + \Lambda_{k-1})^{-1},$	$E_{ij} = E_i E_j,$
$g_k = m_k^f - E_k(F_k m_k^f + c_k),$	$g_{ij} = E_i g_j + g_i,$
$L_k = P_k^f - E_k F_k P_k^f,$	$L_{ij} = E_i L_j E_i^\top + L_i.$

Parallel-in-time Bayesian smoothing using parallel scan method

- Parameterize the element $a_k = p(x_k \mid y_{1:k}, x_{k+1})$ as a set $\{E_k, g_k, L_k\}$ such that
$$p(x_k \mid y_{1:k}, x_{k+1}) = N(x_k; E_k x_{k+1} + g_k, L_k)$$
- The result of the associative smoothing operator:
$$\{E_i, g_i, L_i\} \otimes \{E_j, g_j, L_j\} := \{E_{ij}, g_{ij}, L_{ij}\}$$

Initialization parameters: $\mathbf{a_k}$	**Combination results: $\otimes$**
$E_k = P_k F_k^\top (F_k P_k^f F_k^\top + \Lambda_{k-1})^{-1},$	$E_{ij} = E_i E_j,$
$g_k = m_k^f - E_k(F_k m_k^f + c_k),$	$g_{ij} = E_i g_j + g_i,$
$L_k = P_k^f - E_k F_k P_k^f,$	$L_{ij} = E_i L_j E_i^\top + L_i.$

- Recovering the smoothing distribution at step k:
$a_k \otimes a_{k+1} \otimes \cdots \otimes a_n = p(x_k \mid y_{1:n})$

Parallel-in-time time complexity and work efficiency

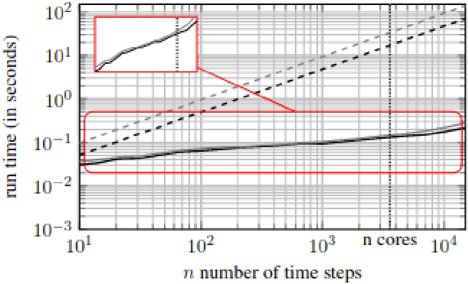
- Prefix-sum algorithms convert sequential $O(n)$ algorithms to $O(\log n)$ parallel algorithms.

Parallel-in-time time complexity and work efficiency

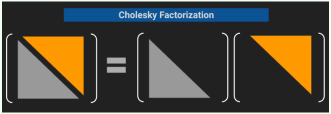
- Prefix-sum algorithms convert sequential $O(n)$ algorithms to $O(\log n)$ parallel algorithms.
- Work complexity (total number of operations) is $O(n)$. However, the parallel-scan algorithm achieves its efficiency by distributing the work across multiple processors.

Parallel-in-time time complexity and work efficiency

- Prefix-sum algorithms convert sequential $O(n)$ algorithms to $O(\log n)$ parallel algorithms.
- Work complexity (total number of operations) is $O(n)$. However, the parallel-scan algorithm achieves its efficiency by distributing the work across multiple processors.
- A parallel algorithm can be slow when the execution resources are saturated due to the low work efficiency.



The development of square-root methods



Remember what was the problem and the goal

Linearized state-space models:

$$\begin{aligned} x_k &\approx F_{k-1}x_{k-1} + c_{k-1} + q_{k-1}, && q_k \sim \mathcal{N}(0, \Lambda_k), \\ y_k &\approx H_k x_k + d_k + v_k, && v_k \sim \mathcal{N}(0, \Omega_k). \end{aligned}$$

- We want to fill in a few thoughts on the following topics:

> ⋆ **How to deal with the underlying SSM,**
>
> ⋆ **How to find an efficient estimation:**
>
> 1. **Reducing computation times –> parallel state estimation**
> 2. **Reducing memory requirements and computational complexity –> square-root state estimation**

General triangular square-root method

- Using square-root methods can reduce the number of bits required for representation, which can reduce the power consumption.

General triangular square-root method

- Using square-root methods can reduce the number of bits required for representation, which can reduce the power consumption.
- Shorter word lengths can improve numerical stability.

General triangular square-root method

- Using square-root methods can reduce the number of bits required for representation, which can reduce the power consumption.
- Shorter word lengths can improve numerical stability.
- Square-root methods can help to optimize the use of available resources, leading to more efficient and energy-saving algorithms.

General triangular square-root method

- Using square-root methods can reduce the number of bits required for representation, which can reduce the power consumption.
- Shorter word lengths can improve numerical stability.
- Square-root methods can help to optimize the use of available resources, leading to more efficient and energy-saving algorithms.
- Square-root form of a covariance matrix A:

General triangular square-root method

- Using square-root methods can reduce the number of bits required for representation, which can reduce the power consumption.
- Shorter word lengths can improve numerical stability.
- Square-root methods can help to optimize the use of available resources, leading to more efficient and energy-saving algorithms.
- Square-root form of a covariance matrix A:
    1. Leverage the QR-decomposition as $A^\top = Q\,R$,

General triangular square-root method

- Using square-root methods can reduce the number of bits required for representation, which can reduce the power consumption.
- Shorter word lengths can improve numerical stability.
- Square-root methods can help to optimize the use of available resources, leading to more efficient and energy-saving algorithms.
- Square-root form of a covariance matrix A:
  1. Leverage the QR-decomposition as $A^\top = Q\, R$,
  2. Define the triangularization operator as $\mathrm{Tria}(A) = R^\top$.

General triangular square-root method

- Using square-root methods can reduce the number of bits required for representation, which can reduce the power consumption.
- Shorter word lengths can improve numerical stability.
- Square-root methods can help to optimize the use of available resources, leading to more efficient and energy-saving algorithms.
- Square-root form of a covariance matrix A:
  1. Leverage the QR-decomposition as $A^\top = Q\,R$,
  2. Define the triangularization operator as $\mathrm{Tria}(A) = R^\top$.
  3. For any matrix $A \in \mathbb{R}^{n \times m}$, we have $\mathrm{Tria}(A)\mathrm{Tria}(A)^\top = AA^\top$.

General triangular square-root method

- Using square-root methods can reduce the number of bits required for representation, which can reduce the power consumption.
- Shorter word lengths can improve numerical stability.
- Square-root methods can help to optimize the use of available resources, leading to more efficient and energy-saving algorithms.
- Square-root form of a covariance matrix A:
  1. Leverage the QR-decomposition as $A^\top = Q\,R$,
  2. Define the triangularization operator as $\mathrm{Tria}(A) = R^\top$.
  3. For any matrix $A \in \mathbb{R}^{n\times m}$, we have $\mathrm{Tria}(A)\mathrm{Tria}(A)^\top = AA^\top$.
  4. For any block matrix $C = \begin{pmatrix} A & B \end{pmatrix} \in \mathbb{R}^{n\times(m+k)}$, with $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{n\times k}$, we have
$$\mathrm{Tria}(C)\mathrm{Tria}(C)^\top = AA^\top + BB^\top,$$

General triangular square-root method

- Using square-root methods can reduce the number of bits required for representation, which can reduce the power consumption.
- Shorter word lengths can improve numerical stability.
- Square-root methods can help to optimize the use of available resources, leading to more efficient and energy-saving algorithms.
- Square-root form of a covariance matrix A:
  1. Leverage the QR-decomposition as $A^\top = Q\,R$,
  2. Define the triangularization operator as $\mathrm{Tria}(A) = R^\top$.
  3. For any matrix $A \in \mathbb{R}^{n\times m}$, we have $\mathrm{Tria}(A)\mathrm{Tria}(A)^\top = AA^\top$.
  4. For any block matrix $C = \begin{pmatrix} A & B \end{pmatrix} \in \mathbb{R}^{n\times(m+k)}$, with $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{n\times k}$, we have
  $$\mathrm{Tria}(C)\mathrm{Tria}(C)^\top = AA^\top + BB^\top,$$
- The aim of the work: developing parallel-in-time square-root state estimation formulations for general nonlinear state-space model.

Remember what was the problem and the goal

Probabilistic state-space models:

$$
\begin{aligned}
x_0 &\sim p(x_0) \\
x_k \mid x_{k-1} &\sim p(x_k \mid x_{k-1}), \quad k \geq 1, \\
y_k \mid x_k &\sim p(y_k \mid x_k), \quad k \geq 1
\end{aligned}
$$

- The aim:

  - ⋆ **How to deal with the underlying SSM**
  - ⋆ **How to find an efficient estimation**

Parallel square-root method

- Cholesky decompositions of the conditional covariances of transition and observation densities defined as $\mathbb{S}_x[x_k \mid x_{k-1}]$ and $\mathbb{S}_y[y_k \mid x_k]$

Parallel square-root method

- Cholesky decompositions of the conditional covariances of transition and observation densities defined as $\mathbb{S}_x[x_k \mid x_{k-1}]$ and $\mathbb{S}_y[y_k \mid x_k]$
- We define the square-root linearization parameters, Γ^{sqrt}, in each iteration, i, as:

$$\Gamma_{1:k}^{\mathrm{sqrt},(i)} = \{F_{0:k-1}^{(i)}, c_{0:k-1}^{(i)}, S_{\Lambda_{0:k-1}}^{(i)}, H_{1:k}^{(i)}, d_{1:k}^{(i)}, S_{\Omega_{1:k}}^{(i)}\},$$

where S_{Λ_k} and S_{Ω_k} are the square-root matrices of Λ_k and Ω_k, respectively.

Parallel square-root method

- Cholesky decompositions of the conditional covariances of transition and observation densities defined as $\mathbb{S}_x[x_k \mid x_{k-1}]$ and $\mathbb{S}_y[y_k \mid x_k]$
- We define the square-root linearization parameters, Γ^{sqrt}, in each iteration, i, as:

$$\Gamma_{1:k}^{\text{sqrt},(i)} = \{F_{0:k-1}^{(i)}, c_{0:k-1}^{(i)}, S_{\Lambda_{0:k-1}}^{(i)}, H_{1:k}^{(i)}, d_{1:k}^{(i)}, S_{\Omega_{1:k}}^{(i)}\},$$

  where S_{Λ_k} and S_{Ω_k} are the square-root matrices of Λ_k and Ω_k, respectively.
- Computing the Linearization parameters $(F_{k-1}, c_{k-1}, S_{\Lambda_{k-1}})$ using the mean and the square root of the covariance of the smoothing density, m_k^s and N_k^s.

Standard version

$$F_{k-1} = \mathbb{C}[\mathbb{E}[x_k \mid x_{k-1}], x_{k-1}]^\top \mathbb{V}[x_{k-1}]^{-1},,$$

$$c_{k-1} = \mathbb{E}[\mathbb{E}[x_k \mid x_{k-1}]] - F_{k-1}\mathbb{E}[x_{k-1}],$$

$$\Lambda_{k-1} = \mathbb{E}[\mathbb{V}[x_k \mid x_{k-1}]] + \mathbb{V}[\mathbb{E}[x_k \mid x_{k-1}]] - F_{k-1}\mathbb{V}[x_{k-1}][F_{k-1}]^\top,$$

Square-root version using sigma-point

$$F_{k-1} \approx \sum_{i=1}^{s} w_i^c (\mathcal{Z}_{x,i} - \bar{z}_x)(\mathcal{X}_i - m_{k-1}^s)^\top (N_{k-1}^s [N_{k-1}^s]^\top)^{-1},$$

$$c_{k-1} \approx \bar{z}_x - F_{k-1} m_{k-1}^s,$$

$$S_{\Lambda_{k-1}} \approx \text{DownDate}(S'_{\Lambda_{k-1}}, F_{k-1} N_{k-1}^s).$$

Parallel square-root method

- Cholesky decompositions of the conditional covariances of transition and observation densities defined as $\mathbb{S}_x[x_k \mid x_{k-1}]$ and $\mathbb{S}_y[y_k \mid x_k]$
- We define the square-root linearization parameters, Γ^{sqrt}, in each iteration, i, as:

$$\Gamma^{\text{sqrt},(i)}_{1:k} = \{F^{(i)}_{0:k-1}, c^{(i)}_{0:k-1}, S^{(i)}_{\Lambda_{0:k-1}}, H^{(i)}_{1:k}, d^{(i)}_{1:k}, S^{(i)}_{\Omega_{1:k}}\},$$

where S_{Λ_k} and S_{Ω_k} are the square-root matrices of Λ_k and Ω_k, respectively.

- Computing the Linearization parameters $(F_{k-1}, c_{k-1}, S_{\Lambda_{k-1}})$ using the mean and the square root of the covariance of the smoothing density, m^s_k and N^s_k.

Standard version

$$F_{k-1} = \mathbb{C}[\mathbb{E}[x_k \mid x_{k-1}], x_{k-1}]^\top \mathbb{V}[x_{k-1}]^{-1},,$$

$$c_{k-1} = \mathbb{E}[\mathbb{E}[x_k \mid x_{k-1}]] - F_{k-1}\mathbb{E}[x_{k-1}],$$

$$\Lambda_{k-1} = \mathbb{E}[\mathbb{V}[x_k \mid x_{k-1}]] + \mathbb{V}[\mathbb{E}[x_k \mid x_{k-1}]] - F_{k-1}\mathbb{V}[x_{k-1}][F_{k-1}]^\top,$$

Square-root version using sigma-point

$$F_{k-1} \approx \sum_{i=1}^{s} w^c_i (\mathcal{Z}_{x,i} - \bar{z}_x)(\mathcal{X}_i - m^s_{k-1})^\top (N^s_{k-1}[N^s_{k-1}]^\top)^{-1},$$

$$c_{k-1} \approx \bar{z}_x - F_{k-1} m^s_{k-1},$$

$$S_{\Lambda_{k-1}} \approx \text{DownDate}(S'_{\Lambda_{k-1}}, F_{k-1} N^s_{k-1}).$$

- Same procedure for (H_k, d_k, S_{Ω_k})

Parallel square-root filtering method

- Parallel square-root version of a_k and $\otimes$ in filtering step

Initialization parameters: $\mathrm{a_k}$		Combination results:
$Y_1 = \Psi_{11}^{-}$,	$Y_k = \Psi_{11}$	
$K_1 = \Psi_{21}^{-}(\Psi_{11}^{-})^{-1}$,	$K_k = \Psi_{21}\Psi_{11}^{-1}$	
$A_1 = 0$	$A_k = (I_{n_x} - K_k H_k)F_{k-1}$,	$A_{ij} = A_j A_i - A_j U_i \Xi_{11}^{-\top} \Xi_{21}^{\top} A_i$
$b_1 = m_1^{-} + K_1[y_1 - H_1 m_1^{-} - d_1]$,	$b_k = c_{k-1} + K_k(y_k - H_k c_{k-1} - d_{k-1})$	$b_{ij} = A_j \left(I_{n_x} - U_i \Xi_{11}^{-\top} \Xi_{21}^{\top}\right)\left(b_i + U_i U_i^{\top} \eta_j\right) + b_j$,
$U_1 = \Psi_{22}^{-}$,	$U_k = \Psi_{22}$,	$U_{ij} = \mathrm{Tria}\left(\begin{pmatrix} A_j U_i \Xi_{11}^{-\top} & U_j \end{pmatrix}\right)$,
$Z_1 = F_0^{\top} H_1^{\top} Y_1^{-\top}$,	$Z_k = F_{k-1}^{\top} H_k^{\top} Y_k^{-\top}$	$Z_{ij} = \mathrm{Tria}\left(\begin{pmatrix} A_i^{\top} \Xi_{22} & Z_i \end{pmatrix}\right)$,
$\eta_1 = Z_1 Y_1^{-1}(y_1 - H_1 c_0 - d_1)$	$\eta_k = Z_k Y_k^{-1}(y_k - H_k c_{k-1} - d_k)$	$\eta_{ij} = A_i^{\top}\left(I_{n_x} - \Xi_{21}\Xi_{11}^{-1} U_i^{\top}\right)\left(\eta_j - Z_j Z_j^{\top} b_i\right) + \eta_i$.

where we define

$$\begin{pmatrix} \Psi_{11}^{-} & 0 \\ \Psi_{21}^{-} & \Psi_{22}^{-} \end{pmatrix} = \mathrm{Tria}\left(\begin{pmatrix} H_1 N_1^{-} & S_{\Omega_1} \\ N_1^{-} & 0 \end{pmatrix}\right), \quad \begin{pmatrix} \Psi_{11} & 0 \\ \Psi_{21} & \Psi_{22} \end{pmatrix} = \mathrm{Tria}\left(\begin{pmatrix} H_k S_{\Lambda_{k-1}} & S_{\Omega_k} \\ S_{\Lambda_{k-1}} & 0 \end{pmatrix}\right), \quad \begin{pmatrix} \Xi_{11} & 0 \\ \Xi_{21} & \Xi_{22} \end{pmatrix} = \mathrm{Tria}\left(\begin{pmatrix} U_i^{\top} Z_j & I_{n_x} \\ Z_j & 0 \end{pmatrix}\right).$$

Parallel square-root smoothing method

- Parallel square-root version of a_k and $\otimes$ in smoothing step

Initialization parameters: $\mathrm{a_k}$	**Combination results:**
$E_k = \Phi_{21}\Phi_{11}^{-1}$	$E_{ij} = E_i E_j$
$g_k = m_k^f - E_k(F_k m_k^f + c_k)$	$g_{ij} = E_i g_j + g_i$
$D_k = \Phi_{22}$	$D_{ij} = \mathrm{Tria}\left(\begin{pmatrix} E_i D_j & D_i \end{pmatrix}\right).$

where we define

$$\begin{pmatrix} \Phi_{11} & 0 \\ \Phi_{21} & \Phi_{22} \end{pmatrix} = \mathrm{Tria}\left(\begin{pmatrix} F_k N_k^f & S_{\Lambda_k} \\ N_k^f & 0 \end{pmatrix}\right),$$

A single iteration of this method consists in the following two steps:

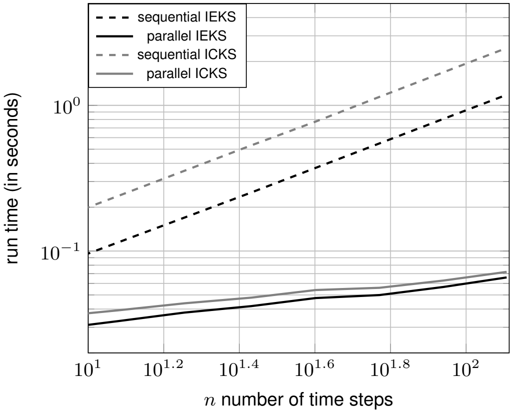
1. Use the square-root results of the smoother from the previous iteration in GSLR to linearize the dynamic and measurement models of the nonlinear system on all time steps $k = 1, \ldots, n$.
2. Implement the square-root version of parallel Kalman filter and smoother on the linearized system

Results

- population model

$$
\begin{aligned}
x_k &= \log(a) + x_{k-1} - \exp(x_{k-1}) + q_k, \quad q_k \sim \mathcal{N}(0, Q_k) \\
y_k \mid x_k &\sim \text{Poisson}(b \exp(x_k)), \\
x_0 &\sim \delta(x_0 - \log(c))
\end{aligned}
$$

GPU run time comparison of the standard parallel and sequential methods of IEKS and ICKS methods for population model



Results

Results of divergence rate comparison

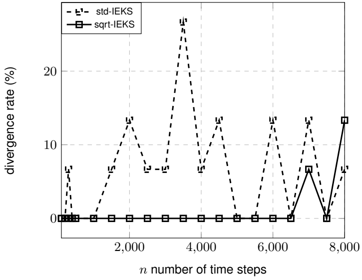
- ⋆ the parallel square-root/standard versions of IEKS and ICKS for coordinated-turn model:

while considering

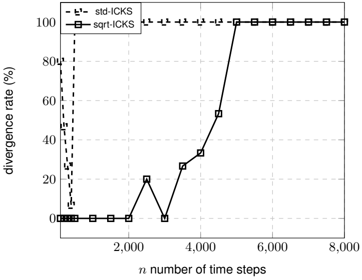
- ⋆ 32-bit floating-point numbers,
- ⋆ 15 runs.

and computing

- ⋆ the percentage of NaN (not a number) resulting log-likelihood estimates.



(a) Divergence rate of IEKS method.



(b) Divergence rate of ICKS method.

Summary

- We considered a general state space model, that is, nonlinear model with non-additive non-Gaussian noise.
- We linearized the system using GSLR method
- We provided parallel square-root formulations for linearized model

References

S. Särkkä and Á. F. García-Fernández, Temporal parallelization of Bayesian smoothers, IEEE Transactions on Automatic Control, 66 (2021).

F. Tronarp, Á. F. García-Fernández, and S. Särkkä, Iterative filtering and smoothing in nonlinear and non-Gaussian systems using conditional moments 2018.

F. Yaghoobi, A. Corenflos, S. Hassan, and S. Särkkä, Parallel iterated extended and sigma-point Kalman smoothers, in ICASSP 2021.

I. Arasaratnam and S. Haykin, Square-root quadrature Kalman filtering 2008.

S. Särkkä, Bayesian Filtering and Smoothing, Cambridge University Press, 2013