Computation-Aware Kalman Filtering and Smoothing

Marvin Pförtner[1] **Jonathan Wenger**[2] **Jon Cockayne**[3] **Philipp Hennig**[1]

[1] Tübingen AI Center, University of Tübingen [2] Columbia University [3] University of Southampton

Abstract

Kalman filtering and smoothing are the foundational mechanisms for efficient inference in Gauss–Markov models. However, their time and memory complexities scale prohibitively with the size of the state space. This is particularly problematic in spatiotemporal regression problems, where the state dimension scales with the number of spatial observations. Existing approximate frameworks leverage low-rank approximations of the covariance matrix. But since they do not model the error introduced by the computational approximation, their predictive uncertainty estimates can be overly optimistic. In this work, we propose a probabilistic numerical method for inference in high-dimensional Gauss–Markov models which mitigates these scaling issues. Our matrix-free iterative algorithm leverages GPU acceleration and crucially enables a tunable trade-off between computational cost and predictive uncertainty. Finally, we demonstrate the scalability of our method on a large-scale climate dataset.

1 INTRODUCTION

From language modeling to robotics to climate science, many application domains of machine learning generate data that are correlated in time. By describing the underlying temporal dynamics via a *state-space model* (SSM), the sequential structure can be leveraged to perform efficient inference. In machine learning, state-space models are widely used in reinforcement learning (Hafner et al., 2019), as well as in deep (Gu and Dao, 2023) and probabilistic (Särkkä and Svensson, 2023) sequence modeling. For example, suppose we aim to forecast temperature as a function of time from a set of K observations. Standard regression approaches have cubic cost $\mathcal{O}(K^3)$ in the number of data points (Rasmussen and Williams, 2006). Instead, one can leverage the temporal structure of the problem by representing it as a state-space model and performing Bayesian filtering and smoothing, which has *linear* time complexity $\mathcal{O}(K)$ (Särkkä and Svensson, 2023).

Challenges of a Large State-Space Dimension However, if the latent state has more than a few dimensions, inference in a state-space model can quickly become prohibitive. The overall computational cost is linear in time, but *cubic* in the size of the state space D with a *quadratic* memory requirement. Returning to the example above, suppose temperatures are given at a set of $N_{\boldsymbol{X}}$ spatial measurement locations around the globe. Assuming a non-zero correlation in temperature between those locations, the computational cost $\mathcal{O}(K \cdot D^3)$ with $D = \mathcal{O}(N_{\boldsymbol{X}})$ quickly becomes prohibitive. In response, many approximate filtering and smoothing algorithms have been proposed, e.g., based on sampling (e.g., Evensen, 1994), Krylov subspace methods (Bardsley et al., 2011), sketching (Berberidis and Giannakis, 2017), and dynamical-low-rank approximation (Schmidt et al., 2023). All of these methods inevitably introduce approximation error, which is *not* accounted for in the uncertainty estimates of the resulting posterior distributions.

Computation-Aware Filtering and Smoothing In this work, we introduce *computation-aware Kalman filters* (CAKFs) and *smoothers* (CAKSs): novel approximate versions of the Kalman filter and Rauch–Tung–Striebel (RTS) smoother. Approximations are introduced both to reduce the computational cost through low-dimensional projection of the data (Section 3.1) and memory burden through covariance truncation (Section 3.2). Alongside their prediction for the underlying dynamics, they return a combined uncertainty estimate quantifying both epistemic uncertainty *and* approximation error. Figure 1 showcases our approach

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

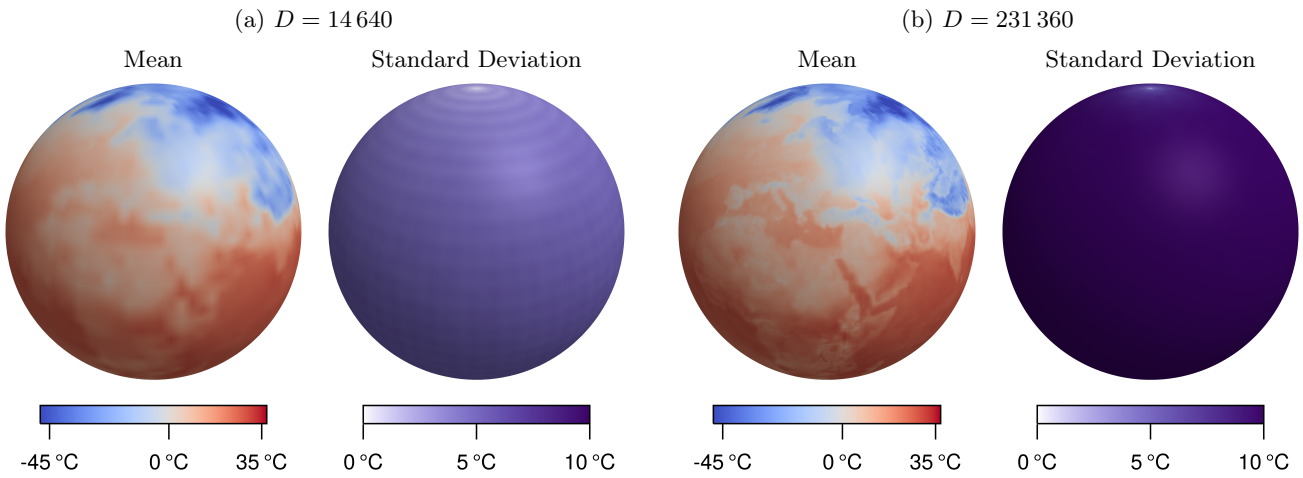


Figure 1: Spatio-temporal Gaussian process regression of Earth's surface temperature using the ERA5 dataset (Hersbach et al., 2020) via computation-aware filtering and smoothing for two different values of state-space dimension D. Kalman filtering and RTS smoothing would require in excess of 1.17 TiB of memory to generate Figure 1(b). Our novel algorithms scale to larger state-space dimension D with lower time and memory costs, resolving finer detail and achieving better predictive performance.

on a large-scale spatiotemporal regression problem.

Contribution We introduce the CAKF and CAKS, novel filtering and smoothing algorithms that are:

(1) *iterative* and *matrix-free*, and can fully leverage modern parallel hardware (i.e., GPUs);
(2) *more efficient* both in time and space than their standard versions (see Section 5.1); and
(3) *computation-aware*, i.e., they come with theoretical guarantees for their uncertainty estimates which capture the inevitable approximation error (Theorem 1).

We demonstrate the scalability of our approach empirically on climate data with up to $K \cdot N_{\boldsymbol{X}} \approx 4\text{M}$ observations and state-space dimension $D \approx 230\text{k}$.

2 BACKGROUND

Many temporal processes can be modeled with a linear-Gaussian state-space model, in which exact Bayesian inference can be done efficiently using the Kalman filter and Rauch–Tung–Striebel smoother.

2.1 Bayesian Inference in Linear-Gaussian State-Space Models

In the following, we want to infer the values of the *states* $\mathbf{u}_k \in \mathbb{R}^D$ of an unobserved discrete-time Gauss–Markov process $\{\mathbf{u}_k\}_{k=0}^K$ (or a discretized continuous-time Gauss–Markov process with $\mathbf{u}_k = \mathbf{u}(t_k^{\text{train}})$; see Appendix B.4) defined by the *dynamics model* $\mathbf{u}_{k+1} = \boldsymbol{A}_k \mathbf{u}_k + \boldsymbol{b}_k + \mathbf{q}_k$ with $\mathbf{u}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $\mathbf{q}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_k)$ from a given set of noisy *observations* $\{\boldsymbol{y}_k\}_{k=1}^K$ made through the *observation model* $\mathbf{y}_k \coloneqq \boldsymbol{H}_k \mathbf{u}_k + \boldsymbol{\epsilon}_k \in \mathbb{R}^{N_k}$. with $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_k)$. Collectively, the dynamics and observation models are referred to as a *linear-Gaussian state-space model* (LGSSM). The *initial state* $\mathbf{u}_0$, the *process noise* $\{\mathbf{q}_k\}_{k=0}^{K-1}$, and the *observation noise* $\{\boldsymbol{\epsilon}_k\}_{k=1}^K$ are pairwise independent. One can show that $\mathbf{u}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ with $\boldsymbol{\mu}_{k+1} = \boldsymbol{A}_k \boldsymbol{\mu}_k + \boldsymbol{b}_k$ and $\boldsymbol{\Sigma}_{k+1} = \boldsymbol{A}_k \boldsymbol{\Sigma}_k \boldsymbol{A}_k^\top + \boldsymbol{Q}_k$. The Kalman filter is an algorithm for computing conditional distributions of the form $\mathbf{u}_k \mid \mathbf{y}_{1:k} = \boldsymbol{y}_{1:k}$, $k = 1, \ldots, K$. It alternates recursively between computing the moments

$$\begin{aligned} \boldsymbol{m}_k^- &\coloneqq \boldsymbol{A}_{k-1} \boldsymbol{m}_{k-1} + \boldsymbol{b}_{k-1} \\ \boldsymbol{P}_k^- &\coloneqq \boldsymbol{A}_{k-1} \boldsymbol{P}_{k-1} \boldsymbol{A}_{k-1}^\top + \boldsymbol{Q}_{k-1} \end{aligned}$$

of $\mathbf{u}_k \mid \mathbf{y}_{1:k-1} = \boldsymbol{y}_{1:k-1} \sim \mathcal{N}(\boldsymbol{m}_k^-, \boldsymbol{P}_k^-)$ in the *predict step* and the moments

$$\boldsymbol{m}_k \coloneqq \boldsymbol{m}_k^- + \boldsymbol{P}_k^- \boldsymbol{H}_k \boldsymbol{G}_k^{-1} (\mathbf{y}_k - \boldsymbol{H}_k \boldsymbol{m}_k^-) \tag{2.1a}$$

$$\boldsymbol{P}_k \coloneqq \boldsymbol{P}_k^- - \boldsymbol{P}_k^- \boldsymbol{H}_k \boldsymbol{G}_k^{-1} \boldsymbol{H}_k^\top \boldsymbol{P}_k^- \tag{2.1b}$$

of $\mathbf{u}_k \mid \mathbf{y}_{1:k} = \boldsymbol{y}_{1:k} \sim \mathcal{N}(\boldsymbol{m}_k, \boldsymbol{P}_k)$ in the *update step*, where $\boldsymbol{G}_k \coloneqq \boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^\top + \boldsymbol{\Lambda}_k$ is the *innovation matrix*. The conditional distributions computed by the filter are mainly useful for forecasting. Interpolation requires the full Bayesian posterior $\mathbf{u}_k \,|\, \mathbf{y}_{1:K} = \boldsymbol{y}_{1:K} \sim \mathcal{N}(\boldsymbol{m}_k^{\text{s}}, \boldsymbol{P}_k^{\text{s}})$. Its moments can be computed from the filter moments via the Rauch–Tung–Striebel (RTS) smoother recursion

$$\begin{aligned} \boldsymbol{m}_k^{\text{s}} &\coloneqq \boldsymbol{m}_k + \boldsymbol{K}_k^{\text{s}} (\boldsymbol{m}_{k+1}^{\text{s}} - \boldsymbol{m}_{k+1}^-) \\ \boldsymbol{P}_k^{\text{s}} &\coloneqq \boldsymbol{P}_k + \boldsymbol{K}_k^{\text{s}} (\boldsymbol{P}_{k+1}^{\text{s}} - \boldsymbol{P}_{k+1}^-) (\boldsymbol{K}_k^{\text{s}})^\top \end{aligned}$$

with $\boldsymbol{K}_k^{\text{s}} \coloneqq \boldsymbol{P}_k \boldsymbol{A}_k^\top (\boldsymbol{P}_{k+1}^-)^{-1}$, $\boldsymbol{m}_K^{\text{s}} = \boldsymbol{m}_K$, and $\boldsymbol{P}_K^{\text{s}} = \boldsymbol{P}_K$. See Särkkä and Svensson (2023) for an in-depth introduction to Kalman filtering and RTS smoothing.

2.2 Spatiotemporal Regression

A major application of state-space models is spatiotemporal Gaussian Process (GP) regression (Hartikainen and Särkkä, 2010; Särkkä et al., 2013). Suppose we aim to learn a function $f^\star : [t_0, T] \times \mathbb{X} \to \mathbb{R}$ from training data $\{((t_k^{\text{train}}, \boldsymbol{X}_k^{\text{train}}), \boldsymbol{y}_k)\}_{k=1}^K$ with $t_k^{\text{train}} \in [t_0, T]$, $\boldsymbol{X}_k^{\text{train}} \in \mathbb{X}^{N_k}$, and $\boldsymbol{y}_k \in \mathbb{R}^{N_k}$. We assume a GP prior $\mathrm{f} \sim \mathcal{GP}(\mu, \Sigma)$ for $f^\star$. If the multi-output GP $\mathbf{f}(t, \boldsymbol{x}) \coloneqq (\partial_t^i \mathrm{f}(t, \boldsymbol{x}))_{i=0}^{D'-1} \in \mathbb{R}^{D'}$ defined by f and $D'-1$ of its time derivatives is a *space-time separable Gauss–Markov process*[1](STSGMP), then one can construct an equivalent linear-Gaussian state-space model (see Lemma C.2) with $\mathbf{u}_k \coloneqq \mathbf{f}(t_k^{\text{train}}, \boldsymbol{X}) \in \mathbb{R}^{D' \times N_{\boldsymbol{X}}}$, where $\boldsymbol{X} \in \mathbb{X}^{N_{\boldsymbol{X}}}$ such that $\boldsymbol{X}_k^{\text{train}} \subset \boldsymbol{X}$ for all $k = 1, \ldots, K$. Therefore, assuming that $\boldsymbol{X}_k^{\text{train}} = \boldsymbol{X}$ and $N_k = N_{\boldsymbol{X}}$, the computational cost of spatiotemporal GP regression can be reduced from $\mathcal{O}(K^3 N_{\boldsymbol{X}}^3)$ to $\mathcal{O}(K N_{\boldsymbol{X}}^3)$ via Bayesian filtering and smoothing.

3 COMPUTATION-AWARE KALMAN FILTERING

While filtering and smoothing are efficient in time, they scale prohibitively with the dimension D of the state space. Direct implementations of the Kalman filter incur two major computational challenges that are addressed with the CAKF:

(C1) The state covariances $\boldsymbol{P}_k^-, \boldsymbol{P}_k \in \mathbb{R}^{D \times D}$ need to be stored in memory, requiring $\mathcal{O}(D^2)$ space.

(C2) The inversion of the innovation matrix $\boldsymbol{G}_k \in \mathbb{R}^{N_k \times N_k}$ costs $\mathcal{O}(N_k^3)$ time and $\mathcal{O}(N_k^2)$ space.

Both of these quickly become prohibitive if D and/or N_k is large. To mitigate these costs, we apply iterative, matrix-free linear algebra in the Kalman recursion.

3.1 From Matrix-y to Matrix-Free

We start by noting that the Kalman filter's update step at time k conditions the predictive belief $\mathbf{u}_k \mid \mathbf{y}_{1:k-1} = \boldsymbol{y}_{1:k-1}$ on the observation that $\mathbf{y}_k = \boldsymbol{y}_k$. To reduce both the time and memory complexity of the update step, we project both sides of the observation onto a low-dimensional subspace: $\check{\mathbf{y}}_k \coloneqq \hat{\boldsymbol{S}}_k^\top \mathbf{y}_k = \hat{\boldsymbol{S}}_k^\top \boldsymbol{y}_k =: \check{\boldsymbol{y}}_k$, where $\hat{\boldsymbol{S}}_k \in \mathbb{R}^{N_k \times \check{N}_k}$ with $\check{N}_k \ll N_k$. The corresponding modified observation model then reads

$$\check{\mathbf{y}}_k = \underbrace{\hat{\boldsymbol{S}}_k^\top \boldsymbol{H}_k}_{=:\check{\boldsymbol{H}}_k} \mathbf{u}_k + \underbrace{\hat{\boldsymbol{S}}_k^\top \boldsymbol{\epsilon}_k}_{=:\check{\boldsymbol{\epsilon}}_k} \in \mathbb{R}^{\check{N}_k}, \tag{3.1}$$

where $\check{\boldsymbol{\epsilon}}_k \sim \mathcal{N}(\boldsymbol{0}, \check{\boldsymbol{\Lambda}}_k)$ with $\check{\boldsymbol{\Lambda}}_k \coloneqq \hat{\boldsymbol{S}}_k^\top \boldsymbol{\Lambda}_k \hat{\boldsymbol{S}}_k$. Consequently, the modified filtering equations can be obtained from (2.1) by substituting $\boldsymbol{y}_k \mapsto \check{\boldsymbol{y}}_k$, $\boldsymbol{H}_k \mapsto \check{\boldsymbol{H}}_k$ and $\boldsymbol{\Lambda}_k \mapsto \check{\boldsymbol{\Lambda}}_k$. The inversion of the innovation matrix $\check{\boldsymbol{G}}_k \in \mathbb{R}^{\check{N} \times \check{N}}$ in the projected update step then costs $\mathcal{O}(\check{N}_k^3)$ time and $\mathcal{O}(\check{N}_k^2)$ memory, which solves (C2).

Since $\hat{\boldsymbol{S}}_k$ is not square, the projection results in a loss of information and the filtering moments $\{\hat{\boldsymbol{m}}_k, \hat{\boldsymbol{P}}_k\}_{k=0}^K$ and $\{\hat{\boldsymbol{m}}_k^-, \hat{\boldsymbol{P}}_k^-\}_{k=0}^K$ obtained from the Kalman recursion with the modified observation model (3.1) are only approximations of the corresponding moments from the unmodified Kalman recursion. We can choose the columns of $\hat{\boldsymbol{S}}_k$, the *actions*, such that they retain the most informative parts of the observation, keeping the approximation error small; more on this in Section 3.3. Moreover, we show in Section 5 that the approximation error in the state mean will be accounted for by an increase in the corresponding state covariance and hence our inference procedure is *computation-aware* (in the sense of Wenger et al. (2022)). In essence, this is because we made the projection onto $\hat{\boldsymbol{S}}_k$ part of the modified observation model (3.1), i.e., the likelihood accounts for the fact that we do not observe the data $\boldsymbol{y}_k$ in the orthogonal complement of span $\left(\hat{\boldsymbol{S}}_k\right)$. The resulting posterior is sometimes called a *partial posterior*, and such posteriors are known to provide sensible uncertainty quantification under certain technical assumptions (Cockayne et al., 2022).

Since $\check{N}_k \ll N_k$, one can show that the updated state covariance $\hat{\boldsymbol{P}}_k$ under the modified observation model differs from the corresponding predictive state covariance $\hat{\boldsymbol{P}}_k^-$ by a low-rank downdate $\hat{\boldsymbol{P}}_k = \hat{\boldsymbol{P}}_k^- - \hat{\boldsymbol{P}}_k^- \check{\boldsymbol{H}}_k^\top \check{\boldsymbol{V}}_k (\hat{\boldsymbol{P}}_k^- \check{\boldsymbol{H}}_k^\top \check{\boldsymbol{V}}_k)^\top$, where $\check{\boldsymbol{V}}_k \in \mathbb{R}^{\check{N}_k \times \check{N}_k}$ with $\check{\boldsymbol{V}}_k \check{\boldsymbol{V}}_k^\top = \check{\boldsymbol{G}}_k^{-1}$. It turns out that the recursion for the state covariances in the Kalman filter is compatible with the low-rank downdate structure. More precisely, in Proposition B.3, we show that

$$\begin{aligned}\hat{\boldsymbol{P}}_k^- &= \boldsymbol{\Sigma}_k - \hat{\boldsymbol{M}}_k^- (\hat{\boldsymbol{M}}_k^-)^\top, \\ \hat{\boldsymbol{P}}_k &= \boldsymbol{\Sigma}_k - \hat{\boldsymbol{M}}_k \hat{\boldsymbol{M}}_k^\top,\end{aligned}$$

where $\hat{\boldsymbol{M}}_k^- = \boldsymbol{A}_{k-1} \hat{\boldsymbol{M}}_{k-1}$ and $\hat{\boldsymbol{M}}_k = (\hat{\boldsymbol{M}}_k^- \quad \hat{\boldsymbol{P}}_k^- \check{\boldsymbol{H}}_k^\top \check{\boldsymbol{V}}_k)$. Incidentally, this observation solves (C1): When implementing the CAKF using the recursions from Proposition B.3, we only need access to matrix-vector products $\boldsymbol{\Sigma}_k \boldsymbol{v}$, $\boldsymbol{A}_k \boldsymbol{v}$, $\boldsymbol{H}_k^\top \boldsymbol{v}$, and $\boldsymbol{\Lambda}_k \boldsymbol{v}$ with $\check{N}_k + 1$ vectors $\boldsymbol{v}$. In many cases, such matrix-vector products can be efficiently implemented or accurately approximated in a "matrix-free" fashion, i.e., without needing to store the matrix in memory, at cost (much) less than $\mathcal{O}(D^2)$ space and sometimes less than $\mathcal{O}(D^2)$ time (though matrix-free time complexity is often higher for dense matrices). For instance, this is possible if $\{\mathbf{u}_k\}_{k=0}^K$ is a discretization of a

[1]See Appendix C.1. While not every GP prior induces an STSGMP, a broad class of common spatiotemporal models do (see Remark C.5 for details).

continuous-time space-time separable Gauss–Markov process with known covariance function, in which case an expression for the entries $(\boldsymbol{\Sigma}_k)_{ij}$ of the state covariance is known. We emphasize the matrix-free implementation of our algorithm by highlighting matrices that are not instantiated in memory (e.g., $\hat{\boldsymbol{P}}_k^-$) in Algorithms 1 to 3. Assuming the rank of the downdates in Proposition B.3 is small (c.f. Section 3.2), such a matrix-free implementation of a Kalman filter incurs linear memory cost per time step.

Pseudocode for the procedure outlined above can be found in Algorithms 1 and 2. In the algorithm, the IsMissing line is included to allow a user to make predictions for intermediate states k for which there is no associated data. The choice of $\hat{\boldsymbol{S}}_k$ is given by a state-dependent Policy. In Algorithm 2, $\hat{\boldsymbol{S}}_k$ is selected in batch through a single call to Policy at the beginning of each update step. This is mostly presented for clarity; in practice we implement the update step as shown in Algorithm B.4. Algorithm B.4 can be derived as successive conditioning of $\hat{\mathbf{u}}_k^{f-}$ on the events $\langle \hat{\boldsymbol{s}}_k^{(i)}, \mathbf{y}_k \rangle_2 = \langle \hat{\boldsymbol{s}}_k^{(i)}, \boldsymbol{y}_k \rangle_2$ for $i = 1, \dots, \check{N}_k$, where the actions $\hat{\boldsymbol{s}}_k^{(i)}$ form the columns of $\hat{\boldsymbol{S}}_k$. One can show that this is equivalent to conditioning on $\hat{\boldsymbol{S}}_k^\top \mathbf{y}_k = \hat{\boldsymbol{S}}_k^\top \boldsymbol{y}_k$. However, such a sequential selection of the actions through calls to Policy that can access the current state of the iteration (e.g., through data residuals) allows the actions to adapt to the problem more effectively.

Algorithm 1 Computation-Aware Kalman Filter (CAKF)

fn Filter($\hat{\boldsymbol{m}}_0, \{\boldsymbol{\Sigma}_k, \boldsymbol{A}_k, \boldsymbol{b}_k, \boldsymbol{H}_k, \boldsymbol{\Lambda}_k, \boldsymbol{y}_k\}_{k=0}^K$)
  $\hat{\boldsymbol{M}}_0^+ \leftarrow (\ \ ) \in \mathbb{R}^{D \times 0}$
  for $k = 1, \dots, K$ **do**
    $\hat{\boldsymbol{m}}_k^- \leftarrow \boldsymbol{A}_{k-1}[\hat{\boldsymbol{m}}_{k-1}] + \boldsymbol{b}_{k-1}$ ▷ *Predict*
    $\hat{\boldsymbol{M}}_k^- \leftarrow \boldsymbol{A}_{k-1}[\hat{\boldsymbol{M}}_{k-1}^+]$
    if ¬IsMissing($\boldsymbol{y}_k$) **then**
      $\hat{\boldsymbol{m}}_k, \hat{\boldsymbol{M}}_k \leftarrow$ Update($\hat{\boldsymbol{m}}_k^-, \hat{\boldsymbol{M}}_k^-, \dots$)
    else
      $\hat{\boldsymbol{m}}_k, \hat{\boldsymbol{M}}_k \leftarrow \hat{\boldsymbol{m}}_k^-, \hat{\boldsymbol{M}}_k^-$
    $\hat{\boldsymbol{M}}_k^+ \leftarrow$ Truncate($\hat{\boldsymbol{M}}_k$)
  return $\{\hat{\boldsymbol{m}}_k, \hat{\boldsymbol{M}}_k\}_{k=0}^K$

3.2 Downdate Truncation

While the algorithm is matrix-free in the sense of not needing to compute and store $D \times D$ matrices, the accumulation of the downdate matrices $\hat{\boldsymbol{M}}_k$ results in an $\mathcal{O}(D \sum_{l=1}^k \check{N}_l)$ memory cost at step k, which can easily exceed $\mathcal{O}(D^2)$ as k grows. To address this, we introduce an optimal truncation of the downdate matrices in the Truncate procedure to control the memory requirements of the algorithm.

Algorithm 2 CAKF Update Step

fn Update($\hat{\boldsymbol{m}}^-, \hat{\boldsymbol{M}}^-, \boldsymbol{\Sigma}, \boldsymbol{H}, \boldsymbol{\Lambda}, \boldsymbol{y}$)
  $\hat{\boldsymbol{P}}^- \leftarrow \boldsymbol{\Sigma} - \hat{\boldsymbol{M}}^-(\hat{\boldsymbol{M}}^-)^\top$
  $\hat{\boldsymbol{S}} \leftarrow$ Policy($\hat{\boldsymbol{m}}^-, \hat{\boldsymbol{P}}^-, \dots$)
  $\check{\boldsymbol{H}}^\top \leftarrow \boldsymbol{H}^\top[\hat{\boldsymbol{S}}]$
  $\check{\boldsymbol{\Lambda}} \leftarrow \hat{\boldsymbol{S}}^\top \boldsymbol{\Lambda}[\hat{\boldsymbol{S}}]$
  $\check{\boldsymbol{y}} \leftarrow \hat{\boldsymbol{S}}^\top \boldsymbol{y}$
  $\check{\boldsymbol{G}} \leftarrow \check{\boldsymbol{H}} \hat{\boldsymbol{P}}^-[\check{\boldsymbol{H}}^\top] + \check{\boldsymbol{\Lambda}}$
  $\check{\boldsymbol{V}} \leftarrow$ LSQRT($\check{\boldsymbol{G}}^\dagger$)
  $\hat{\boldsymbol{w}} \leftarrow \check{\boldsymbol{H}}^\top \check{\boldsymbol{G}}^\dagger (\check{\boldsymbol{y}} - \check{\boldsymbol{H}} \hat{\boldsymbol{m}}^-)$
  $\hat{\boldsymbol{W}} \leftarrow \check{\boldsymbol{H}}^\top \check{\boldsymbol{V}}$
  $\hat{\boldsymbol{m}} \leftarrow \hat{\boldsymbol{m}}^- + \hat{\boldsymbol{P}}^-[\hat{\boldsymbol{w}}]$
  $\hat{\boldsymbol{M}} \leftarrow \begin{pmatrix} \hat{\boldsymbol{M}}^- & \hat{\boldsymbol{P}}^-[\hat{\boldsymbol{W}}] \end{pmatrix}$
  return $(\hat{\boldsymbol{m}}, \hat{\boldsymbol{M}})$

Consider the square root $\hat{\boldsymbol{M}}_k \in \mathbb{R}^{D \times r_k}$ of a belief covariance downdate. We truncate the downdate matrices by selecting $r_k^+ \leq r_k$ (typically $r_k^+ \ll r_k$), $\hat{\boldsymbol{M}}_k^+ \in \mathbb{R}^{D \times r_k^+}$, and $\boldsymbol{N}_k \in \mathbb{R}^{D \times (r_k - r_k^+)}$ such that $\hat{\boldsymbol{M}}_k \hat{\boldsymbol{M}}_k^\top = \hat{\boldsymbol{M}}_k^+ (\hat{\boldsymbol{M}}_k^+)^\top + \boldsymbol{N}_k \boldsymbol{N}_k^\top$ and approximating $\hat{\boldsymbol{M}}_k \approx \hat{\boldsymbol{M}}_k^+$ as well as $\hat{\boldsymbol{P}}_k \approx \hat{\boldsymbol{P}}_k^+ \coloneqq \boldsymbol{\Sigma}_k - \hat{\boldsymbol{M}}_k^+ (\hat{\boldsymbol{M}}_k^+)^\top$. Noting that $\hat{\boldsymbol{P}}_k^+ = \hat{\boldsymbol{P}}_k + \boldsymbol{N}_k \boldsymbol{N}_k^\top$, we realise that the truncation of the downdate can be interpreted as the addition of independent *computational uncertainty* (Wenger et al., 2022): additional noise $\mathbf{q}_k^{\text{comp}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{N}_k \boldsymbol{N}_k^\top)$ added to the posterior covariance to account for uncertainty due to incomplete computation, in this case, truncation. To represent this, we augment the dynamics model with additional prior states $\mathbf{u}_k^+, \mathbf{u}_k^-$ as visualized in Figure B.1, such that $\mathbf{u}_k^+ \coloneqq \mathbf{u}_k + \mathbf{q}_k^{\text{comp}}$, $\mathbf{u}_{k+1}^- \coloneqq \boldsymbol{A}_k \mathbf{u}_k^+ + \boldsymbol{b}_k + \mathbf{q}_k$, and $\mathbf{u}_k \coloneqq \mathbf{u}_k^-$. Even though the truncation leads to a further approximation of the state beliefs, this approximation will be *conservative*, which directly follows from the computational noise interpretation above.

We truncate by computing a singular-value decomposition of $\hat{\boldsymbol{M}}_k \hat{\boldsymbol{M}}_k^\top$, and dropping the subspace corresponding to the smallest singular vectors. By the Eckart–Young–Mirsky theorem (Mirsky, 1960), this truncation is optimal with respect to all unitarily invariant matrix norms. The effect of rank truncation is that at most $\mathcal{O}(dr_k^+)$ memory is required to store the downdate matrices, and that the cost of computing matrix-vector products with the truncated covariance $\hat{\boldsymbol{P}}_k^+$ is at most $\mathcal{O}(\rho_k + dr_k^+)$, where ρ_k is the cost of computing a matrix-vector product with $\boldsymbol{\Sigma}_k$.

3.3 Choice of Policy

It remains to specify a Policy defining the actions $\hat{\boldsymbol{S}}_k$. This can have a significant impact on the algorithm, both from the perspective of how close the CAKF states are to the states of the true Kalman filter and in terms of its computational cost. Heuristically, we would like to make $\check{N}_k$ as small as possible while

keeping $\mathcal{N}\left(\hat{\boldsymbol{m}}_k, \hat{\boldsymbol{P}}_k\right)$ close to $\mathcal{N}\left(\boldsymbol{m}_k, \boldsymbol{P}_k\right)$. We discuss and compare a number of natural policy choices in more detail in Appendix D.3. In the experiments in Section 7 we exclusively use Lanczos/CG-based directions, corresponding to choosing the current residual $\hat{\boldsymbol{r}}_k^{(i)}$ as the action in iteration i of Algorithm B.4, i.e., $\textsc{Policy}(i, \dots) = \hat{\boldsymbol{r}}_k^{(i)}$. We found these to perform well empirically compared to other choices (see Figure D.3), and similar policies have been found effective in related work (Wenger et al., 2022; Cockayne et al., 2019a).

4 COMPUTATION-AWARE RTS SMOOTHING

Algorithm 3 Computation-Aware RTS Smoother (CAKS)

fn $\textsc{Smooth}(\{\dots, \hat{\boldsymbol{m}}_k, \hat{\boldsymbol{M}}_k, \hat{\boldsymbol{w}}_k, \hat{\boldsymbol{W}}_k, \dots\}_{k=1}^K)$
  $\hat{\boldsymbol{w}}_K^{\mathrm{s}} \leftarrow \hat{\boldsymbol{w}}_K$
  $\hat{\boldsymbol{W}}_K^{\mathrm{s}} \leftarrow \hat{\boldsymbol{W}}_K$
  for $k = K-1, \dots, 0$ **do**
    $\hat{\boldsymbol{m}}_k^{\mathrm{s}} \leftarrow \hat{\boldsymbol{m}}_k + \hat{\boldsymbol{P}}_k \boldsymbol{A}_k^\top [\hat{\boldsymbol{w}}_{k+1}^{\mathrm{s}}]$
    $\hat{\boldsymbol{M}}_k^{\mathrm{s}} \leftarrow \begin{pmatrix} \hat{\boldsymbol{M}}_k & \hat{\boldsymbol{P}}_k \boldsymbol{A}_k^\top [\hat{\boldsymbol{W}}_{k+1}^{\mathrm{s}}] \end{pmatrix}$
    $\hat{\boldsymbol{w}}_k^{\mathrm{s}} \leftarrow \hat{\boldsymbol{w}}_k + (\boldsymbol{I} - \hat{\boldsymbol{W}}_k \hat{\boldsymbol{W}}_k^\top \hat{\boldsymbol{P}}_k^-) \boldsymbol{A}_k^\top [\hat{\boldsymbol{w}}_{k+1}^{\mathrm{s}}]$
    $\hat{\boldsymbol{W}}_k^{\mathrm{s}} \leftarrow \begin{pmatrix} \hat{\boldsymbol{W}}_k & (\boldsymbol{I} - \hat{\boldsymbol{W}}_k \hat{\boldsymbol{W}}_k^\top \hat{\boldsymbol{P}}_k^-) \boldsymbol{A}_k^\top [\hat{\boldsymbol{W}}_{k+1}^{\mathrm{s}}] \end{pmatrix}$
    $\hat{\boldsymbol{W}}_k^{\mathrm{s}} \leftarrow \textsc{Truncate}(\hat{\boldsymbol{W}}_k^{\mathrm{s}})$
  return $\{\hat{\boldsymbol{m}}_k^{\mathrm{s}}, \hat{\boldsymbol{M}}_k^{\mathrm{s}}\}_{k=0}^K$

If the state-space dimension D is large, naive implementations of the RTS smoother face similar challenges to those outlined for the Kalman filter in Section 3. This is due to the *smoother gain* matrices $\boldsymbol{K}_k^{\mathrm{s}}$ defined in Section 2.1 needing to be stored and inverted at $\mathcal{O}(D^3)$ time and $\mathcal{O}(D^2)$ memory cost. Fortunately, we can apply a similar strategy to Section 3.1 to make the smoother matrix-free. Specifically, in Proposition B.5 we show that the mean and covariance of the RTS smoother can be computed from quantities precomputed in the Kalman filter, i.e., without the need to compute any additional inverses: $\boldsymbol{m}_k^{\mathrm{s}} = \boldsymbol{m}_k + \boldsymbol{P}_k \boldsymbol{A}_k^\top \boldsymbol{w}_k^{\mathrm{s}}$ and $\boldsymbol{P}_k^{\mathrm{s}} = \boldsymbol{\Sigma}_k - \boldsymbol{M}_k^{\mathrm{s}} (\boldsymbol{M}_k^{\mathrm{s}})^\top$ with $\boldsymbol{M}_k^{\mathrm{s}} = \begin{pmatrix} \boldsymbol{M}_k & \boldsymbol{P}_k \boldsymbol{A}_k^\top \boldsymbol{W}_k^{\mathrm{s}} \end{pmatrix}$, where recursive expressions for $\boldsymbol{w}_k^{\mathrm{s}}$ and $\boldsymbol{W}_k^{\mathrm{s}}$ are given in Equations (B.1) and (B.2). Hence, just as for the filtering covariances, the smoother covariances take the form of a downdated prior covariance. The terms $\boldsymbol{w}_k^{\mathrm{s}}$ and $\boldsymbol{W}_k^{\mathrm{s}}$ can be efficiently computed from quantities cached in Algorithms 1 and 2 without materializing any $\mathcal{O}(D^2)$ matrices in memory. Applying Proposition B.5[2] in matrix-free form to the modified state-space model of the CAKF introduced in Section 3 yields Algorithm 3 – the *computation-aware RTS smoother* (CAKS).

While the cost of filtering is reduced by the low-dimensional projection of the data, the same does not hold for the smoother. Examining Algorithm 2 we see that $\hat{\boldsymbol{P}}_k^-$ only appears in a product with $\check{\boldsymbol{H}}$, whereas in Algorithm 3 products with $\hat{\boldsymbol{P}}_k^-$ appear directly. It is also necessary to truncate the directions $\hat{\boldsymbol{W}}_k^{\mathrm{s}}$ accumulated over the course of the smoother to mitigate a further $\mathcal{O}(D \sum_{l=k}^K \check{N}_l)$ storage cost. This is implemented using the same procedure as in Section 3.2.

In Appendix B.3 we use Matheron's rule to derive an algorithm for sampling from the posterior process, i.e., the CAKS states $\hat{\mathbf{u}}_k^{\mathrm{s}}$, pseudocode for which is given in Algorithm B.1. Counterintuitively, Algorithm B.1 can draw samples from the smoother states without the requirement to run the CAKS, since all necessary quantities have already been computed in the CAKF.

In Appendix B.4 we show that both the CAKF and CAKS can also be applied if the dynamics model is a continuous-time Gauss–Markov process. In this case, the CAKS provides efficient access to intermediate posterior states $\mathbf{u}(t) \,|\, \mathbf{y}_{1:K} = \boldsymbol{y}_{1:K}$ for arbitrary $t \in \mathbb{T}$.

5 THEORETICAL ANALYSIS

5.1 Computational Complexity

As mentioned in Section 3, the CAKF and CAKS assume that we can efficiently evaluate matrix-vector products with $\boldsymbol{\Sigma}_k$, $\boldsymbol{A}_k$, $\boldsymbol{H}_k^\top$, and $\boldsymbol{\Lambda}_k$, without synthesizing the matrices in memory. More precisely, we assume that matrix-vector products with these matrices can be computed with a memory complexity linear in the larger of their two dimensions and with the same worst-case time complexity.

Filtering The CAKF predict step at time k costs at most $\mathcal{O}(D^2 r_{k-1}^+)$ time and $\mathcal{O}(D r_{k-1}^+)$ memory. The CAKF update step at time k costs at most $\mathcal{O}\big((D N_k + N_k^2 + D^2)\check{N}_k\big)$ time and $\mathcal{O}\big((N_k + D)\check{N}_k\big)$ memory. The SVD downdate truncation has a time complexity of at most $\mathcal{O}(D(r_{k-1}^+ + \check{N}_k)^2)$.

Smoothing CAKS iteration k costs $\mathcal{O}\big(D(D + \check{N}_k) r_{k+1}^+ + D\check{N}_k^2\big)$ time and $\mathcal{O}\big((D + \check{N}_k) r_{k+1}^+\big)$ memory.

Simplified Complexities In practice, especially for spatiotemporal GP regression, it virtually always holds that $D = \mathcal{O}(N_k)$. With this assumption, the time and memory complexities of the CAKF update step simplify to $\mathcal{O}(D^2 \check{N}_k)$ and $\mathcal{O}(D\check{N}_k)$, respectively. Similarly, iteration k of the smoother then costs $\mathcal{O}(D^2(r_{k+1}^+ + \check{N}_k))$ time and $\mathcal{O}(D r_{k+1}^+)$ memory. It is also sometimes desirable to set $r_k^+ \leq r^{\max}$ and $\check{N}_k \leq \check{N}^{\max}$, i.e., uniform

[2] We would like to point out that Proposition B.5 may be of independent interest since it is (to the best of our knowledge) a novel result about the RTS smoother that can be used as an alternative to the standard RTS smoothing recursions for increased numerical stability.

in k, with $r^{\max} = \mathcal{O}(\check{N}^{\max})$. In this case, running both CAKF and CAKS for K time steps results in **worst-case time and memory complexities of $\mathcal{O}\big(KD^2\check{N}^{\max}\big)$ and $\mathcal{O}\big(KD\check{N}^{\max}\big)$.**

5.2 Error Bound for Spatiotemporal Regression

It is important to understand the impact of the approximations made by the CAKF and CAKS on the resulting predictions. So far we have argued informally, that the additional uncertainty of the CAKS captures the approximation error. We will now make this statement rigorous for the case of spatiotemporal regression.

Theorem 1 (Pointwise Worst-Case Prediction Error)**.** *Let $\mathbb{Z} = [t_0, T] \times \mathbb{X}$ and define a space-time separable Gauss–Markov process $\mathbf{f} \sim \mathcal{GP}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ such that its first component $\mathrm{f} \coloneqq \mathbf{f}_0 \sim \mathcal{GP}(\mu, \Sigma)$ defines a prior for the latent function $f^\star \in \mathbb{H}_\Sigma$ generating the data, assumed to be an element of the RKHS defined by $\mathbb{H}_\Sigma$. Given observation noise $\sigma^2 \geq 0$, let $y^\star(\cdot) \in \mathbb{H}_{\Sigma^\sigma}$ be the observed process with $\Sigma^\sigma(\boldsymbol{z}, \boldsymbol{z}') \coloneqq \Sigma(\boldsymbol{z}, \boldsymbol{z}') + \sigma^2\delta(\boldsymbol{z}, \boldsymbol{z}')$. Given training inputs $\boldsymbol{Z}^{\text{train}} \in \mathbb{Z}^N$ and targets $\boldsymbol{y}^{\text{train}} = y^\star(\boldsymbol{Z}^{\text{train}}) \in \mathbb{R}^N$, let $\hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z} \mid y^\star)$ and $\hat{\boldsymbol{P}}^{\text{s}}(\boldsymbol{z})$ be the mean and variance of the CAKS for an arbitrary test input $\boldsymbol{z} = (t, \boldsymbol{x}) \in \mathbb{Z} \setminus \boldsymbol{Z}^{\text{train}}$. Then*

$$\sup_{y \in \mathbb{H}_{\Sigma^\sigma} \setminus \{0\}} \frac{|y(\boldsymbol{z}) - \hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z} \mid y)_0|}{\|y\|_{\mathbb{H}_{\Sigma^\sigma}}} = \sqrt{\hat{\boldsymbol{P}}^{\text{s}}(\boldsymbol{z})_{0,0} + \sigma^2}. \tag{5.1}$$

If $\sigma^2 = 0$, this also holds for training inputs $\boldsymbol{z} \in \boldsymbol{Z}^{\text{train}}$.

The proof can be found in Appendix C.2. Theorem 1 says that the (relative) worst-case error of the CAKS's posterior mean $\hat{\boldsymbol{m}}^{\text{s}}(\cdot \mid y^\star)_0$ computed for data from the data-generating process $y^\star(\cdot)$ is tightly bounded by its predictive standard deviation $(\hat{\boldsymbol{P}}^{\text{s}}(\boldsymbol{z})_{0,0} + \sigma^2)^{1/2}$ (assuming no truncation). Importantly, this guarantee is of the same form as the one satisfied by the *exact* posterior predictive $\mathcal{GP}\left(\bar{\mu}^{y^\star}, \bar{\Sigma} + \sigma^2\delta\right)$ for the same prior (see Prop. 3.8 of Kanagawa et al. (2018)), except for the corresponding *approximations*. In this sense, **Theorem 1 makes the nomenclature *computation-aware* rigorous, since both the error due to finite data and the inevitable approximation error incurred by using $\hat{\boldsymbol{m}}^{\text{s}}_0 \approx \bar{\mu}^{y^\star}$ for prediction is quantified by its uncertainty estimate.** Finally, truncation only increases the marginal variance of the CAKS, which leads us to conject that the same guarantee as in Equation (5.1) holds with inequality for truncation.

6 RELATED WORK

Reducing the cost of Kalman filtering and smoothing in the high-dimensional regime is a fundamental problem. A large family of methods accelerates Kalman filtering by truncating state covariance matrices. This includes the *ensemble Kalman filter* (EnKF) (Evensen, 1994) and its variants, as well as the *reduced-rank Kalman filter* (RRKF) (Schmidt et al., 2023). However in contrast to this work, these approaches truncate the full state covariance rather than downdates, which can lead to overconfident uncertainty estimates (Schmidt et al., 2023, Appendix E). Some authors also propose dimension reduction techniques for the state space (e.g., Solonen et al., 2016) with the notable exception of Berberidis and Giannakis (2017), who focus on data dimension reduction, as we do here. Bardsley et al. (2011) use a similar Lanczos-inspired methodology; in certain settings, this is equivalent to low-dimensional projections of the data.

The main application of high-dimensional filtering and smoothing considered in Sections 5 and 7 is to spatiotemporal GP regression. This connection was first expounded in Hartikainen and Särkkä (2010); Särkkä et al. (2013) and generalized to a wider class of covariance functions in Todescato et al. (2020). These works focus on discretizing the GP to obtain a state-space model. One can also apply the Kalman filter directly in the infinite-dimensional setting, as proposed by Särkkä and Hartikainen (2012); Solin and Särkkä (2013).

The CAKF is a probabilistic numerical method (Hennig et al., 2015; Cockayne et al., 2019b; Hennig et al., 2022). In particular, Algorithm B.4 is closely related to the literature on *probabilistic linear solvers* (Cockayne et al., 2019a; Hennig, 2015; Wenger and Hennig, 2020), which frequently employ the Lanczos process. The idea of using such solvers for GP regression was explored in Wenger et al. (2022), which first proposed the construction of computation-aware solvers; the sense in which the CAKF is computation-aware is slightly different, in that the truncation of the covariance downdates also plays a role. Tatzel et al. (2023) explore similar ideas in the context of Bayesian generalized linear models.

7 EXPERIMENTS

To evaluate the computation-aware Kalman filter and smoother empirically, we apply it to spatiotemporal regression problems with synthetic data and a large-scale dataset from the geosciences.

Model All experiments will use Gaussian process priors $\mathrm{f} \sim \mathcal{GP}(0, \Sigma)$ with space-time separable covariance functions $\Sigma((t, \boldsymbol{x}), (t', \boldsymbol{x}')) = \Sigma^t(t, t')\Sigma^{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{x}')$, where Σ^t is chosen such that the prior can be represented by an equivalent STSGMP (see Appendix C.1). We will assume that the data is corrupted by i.i.d. Gaussian measurement noise with standard deviation λ. See Appendix D for details on the model hyperparameters

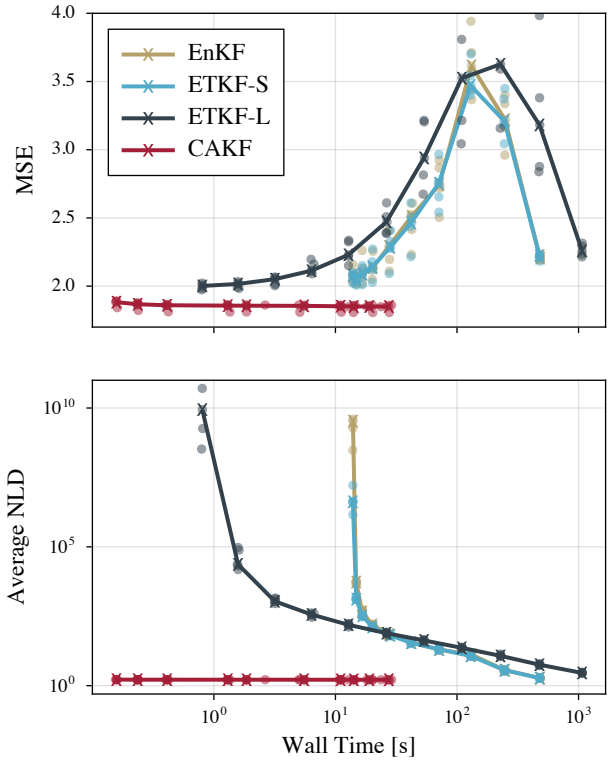


Figure 2: Comparison of the CAKF, the EnKF, and two variants of the ETKF on on-model data with state-space dimension $D = 20\,000$ while varying the rank parameters that govern the computational budget of the algorithms. The CAKF significantly outperforms the other filter variants for high-dimensional state spaces.

for the respective experiments.

Evaluation We measure the performance of each method via the mean squared error (MSE) (↓) of its predictive mean as well as via the average negative log density (NLD) (↓), which additionally takes uncertainty quantification into account. See Appendix D.1 for details on the evaluation metrics.

Implementation A flexible and efficient implementation of the CAKF and CAKS, including Matheron sampling and support for spatiotemporal modeling, is available as an open-source Julia library at

/ marvinpfoertner / ComputationAwareKalmanExperiments.jl.

When applying the CAKF and CAKS to separable spatiotemporal Gauss–Markov models, the main performance bottleneck is the computation of matrix-vector products with the prior's state covariance, since this involves a multiplication with a large kernel Gram matrix $\Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})$. Our Julia implementation includes a custom CUDA kernel for multiplying with Gramians generated by covariance kernels *without* materializing the matrix in memory.[3]

Hardware All experiments were run on a single dedicated machine equipped with an Intel i7-8700K CPU with 32 GB of RAM and an NVIDIA GeForce RTX 2080 Ti GPU with 11 GB of VRAM.

7.1 Comparison to Other Methods

We compare the performance of the CAKF/CAKS both to the standard Kalman filter and RTS smoother, as well as ensemble Kalman filters.

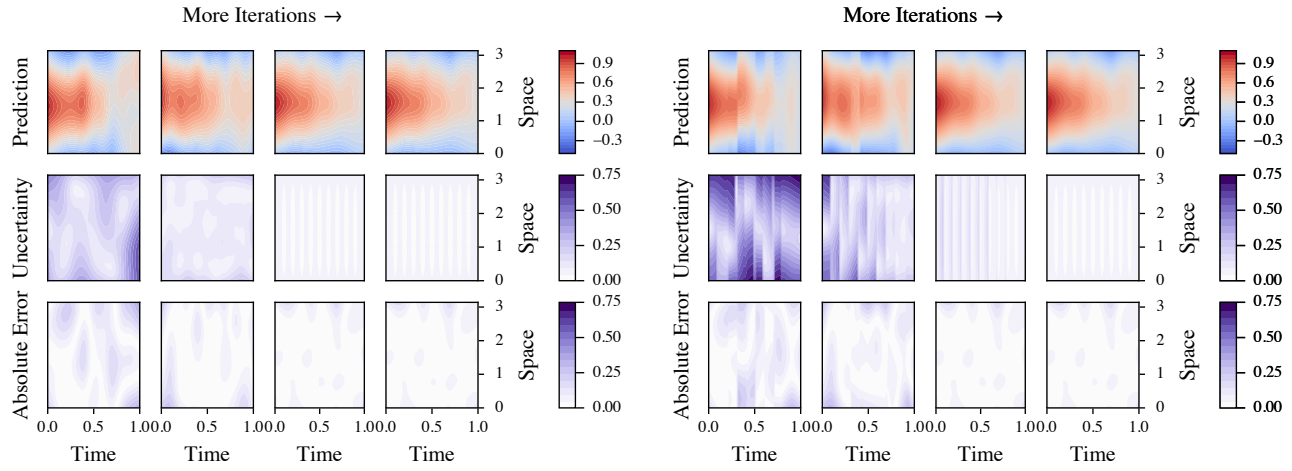
Data To isolate the effect of the algorithm on the prediction, we sample on-model datasets from the prior. To this end, we discretize the prior on regular grids in both time and space and draw a sample from the resulting discretized Gauss–Markov process. We pick a random subset of these points as training data which are subseqently corrupted by additive Gaussian measurement noise.

7.1.1 Comparison to EnKF and ETKF

We start by comparing the CAKF against its main competitors: ensemble Kalman filters. Among those, we consider the ensemble Kalman filter (EnKF) (Evensen, 1994) and two variants of the ensemble transform Kalman filter (ETKF) (Bishop et al., 2001). The variants of the ETKF differ only in their initialization and prediction steps. Namely, the first variant (ETKF-S) uses the same sampling-based initialization and prediction steps as the EnKF, while the second variant (ETKF-L) uses the Lanczos process for both (see Appendix D.2.1). We vary the rank parameter $r = 1, 2, 4, 8, \ldots, 1024$ that governs the computational budget of the algorithms, starting at $r = 2$ for the EnKF and the ETKF-S and at $r = 1$ for the ETKF-L and the CAKF (and we only run the latter up to $r = 512$). For a fair comparison, we set both the maximal number of iterations as well as the truncation rank of the CAKF to the same constant value, i.e., $r_k^+ = \check{N}_k^{\max} = r$. Note that the scalability issues of the EnKF and ETKF-S outlined in Appendix D.4 limit the state-space dimension of this problem. The results are visualized in Figures 2 and D.1.

Interpretation The CAKF consistently outperforms the ensemble methods in terms of MSE and average NLD across all ranks. While the difference in MSE is comparatively small, the CAKF achieves a significantly lower average NLD. This indicates that the uncertainty quantification of the CAKF is considerably

[3] For reference, we observed up to a 600-fold speedup over the default CPU implementation when multiplying a 9600×9600 kernel Gram matrix of a three-dimensional Matérn($3/2$) kernel with a 9600×128 matrix.



(a) CAKS

(b) CAKS + Truncation

Figure 3: Predictive mean, predictive standard deviation, and pointwise absolute error for an increasing maximal number of iterations $\check{N}^{\max} \geq \check{N}_k$ per time step on a synthetic spatiotemporal regression problem.

more accurate than that of the ensemble methods.

7.1.2 Comparison to Kalman Filter and RTS Smoother

As the CAKF and CAKS are approximations to the Kalman filter and RTS smoother, respectively, it is important to assess the approximation error. To this end, we compute the errors in the mean and covariance estimates provided by the CAKF and CAKS with varying rank parameters $r_k^+ = \check{N}_k^{\max} \in \{1, 16, 32, 64\}$ as compared to the respective mean and covariance estimates produced by the Kalman filter and RTS smoother on a sufficiently small problem for which we can run these. The results can be found in Figure D.2.

Interpretation As expected, the error introduced by the CAKF and CAKS decreases with increasing rank. Moreover, both the error in the mean and the error in the covariance are bounded and stable over time.

7.2 Impact of Truncation

To visualize the effect of the downdate truncation, we consider a synthetic dataset generated by adding Gaussian measurement noise to the target function $f^\star(t, x) \coloneqq \sin(x)\exp(-t)$. In Figure 3, we illustrate the predictive mean of the CAKS, its predictive standard deviation, and the corresponding pointwise (absolute) error, both with (Figure 3(a)) and without (Figure 3(b)) truncation. Each column corresponds to a larger number of iterations $\check{N}^{\max}$. Here, the truncation rank is chosen as $r_k^+ = \min(2\check{N}_k, r_k^+)$.

Interpretation For an increasing number of iterations, the error in the predictive mean $\hat{\boldsymbol{m}}^{\mathrm{s}}(\boldsymbol{z})_0$ decreases and its uncertainty $\hat{\boldsymbol{P}}^{\mathrm{s}}(\boldsymbol{z})_{0,0}$ reduces correspondingly. When the belief is (optimally) truncated to save memory, the uncertainty tends to increase as Figure 3(b) illustrates. Notice the trade-off between the degree of truncation and the impact on the prediction. Finally, Figure 3 also illustrates that the uncertainty bounds the error in the predictive mean as shown in Theorem 1.

7.3 Large-Scale Climate Dataset

To demonstrate that our approach scales to large, real-world problems, we use the CAKS to interpolate earth surface temperature data over time using an STSGMP prior on the sphere.

Data We consider the "2 m-temperature" variable from the ERA5 global reanalysis dataset (Hersbach et al., 2020). The data reside on a 1440×721 spatial latitude-longitude grid with an hourly temporal resolution. For our experiments, we selected the first 48 h of 2022 with a temporal stride of 1 h, i.e., $K = 48$. To show the effect of different problem sizes on our algorithms, we downsample the dataset by factors of 3, 6, 12, and 24 along both spatial dimensions using nearest neighbor downsampling. A regular subgrid consisting of 25 % of the points in the downsampled dataset is used for testing, while the remaining points are used as a training set. The total number of spatial points and the number of spatial training points for each downsampled version of the dataset can be found in Table D.1.

Evaluation We run the CAKF and the CAKS for the four different problem sizes (spatial downsampling factors of 3, 6, 12, and 24) corresponding to increasing state-space dimension (see Table D.1), for a maximum of $\approx 4\,000\,000$ training datapoints. For each problem size, we vary the computational budget, defined by the number of actions $\check{N}_k$ and the rank $r_k^+ = \min(2\check{N}_k, r_k)$ of the downdates after truncation. As the $\check{N}_k$ increases,

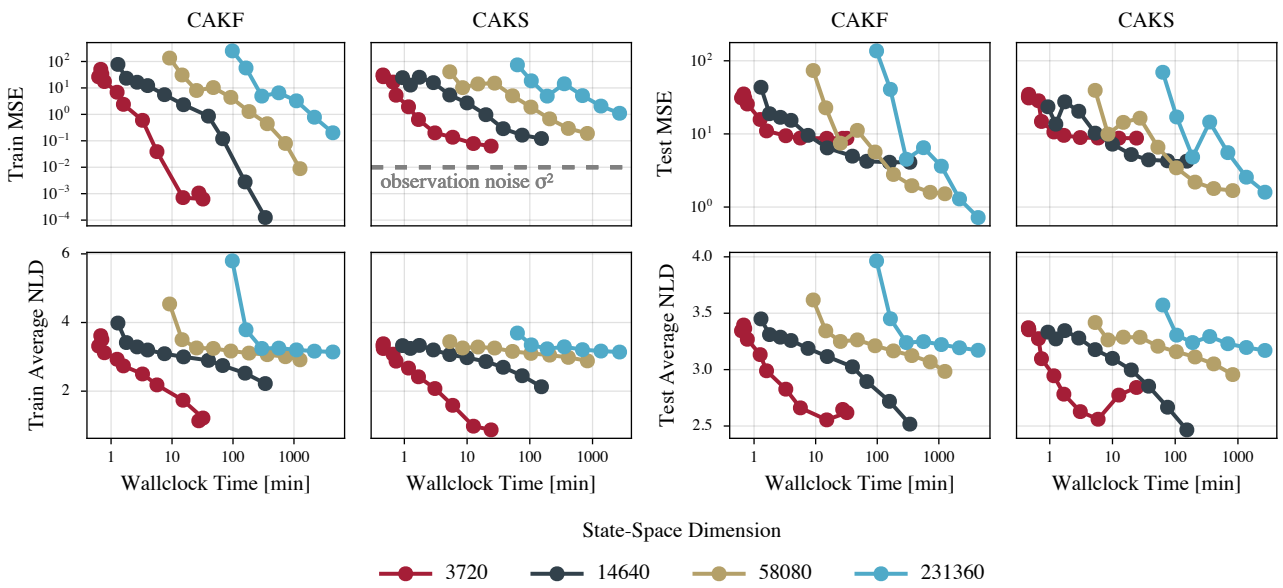


Figure 4: *Work-precision diagrams for the CAKF and CAKS on the ERA5 climate dataset.* The plot shows the mean squared error (MSE) and average negative log density (NLD) of the computation-aware filter and smoother for different problem sizes (i.e., state-space dimension) and number of iterations on the train and test set. The predictive error measured by test MSE *decreases* with larger problem sizes, while the test NLD *increases*. This is because we limit the computational budget and thus run *fewer* iterations for larger problems, i.e., we trade reduced computation cost for increased uncertainty.

the CAKF and CAKS approach the standard Kalman filter and RTS smoother. A comparison to these methods on this problem is not practically realisable, and nor is comparison to ensemble Kalman filters; see Appendix D.4 for a detailed explanation. For the smallest problem, we use up to $\check{N}_k = 2^{10}$ actions, while for the largest problem, we use up to $\check{N}_k = 2^6$. The experimental results are visualized in a work-precision diagram in Figure 4.

Figure 1(b) was generated by running the CAKF and the CAKS with a spatial downsampling factor of 3, corresponding to a state-space dimension of 231 360 and $\approx 4\,000\,000$ total training data points. The number of actions is set to $\check{N}_k = 64$ and the maximal rank after truncation is set to $2\check{N}_k = 128 \geq r_k^+$.

Interpretation As we increase the number of actions, i.e., the computational budget, the MSE and average NLD improve for both the CAKF and CAKS. As the state-space dimension increases, inference becomes more computationally demanding and the CAKF and CAKS take longer to compute the posterior marginals. However, with more data, both improve their generalization performance as measured by the MSE. To stay within a fixed upper limit on the time and memory budget, we constrain the number of iterations $\check{N}_k$ more as the problem size increases, which results in an increase in average NLD. This is an example of the aforementioned trade-off between reduced computational resources and increased uncertainty.

8 CONCLUSION

Kalman filtering and smoothing enable efficient inference in linear-Gaussian state-space models from a set of noisy observations. However, in many practical applications such as spatiotemporal regression, the latent state is high-dimensional. This results in prohibitive computational demands. In this work, we introduced computation-aware versions of the Kalman filter and smoother, which significantly reduce the time and memory complexity, while quantifying their inevitable approximation error via an appropriate increase in predictive uncertainty. Our experiments show that the CAKF and CAKS significantly outperform their main competitors, ensemble Kalman filtering methods, already on problems with moderately large state-space dimension. This is mostly due to the fact that, unlike the ensemble methods, the CAKF and CAKS account for their inherent approximation error by design. Further the CAKF and CAKS scale to significantly larger problems as evidenced by our benchmark experiment on a real-world climate dataset. A natural next step is to extend our approach such that model selection via evidence maximization becomes possible. Since the CAKF and CAKS are performing exact inference in a modified linear Gaussian state-space model, this is in theory directly possible by exploiting known techniques for the vanilla filter and smoother (Sec. 16.3.2, Särkkä and Svensson, 2023), however, the need for truncation complicates this.

Acknowledgments

MP and PH gratefully acknowledge financial support by the European Research Council through ERC StG Action 757275 / PANAMA and ERC CoG Action 101123955 / ANUBIS; the DFG Cluster of Excellence "Machine Learning - New Perspectives for Science", EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); the DFG SPP 2298 (Project HE 7114/5-1); and the Carl Zeiss Foundation (project "Certification and Foundations of Safe Machine Learning Systems in Healthcare"); as well as funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting MP. JW was supported by the Gatsby Charitable Foundation (GAT3708), the Simons Foundation (542963), the NSF AI Institute for Artificial and Natural Intelligence (ARNI: NSF DBI 2229929) and the Kavli Foundation. JC is supported by EPSRC grant EP/Y001028/1.

References

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. In *International Conference on Machine Learning (ICML)*, 2019. doi:10.48550/arXiv.1811.04551. URL http://arxiv.org/abs/1811.04551.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, 2023. URL http://arxiv.org/abs/2312.00752.

Simo Särkkä and Lennart Svensson. *Bayesian Filtering and Smoothing*, volume 17. Cambridge University Press, 2nd edition, 2023. ISBN 978-1-108-91230-3.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Geir Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162, 1994. doi:10.1029/94JC00572.

Johnathan M. Bardsley, Albert Parker, Antti Solonen, and Marylesa Howard. Krylov space approximate Kalman filtering. *Numerical Linear Algebra with Applications*, 20(2):171–184, December 2011. ISSN 1099-1506. doi:10.1002/nla.805. URL http://dx.doi.org/10.1002/nla.805.

Dimitris Berberidis and Georgios B. Giannakis. Data sketching for large-scale Kalman filtering. *IEEE Transactions on Signal Processing*, 65(14):3688–3701, July 2017. ISSN 1941-0476. doi:10.1109/tsp.2017.2691662. URL http://dx.doi.org/10.1109/tsp.2017.2691662.

Jonathan Schmidt, Philipp Hennig, Jörg Nick, and Filip Tronarp. The Rank-Reduced Kalman Filter: Approximate Dynamical-Low-Rank Filtering In High Dimensions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. doi:10.48550/arXiv.2306.07774. URL http://arxiv.org/abs/2306.07774.

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean Bidlot, Massimo Bonavita, Giovanna De Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elías Hólm, Marta Janisková, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia de Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean-Noël Thépaut. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020. ISSN 1477-870X. doi:10.1002/qj.3803. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/qj.3803.

Jouni Hartikainen and Simo Särkkä. Kalman filtering and smoothing solutions to temporal Gaussian process regression models. In *IEEE International Workshop on Machine Learning for Signal Processing*, pages 379–384, 2010. doi:10.1109/MLSP.2010.5589113.

Simo Särkkä, Arno Solin, and Jouni Hartikainen. Spatiotemporal Learning via Infinite-Dimensional Bayesian Filtering and Smoothing: A Look at Gaussian Process Regression Through Kalman Filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, 2013. ISSN 1558-0792. doi:10.1109/MSP.2013.2246292.

Jonathan Wenger, Geoff Pleiss, Marvin Pförtner, Philipp Hennig, and John P. Cunningham. Posterior and computational uncertainty in Gaussian processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Jon Cockayne, Matthew M. Graham, Chris J. Oates, T. J. Sullivan, and Onur Teymur. Testing whether a learning procedure is calibrated. *Journal of Machine Learning Research*, 23(203):1–36, 2022. URL http://jmlr.org/papers/v23/21-1065.html.

L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, 11(1):50–59, 1960. ISSN 1464-3847. doi:10.1093/qmath/11.1.50. URL http://dx.doi.org/10.1093/qmath/11.1.50.

Jon Cockayne, Chris J. Oates, Ilse C.F. Ipsen, and Mark Girolami. A Bayesian conjugate gradient method (with discussion). *Bayesian Analysis*, 14(3), September 2019a. ISSN 1936-0975. doi:10.1214/19-ba1145. URL http://dx.doi.org/10.1214/19-BA1145.

Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K. Sriperumbudur. Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences, July 2018. URL http://arxiv.org/abs/1807.02582.

Antti Solonen, Tiangang Cui, Janne Hakkarainen, and Youssef Marzouk. On dimension reduction in Gaussian filters. *Inverse Problems*, 32(4):045003, March 2016. ISSN 1361-6420. doi:10.1088/0266-5611/32/4/045003. URL http://dx.doi.org/10.1088/0266-5611/32/4/045003.

Marco Todescato, Andrea Carron, Ruggero Carli, Gianluigi Pillonetto, and Luca Schenato. Efficient spatio-temporal Gaussian regression via Kalman filtering. *Automatica*, 118, 2020. ISSN 0005-1098. doi:10.1016/j.automatica.2020.109032.

Simo Särkkä and Jouni Hartikainen. Infinite-Dimensional Kalman Filtering Approach to Spatio-Temporal Gaussian Process Regression. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 993–1001, 2012. URL https://proceedings.mlr.press/v22/sarkka12.html.

Arno Solin and Simo Särkkä. Infinite-dimensional Bayesian filtering for detection of quasi-periodic phenomena in spatio-temporal data. *Physical Review E*, 88(5), November 2013. ISSN 1539-3755, 1550-2376. doi:10.1103/PhysRevE.88.052909.

Philipp Hennig, Mike A. Osborne, and Mark Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2179), 2015.

Jon Cockayne, Chris Oates, TJ Sullivan, and Mark Girolami. Bayesian probabilistic numerical methods. *SIAM Review*, 61(4):756–789, 2019b.

Philipp Hennig, Michael A. Osborne, and Hans P. Kersting. *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022. ISBN 978-1-316-68141-1. doi:10.1017/9781316681411.

Philipp Hennig. Probabilistic interpretation of linear solvers. *SIAM Journal on Optimization*, 25(1):234–260, January 2015. ISSN 1095-7189. doi:10.1137/140955501. URL http://dx.doi.org/10.1137/140955501.

Jonathan Wenger and Philipp Hennig. Probabilistic linear solvers for machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://github.com/JonathanWenger/probabilistic-linear-solvers-for-ml.

Lukas Tatzel, Jonathan Wenger, Frank Schneider, and Philipp Hennig. Accelerating Generalized Linear Models by Trading off Computation for Uncertainty, 2023. URL http://arxiv.org/abs/2310.20285. arXiv:2310.20285 [cs, stat].

Craig H Bishop, Brian J Etherton, and Sharanya J Majumdar. Adaptive sampling with the ensemble transform Kalman filter. Part I: Theoretical aspects. *Monthly weather review*, 129(3):420–436, 2001.

Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963. Publisher: Society of Economic Geologists.

James T Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Deisenroth. Efficiently sampling functions from Gaussian process posteriors. In *International Conference on Machine Learning (ICML)*, 2020.

Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2007.

Geoff Pleiss, Martin Jankowiak, David Eriksson, Anil Damle, and Jacob Gardner. Fast matrix square roots with applications to Gaussian processes and Bayesian optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 22268–22281. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fcf55a303b71b84d326fb1d06e332a26-Paper.pdf.

Arno Solin. *Stochastic differential equation methods for spatio-temporal Gaussian process regression*. PhD thesis, Aalto University, 2016.

Oliver Hamelijnck, William J. Wilkinson, Niki A. Loppi, Arno Solin, and Theodoros Damoulas. Spatio-Temporal Variational Gaussian Processes, 2021. URL http://arxiv.org/abs/2111.01732. arXiv:2111.01732 [cs, stat].

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

Jean-François Le Gall. *Brownian Motion, Martingales, and Stochastic Calculus*, volume 274 of *Graduate Texts in Mathematics*. Springer International Publishing, Cham, 2016. doi:10.1007/978-3-319-31089-3.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

Simo Särkkä. *Recursive Bayesian Inference on Stochastic Differential Equations*. PhD thesis, Helsinki University of Technology, 2006.

Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, 1 edition, 2019. ISBN 978-1-108-18673-5. doi:10.1017/9781108186735.

Per-Gunnar Martinsson and Joel A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, May 2020. ISSN 1474-0508. doi:10.1017/s0962492920000021. URL `http://dx.doi.org/10.1017/S0962492920000021`.

James O. Berger. *Statistical Decision Theory*. Springer New York, 1980. ISBN 9781475717273. doi:10.1007/978-1-4757-1727-3. URL `http://dx.doi.org/10.1007/978-1-4757-1727-3`.

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, January 2003. ISBN 9780898718003. doi:10.1137/1.9780898718003. URL `http://dx.doi.org/10.1137/1.9780898718003`.

Jorg Liesen and Zdenek Strakos. *Krylov subspace methods*. Numerical Mathematics and Scientific Computation. Oxford University Press, London, England, December 2012.

Jeffrey L Anderson. An ensemble adjustment Kalman filter for data assimilation. *Monthly weather review*, 129(12):2884–2903, 2001.

Gerrit Burgers, Peter Jan van Leeuwen, and Geir Evensen. Analysis scheme in the ensemble kalman filter. *Monthly weather review*, 126(6):1719–1724, 1998.

Michael K Tippett, Jeffrey L Anderson, Craig H Bishop, Thomas M Hamill, and Jeffrey S Whitaker. Ensemble square root filters. *Monthly weather review*, 131(7): 1485–1490, 2003.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes. For instance, see Sections 2 to 5 and 7 and Appendices B to D.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes, see Section 5.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes, see Section 7.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes, see Section 5 and Appendices B and C.
   (b) Complete proofs of all theoretical results. Yes, see Appendices B and C.
   (c) Clear explanations of any assumptions. Not Applicable.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes, see Section 7.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes, see Section 7 and Appendix D.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes, see Appendices D.1 and D.2.1.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes, see Section 7.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes, see Section 7.
   (b) The license information of the assets, if applicable. Yes, see `https://apps.ecmwf.int/datasets/licences/copernicus/`.
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable.
   (d) Information about consent from data providers/curators. Not Applicable.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable.

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable.

Computation-Aware Kalman Filtering and Smoothing: Supplementary Materials

The supplementary materials contain derivations for our theoretical framework and proofs for the mathematical statements in the main text. We also provide implementation specifics and describe our experimental setup in more detail.

A NOTATION

Dynamics Model

D	Dimension of the state space	2, 18
K	Total number of states	2, 18
$\mathbf{u}_k$	Unobserved state at time step k	2, 18
$\boldsymbol{\mu}_k$	Prior mean of the state at time step k	2
$\boldsymbol{\Sigma}_k$	Prior covariance of the state at time step k	2
$\boldsymbol{A}_k$	Transition matrix from time step k to time step $k+1$	2, 18
$\boldsymbol{b}_k$	Transition offset from time step k to time step $k+1$	2, 18
$\mathbf{q}_k$	Process noise from time step k to time step $k+1$	2, 18
$\boldsymbol{Q}_k$	Process noise covariance matrix from time step k to time step $k+1$	2, 18

Observation Model

N_k	Number of observations (dimension of the observation vector) at time step k	2, 18
$\boldsymbol{y}_k$	Observation vector at time step k	2, 18
$\mathbf{y}_k$	Random variable encoding the belief about the observations at time step k	2, 18
$\boldsymbol{H}_k$	Observation matrix at time step k	2, 18
$\boldsymbol{\epsilon}_k$	Observation noise at time step k	2, 18
$\boldsymbol{\Lambda}_k$	Observation noise covariance matrix at time step k	2, 18

Kalman Filter

$\boldsymbol{m}_k^-$	Predictive filter mean at time step k	2, 18
$\boldsymbol{M}_k^-$	Left square root of the downdate term in the predictive filter covariance at time step k	19
$\boldsymbol{P}_k^-$	Predictive filter covariance at time step k	2, 18
$\boldsymbol{m}_k$	Updated filter mean at time step k	2, 18
$\boldsymbol{M}_k$	Left square root of the downdate term in the updated filter covariance at time step k	19
$\boldsymbol{P}_k$	Updated filter covariance at time step k	2, 18
$\boldsymbol{r}_k$	Prediction-measurement residual at time step k	19
$\boldsymbol{G}_k$	Innovation matrix at time step k	2, 19
$\boldsymbol{V}_k$	Left square root of the inverse innovation matrix at time step k	19
$\boldsymbol{W}_k$	Filter "covariance message" propagated from time step k to time step $k+1$ in the inverse-free RTS smoother	19
$\boldsymbol{K}_k$	Kalman gain at time step k	19

Computation-Aware Kalman Filter

$\hat{(\cdot)}$	"Hatted" quantities are associated with the CAKF/CAKS. Typically, these are counterparts of quantities in the standard Kalman filter / RTS smoother.	
$\hat{\boldsymbol{m}}_k^-$	Predictive CAKF mean at time step k	3
$\hat{\boldsymbol{M}}_k^-$	Left square root of the low-rank downdate representing the predictive CAKF covariance at time step k	3
$\hat{\boldsymbol{P}}_k^-$	Predictive CAKF covariance at time step k	3
$\hat{\boldsymbol{m}}_k$	Updated CAKF mean at time step k	3
$\hat{\boldsymbol{M}}_k$	Left square root of the low-rank downdate representing the updated CAKF covariance at time step k	3
r_k	Rank of the downdate representing the updated CAKF covariance at time step k	4
$\hat{\boldsymbol{P}}_k$	Updated CAKF covariance at time step k	3

$\check{N}_k$	Dimension of the projected observation vector (or equivalently number of actions) at time step k	3
$\hat{\boldsymbol{S}}_k$	Action matrix whose columns (the actions) span the low-dimensional subspace onto which the CAKF projects the observation at time step k	3
$\hat{\boldsymbol{s}}_k^{(i)}$	i-th action at time step k given by the i-th column of $\hat{\boldsymbol{S}}_k$	4
$\check{\boldsymbol{y}}_k$	Projected observation vector at time step k	3
$\check{\mathbf{y}}_k$	Random variable modeling the belief about the projected observation vector at time step k	3
$\check{\boldsymbol{H}}_k$	Projected observation matrix at time step k	3
$\check{\boldsymbol{\epsilon}}_k$	Projected observation noise at time step k	3
$\check{\boldsymbol{\Lambda}}_k$	Projected observation noise covariance matrix at time step k	3
$\check{\boldsymbol{G}}_k$	Projected innovation matrix at time step k	3
$\check{\boldsymbol{V}}_k$	Left square root of the projected inverse innovation matrix at time step k	3
$\hat{\boldsymbol{w}}_k$	CAKF "mean message" propagated from time step k to time step $k+1$ in the CAKS	4, 25
$\hat{\boldsymbol{W}}_k$	CAKF "covariance message" propagated from time step k to time step $k+1$ in the CAKS	4, 25
$\hat{\boldsymbol{M}}_k^+$	Left square root of the truncated low-rank downdate defining the truncated CAKF covariance at time step k	4
r_k^+	Rank of the truncated downdate defining the truncated CAKF covariance at time step k	4
$\hat{\boldsymbol{P}}_k^+$	Truncated CAKF covariance at time step k	4
$\mathbf{u}_k^+$	Additional state in the augmented state-space model of the CAKF modeling computational noise due to downdate truncation "infinitesimally after" time step k	4

Rauch–Tung–Striebel Smoother

$\boldsymbol{m}_k^{\mathrm{s}}$	Smoother mean at time step k	2, 19
$\boldsymbol{P}_k^{\mathrm{s}}$	Smoother covariance at time step k	2, 19
$\boldsymbol{w}_k^{\mathrm{s}}$	Smoother "mean message" propagated from time step k to time step $k-1$ in the inverse-free RTS smoother	20
$\boldsymbol{W}_k^{\mathrm{s}}$	Smoother "covariance message" propagated from time step k to time step $k-1$ in the inverse-free RTS smoother	20
$\boldsymbol{K}_k^{\mathrm{s}}$	Smoother gain at time step k	2, 19

Computation-Aware Rauch–Tung–Striebel Smoother

$\hat{\boldsymbol{m}}_k^{\mathrm{s}}$	CAKS mean at time step k	5
$\hat{\boldsymbol{M}}_k^{\mathrm{s}}$	Left square root of the low-rank downdate representing the CAKS covariance at time step k	5
$\hat{\boldsymbol{P}}_k^{\mathrm{s}}$	CAKS covariance at time step k	
$\hat{\boldsymbol{w}}_k^{\mathrm{s}}$	CAKS "mean message" propagated from time step k to time step $k-1$	5
$\hat{\boldsymbol{W}}_k^{\mathrm{s}}$	(Truncated) CAKS "covariance message" propagated from time step k to time step $k-1$	5

Matheron Sampling

$\mathbf{u}_k^{f-}$	Random variable with distribution $\mathcal{N}\left(\boldsymbol{m}_k^-, \boldsymbol{P}_k^-\right)$	21
$\mathbf{y}_k^{f-}$	Random variable with distribution $\mathcal{N}\left(\boldsymbol{H}_k \boldsymbol{m}_k^-, \boldsymbol{G}_k\right)$	21
$\mathbf{u}_k^f$	Random variable with distribution $\mathcal{N}\left(\boldsymbol{m}_k, \boldsymbol{P}_k\right)$	21
$\mathbf{w}_k^{\mathrm{s}}$	Random variable propagating the "smoother message" from time step k to time step $k-1$ during inverse-free posterior sampling	22
$\mathbf{u}_k^{\mathrm{s}}$	Random variable with distribution $\mathcal{N}\left(\boldsymbol{m}_k^{\mathrm{s}}, \boldsymbol{P}_k^{\mathrm{s}}\right)$	21

Computation-Aware Matheron Sampling

$\hat{\mathbf{u}}_k^{f-}$	Random variable with distribution $\mathcal{N}\left(\hat{\boldsymbol{m}}_k^-, \hat{\boldsymbol{P}}_k^-\right)$	23
$\hat{\mathbf{w}}_k$	Random variable propagating the "filter message" from time step k to time step $k+1$ during CAKF/CAKS sampling	23
$\hat{\mathbf{u}}_k^{f}$	Random variable with distribution $\mathcal{N}\left(\hat{\boldsymbol{m}}_k, \hat{\boldsymbol{P}}_k\right)$	23
$\hat{\mathbf{w}}_k^{\mathrm{s}}$	Random variable propagating the "smoother message" from time step k to time step $k-1$ during CAKS sampling	23
$\hat{\mathbf{u}}_k^{\mathrm{s}}$	Random variable with distribution $\mathcal{N}\left(\hat{\boldsymbol{m}}_k^{\mathrm{s}}, \hat{\boldsymbol{P}}_k^{\mathrm{s}}\right)$	23

Spatiotemporal GP Regression

$[t_0, T]$	Temporal domain	3, 26
$\mathbb{X}$	Spatial domain	3, 26
$\mathbb{Z}$	Input domain $\mathbb{Z} = [t_0, T] \times \mathbb{X}$	26
$f^\star$	Unknown target function	3, 26
$y^\star$	Noisy observed function	6, 31
t_k^{train}	k-th time step at which training data is available	3
$\boldsymbol{X}_k^{\mathrm{train}}$	Vector of spatial points $\boldsymbol{x}_{k,n}^{\mathrm{train}}$ at which training data is available at time step t_k^{train}	3
$\boldsymbol{Z}^{\mathrm{train}}$	Vector of ordered pairs $(t_k^{\mathrm{train}}, \boldsymbol{x}_{k,n}^{\mathrm{train}})$ of training input points	6, 31
$\boldsymbol{y}^{\mathrm{train}}$	Vector containing all training targets	6, 31
N	Total number of training data points	6, 31
$\boldsymbol{X}$	Vector containing all (unique) spatial training and test points from all time steps	3
$N_{\boldsymbol{X}}$	Total number of unique spatial training and test points over all time steps	3
f	Spatiotemporal Gaussian process prior for the unknown target function $f^\star$	3, 26
μ	Mean function of f	3, 26
Σ	Covariance function of f	3, 26
$\mathbb{H}_\Sigma$	Reproducing kernel Hilbert space (RKHS) associated with Σ	6, 31
σ	Observation noise scale	6, 31
Σ^σ	Covariance function of the noisy observed process	6, 31
$\bar{\mu}^{y^\star}$	Posterior mean function corresponding to the observed function $y^\star$	6
$\bar{\Sigma}$	Posterior covariance function	6

Space-Time Separable Gauss(–Markov) Processes

$\mathbf{f}$	Space-time separable Gaussian (or Gauss–Markov) process (STSG(M)P). Typically obtained by combining f and $D'-1$ of its time derivatives in a multi-output GP	3, 26
D'	Output dimension of the STSG(M)P	3, 26
$\boldsymbol{\mu}$	Mean function of the STSG(M)P	26
$\boldsymbol{\mu}^t$	Temporal factor of the mean function of the STSG(M)P	26
$\mu^{\boldsymbol{x}}$	Spatial factor of the mean function of the STSG(M)P	26
$\boldsymbol{\Sigma}$	Covariance function of the STSG(M)P	26
$\boldsymbol{\Sigma}^t$	Temporal factor of the covariance function of the STSG(M)P	26
$\Sigma^{\boldsymbol{x}}$	Spatial factor of the covariance function of the STSG(M)P	26

Iteratively-Approximated Gaussian Process Regression

$\boldsymbol{S}$	Matrix of actions	29
$\check{N}$	Number of actions	29
$\check{\boldsymbol{y}}^{\mathrm{train}}$	Projected training targets	29

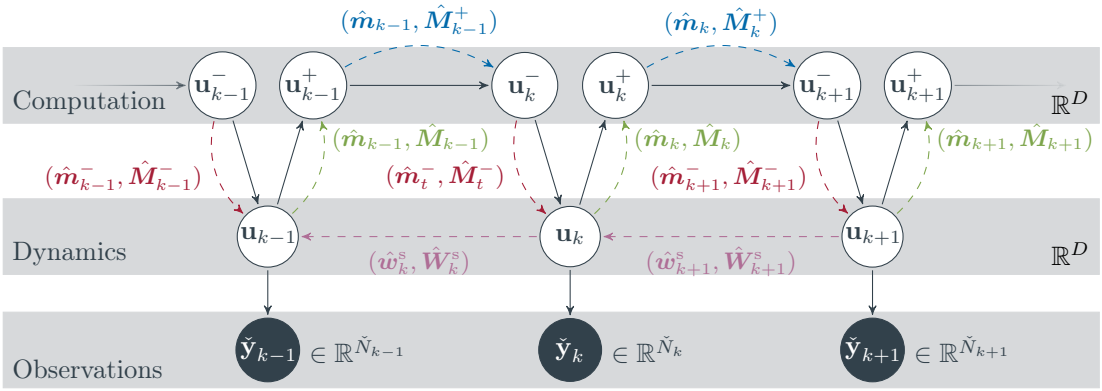


Figure B.1: Probabilistic graphical model for the computation-aware Kalman filter and RTS smoother. Solid arrows and circles define the joint generative model (i.e., the posterior computed by filter and smoother). Dashed arrows visualize the information flow between nodes, with the corresponding "messages" in parentheses.

B DERIVATION OF THE ALGORITHM

Definition B.1 (Linear-Gaussian State-Space Model)**.** A *linear-Gaussian state-space model* (LGSSM) is a pair $(\{\mathbf{u}_k\}_{k=0}^K, \{\mathbf{y}_k\}_{k=1}^K)$ of discrete-time stochastic processes defined by

$$\begin{aligned}\mathbf{u}_k &:= \boldsymbol{A}_{k-1}\mathbf{u}_{k-1} + \boldsymbol{b}_{k-1} + \mathbf{q}_{k-1} \in \mathbb{R}^D,\\ \mathbf{y}_k &:= \boldsymbol{H}_k\mathbf{u}_k + \boldsymbol{c}_k + \boldsymbol{\epsilon}_k \in \mathbb{R}^{N_k},\end{aligned}$$

where

$$\begin{aligned}\mathbf{u}_0 &\sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)\\ \mathbf{q}_{k-1} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_{k-1})\\ \boldsymbol{\epsilon}_k &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_k)\end{aligned}$$

are pairwise independent.

B.1 Filtering

Theorem B.2 (Kalman Filter)**.** *Let* $(\mathbf{u}, \mathbf{y})$ *be the LGSSM from Definition B.1. Fix* $\{\boldsymbol{y}_k\}_{k=1}^K$ *with* $\boldsymbol{y}_k \in \mathbb{R}^{N_k}$*. Then*

$$\mathbf{u}_k \,|\, \mathbf{y}_{1:k-1} = \boldsymbol{y}_{1:k-1} \sim \mathcal{N}\left(\boldsymbol{m}_k^-, \boldsymbol{P}_k^-\right),$$

where

$$\begin{aligned}\boldsymbol{m}_k^- &:= \boldsymbol{A}_{k-1}\boldsymbol{m}_{k-1} + \boldsymbol{b}_{k-1},\\ \boldsymbol{P}_k^- &:= \boldsymbol{A}_{k-1}\boldsymbol{P}_{k-1}\boldsymbol{A}_{k-1}^\top + \boldsymbol{Q}_{k-1},\end{aligned}$$

and

$$\mathbf{u}_k \,|\, \mathbf{y}_{1:k} = \boldsymbol{y}_{1:k} \sim \mathcal{N}(\boldsymbol{m}_k, \boldsymbol{P}_k),$$

where $\boldsymbol{m}_0 = \boldsymbol{\mu}_0$*,* $\boldsymbol{P}_0 = \boldsymbol{\Sigma}_0$*, and*

$$\begin{aligned}\boldsymbol{m}_k &:= \boldsymbol{m}_k^- + \boldsymbol{K}_k\boldsymbol{r}_k,\\ \boldsymbol{P}_k &:= \boldsymbol{P}_k^- - \boldsymbol{K}_k\boldsymbol{G}_k\boldsymbol{K}_k^\top,\end{aligned}$$

for $k = 1, \dots, K$ with

$$\begin{aligned} \boldsymbol{r}_k &\coloneqq \boldsymbol{y}_k - \boldsymbol{H}_k \boldsymbol{m}_k^- - \boldsymbol{c}_k, \\ \boldsymbol{G}_k &\coloneqq \boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^\top + \boldsymbol{\Lambda}_k, \\ \boldsymbol{K}_k &\coloneqq \boldsymbol{P}_k^- \boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1}. \end{aligned}$$

Proposition B.3 (Downdate-Form Kalman Filter)**.** *The Kalman state covariances can equivalently be computed via*

$$\begin{aligned} \boldsymbol{P}_k^- &= \boldsymbol{\Sigma}_k - \boldsymbol{M}_k^- (\boldsymbol{M}_k^-)^\top, \\ \boldsymbol{P}_k &= \boldsymbol{\Sigma}_k - \boldsymbol{M}_k \boldsymbol{M}_k^\top, \end{aligned}$$

where $\boldsymbol{M}_0 \coloneqq \big() \in \mathbb{R}^{D \times 0}$ and

$$\begin{aligned} \boldsymbol{M}_k^- &\coloneqq \boldsymbol{A}_{k-1} \boldsymbol{M}_{k-1}, \\ \boldsymbol{M}_k &\coloneqq \begin{pmatrix} \boldsymbol{M}_k^- & \boldsymbol{P}_k^- \boldsymbol{W}_k \end{pmatrix} \end{aligned}$$

with $\boldsymbol{W}_k \coloneqq \boldsymbol{H}_k^\top \boldsymbol{V}_k$, and $\boldsymbol{V}_k \boldsymbol{V}_k^\top = \boldsymbol{G}_k^{-1}$ for $k = 1, \dots, K$.

Proof. For $k = 0$, we find that

$$\boldsymbol{P}_0 = \boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_0 - \boldsymbol{0}_{D \times D} = \boldsymbol{\Sigma}_0 - \boldsymbol{M}_0 \boldsymbol{M}_0^\top.$$

Now let $1 \leq k \leq K$ and assume that the statement holds for $k - 1$. Then

$$\begin{aligned} \boldsymbol{P}_k^- &= \boldsymbol{A}_{k-1} \boldsymbol{P}_{k-1} \boldsymbol{A}_{k-1}^\top + \boldsymbol{Q}_{k-1} \\ &= \boldsymbol{A}_{k-1} \boldsymbol{\Sigma}_{k-1} \boldsymbol{A}_{k-1}^\top + \boldsymbol{Q}_{k-1} - \boldsymbol{A}_{k-1} \boldsymbol{M}_{k-1} \boldsymbol{M}_{k-1}^\top \boldsymbol{A}_{k-1}^\top \\ &= \boldsymbol{\Sigma}_k - \boldsymbol{A}_{k-1} \boldsymbol{M}_{k-1} (\boldsymbol{A}_{k-1} \boldsymbol{M}_{k-1})^\top \\ &= \boldsymbol{\Sigma}_k - \boldsymbol{M}_k^- (\boldsymbol{M}_k^-)^\top, \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{P}_k &= \boldsymbol{P}_k^- - \boldsymbol{P}_k^- \boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1} \boldsymbol{H}_k \boldsymbol{P}_k^- \\ &= \boldsymbol{\Sigma}_k - \boldsymbol{M}_k^- (\boldsymbol{M}_k^-)^\top - \boldsymbol{P}_k^- \boldsymbol{H}_k^\top \boldsymbol{V}_k \boldsymbol{V}_k^\top \boldsymbol{H}_k \boldsymbol{P}_k^- \\ &= \boldsymbol{\Sigma}_k - \begin{pmatrix} \boldsymbol{M}_k^- & \boldsymbol{P}_k^- \boldsymbol{H}_k^\top \boldsymbol{V}_k \end{pmatrix} \begin{pmatrix} \boldsymbol{M}_k^- & \boldsymbol{P}_k^- \boldsymbol{H}_k^\top \boldsymbol{V}_k \end{pmatrix}^\top \\ &= \boldsymbol{\Sigma}_k - \boldsymbol{M}_k \boldsymbol{M}_k^\top. \end{aligned}$$

□

B.2 Smoothing

Theorem B.4 (RTS Smoother)**.** *Let $(\mathbf{u}, \mathbf{y})$ be the LGSSM from Definition B.1. Then*

$$\mathbf{u}_k \mid \mathbf{y}_{1:K} = \boldsymbol{y}_{1:K} \sim \mathcal{N}(\boldsymbol{m}_k^\mathrm{s}, \boldsymbol{P}_k^\mathrm{s}),$$

where $\boldsymbol{m}_K^\mathrm{s} = \boldsymbol{m}_K$, $\boldsymbol{P}_K^\mathrm{s} = \boldsymbol{P}_K$, and

$$\begin{aligned} \boldsymbol{m}_k^\mathrm{s} &\coloneqq \boldsymbol{m}_k + \boldsymbol{K}_k^\mathrm{s} (\boldsymbol{m}_{k+1}^\mathrm{s} - \boldsymbol{m}_{k+1}^-) \\ \boldsymbol{P}_k^\mathrm{s} &\coloneqq \boldsymbol{P}_k + \boldsymbol{K}_k^\mathrm{s} (\boldsymbol{P}_{k+1}^\mathrm{s} - \boldsymbol{P}_{k+1}^-) (\boldsymbol{K}_k^\mathrm{s})^\top \end{aligned}$$

for $k = 1, \dots, K-1$ with $\boldsymbol{K}_k^\mathrm{s} \coloneqq \boldsymbol{P}_k \boldsymbol{A}_k^\top (\boldsymbol{P}_{k+1}^-)^{-1}$.

Proposition B.5 (Inverse-Free RTS Smoother)**.** *The RTS smoother moments can be equivalently computed by the recursion*

$$\boldsymbol{m}_k^\mathrm{s} = \boldsymbol{m}_k^- + \boldsymbol{P}_k^- \boldsymbol{w}_k^\mathrm{s}$$

$$\boldsymbol{P}_k^{\mathrm{s}} = \boldsymbol{P}_k^- - \boldsymbol{P}_k^- \boldsymbol{W}_k^{\mathrm{s}} (\boldsymbol{P}_k^- \boldsymbol{W}_k^{\mathrm{s}})^\top,$$

where $\boldsymbol{w}_K^{\mathrm{s}} = \boldsymbol{H}_K^\top \boldsymbol{G}_K^{-1} \boldsymbol{r}_K$, $\boldsymbol{W}_K^{\mathrm{s}} (\boldsymbol{W}_K^{\mathrm{s}})^\top = \boldsymbol{H}_K^\top \boldsymbol{G}_K^{-1} \boldsymbol{H}_K$, *and*

$$\begin{aligned}
\boldsymbol{w}_k^{\mathrm{s}} &= \boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1} \boldsymbol{r}_k + (\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \boldsymbol{w}_{k+1}^{\mathrm{s}} \\
&= \boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1} \boldsymbol{r}_k + (\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1} \boldsymbol{H}_k \boldsymbol{P}_k^-) \boldsymbol{A}_k^\top \boldsymbol{w}_{k+1}^{\mathrm{s}} && \text{(B.1)} \\
\boldsymbol{W}_k^{\mathrm{s}} (\boldsymbol{W}_k^{\mathrm{s}})^\top &= \boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1} \boldsymbol{H}_k + (\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}} \big((\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}} \big)^\top \\
&= \boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1} \boldsymbol{H}_k + (\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1} \boldsymbol{H}_k \boldsymbol{P}_k^-) \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}} \big((\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1} \boldsymbol{H}_k \boldsymbol{P}_k^-) \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}} \big)^\top && \text{(B.2)}
\end{aligned}$$

for $k = 1, \ldots, K-1$. *Moreover,*

$$\begin{aligned}
\boldsymbol{m}_k^{\mathrm{s}} &= \boldsymbol{m}_k + \boldsymbol{P}_k \boldsymbol{A}_k^\top \boldsymbol{w}_{k+1}^{\mathrm{s}}, && \text{(B.3)} \\
\boldsymbol{P}_k^{\mathrm{s}} &= \boldsymbol{P}_k - \boldsymbol{P}_k \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}} (\boldsymbol{P}_k \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}})^\top && \text{(B.4)}
\end{aligned}$$

for $k = 1, \ldots, K-1$.

Proof. For $k = K$, we have

$$\boldsymbol{m}_K^{\mathrm{s}} = \boldsymbol{m}_K = \boldsymbol{m}_K^- + \boldsymbol{K}_K \boldsymbol{r}_K = \boldsymbol{m}_K^- + \boldsymbol{P}_K^- \underbrace{\boldsymbol{H}_K^\top \boldsymbol{G}_K^{-1} \boldsymbol{r}_K}_{=\boldsymbol{w}_K^{\mathrm{s}}}$$

and

$$\boldsymbol{P}_K^{\mathrm{s}} = \boldsymbol{P}_K = \boldsymbol{P}_K^- - \boldsymbol{K}_K \boldsymbol{G}_K \boldsymbol{K}_K^\top = \boldsymbol{P}_K^- - \boldsymbol{P}_K^- \underbrace{\boldsymbol{H}_K^\top \boldsymbol{G}_K^{-1} \boldsymbol{G}_K \boldsymbol{G}_K^{-1} \boldsymbol{H}_K}_{=\boldsymbol{W}_K^{\mathrm{s}} (\boldsymbol{W}_K^{\mathrm{s}})^\top} \boldsymbol{P}_K^-.$$

Now let $1 \leq k < K$ and assume that

$$\begin{aligned}
\boldsymbol{m}_{k+1}^{\mathrm{s}} = \boldsymbol{m}_{k+1}^- + \boldsymbol{P}_{k+1}^- \boldsymbol{w}_{k+1}^{\mathrm{s}} \quad &\Leftrightarrow \quad \boldsymbol{m}_{k+1}^{\mathrm{s}} - \boldsymbol{m}_{k+1}^- = \boldsymbol{P}_{k+1}^- \boldsymbol{w}_{k+1}^{\mathrm{s}}, \\
\boldsymbol{P}_{k+1}^{\mathrm{s}} = \boldsymbol{P}_{k+1}^- - \boldsymbol{P}_{k+1}^- \boldsymbol{W}_{k+1}^{\mathrm{s}} (\boldsymbol{P}_{k+1}^- \boldsymbol{W}_{k+1}^{\mathrm{s}})^\top \quad &\Leftrightarrow \quad \boldsymbol{P}_{k+1}^{\mathrm{s}} - \boldsymbol{P}_{k+1}^- = -\boldsymbol{P}_{k+1}^- \boldsymbol{W}_{k+1}^{\mathrm{s}} (\boldsymbol{P}_{k+1}^- \boldsymbol{W}_{k+1}^{\mathrm{s}})^\top.
\end{aligned}$$

It follows that

$$\begin{aligned}
\boldsymbol{m}_k^{\mathrm{s}} &= \boldsymbol{m}_k + \boldsymbol{K}_k^{\mathrm{s}} (\boldsymbol{m}_{k+1}^{\mathrm{s}} - \boldsymbol{m}_{k+1}^-) \\
&= \boldsymbol{m}_k + \boldsymbol{K}_k^{\mathrm{s}} \boldsymbol{P}_{k+1}^- \boldsymbol{w}_{k+1}^{\mathrm{s}} \\
&= \boldsymbol{m}_k + \boldsymbol{P}_k \boldsymbol{A}_k^\top \boldsymbol{w}_{k+1}^{\mathrm{s}} \\
&= \boldsymbol{m}_k^- + \boldsymbol{K}_k \boldsymbol{r}_k + (\boldsymbol{P}_k^- - \boldsymbol{K}_k \boldsymbol{G}_k \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \boldsymbol{w}_{k+1}^{\mathrm{s}} \\
&= \boldsymbol{m}_k^- + \boldsymbol{P}_k^- \Big(\boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1} \boldsymbol{r}_k + (\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \boldsymbol{w}_{k+1}^{\mathrm{s}} \Big) \\
&= \boldsymbol{m}_k^- + \boldsymbol{P}_k^- \boldsymbol{w}_k^{\mathrm{s}}
\end{aligned}$$

and

$$\begin{aligned}
\boldsymbol{P}_k^{\mathrm{s}} &= \boldsymbol{P}_k + \boldsymbol{K}_k^{\mathrm{s}} (\boldsymbol{P}_{k+1}^{\mathrm{s}} - \boldsymbol{P}_{k+1}^-) (\boldsymbol{K}_k^{\mathrm{s}})^\top \\
&= \boldsymbol{P}_k - \boldsymbol{P}_k \boldsymbol{A}_k^\top (\boldsymbol{P}_{k+1}^-)^{-1} \boldsymbol{P}_{k+1}^- \boldsymbol{W}_{k+1}^{\mathrm{s}} (\boldsymbol{P}_{k+1}^- \boldsymbol{W}_{k+1}^{\mathrm{s}})^\top (\boldsymbol{P}_{k+1}^-)^{-1} \boldsymbol{A}_k \boldsymbol{P}_k \\
&= \boldsymbol{P}_k - \boldsymbol{P}_k \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}} \big(\boldsymbol{P}_k \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}} \big)^\top \\
&= \boldsymbol{P}_k^- - \boldsymbol{K}_k \boldsymbol{G}_k \boldsymbol{K}_k^\top - (\boldsymbol{P}_k^- - \boldsymbol{K}_k \boldsymbol{G}_k \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}} \big((\boldsymbol{P}_k^- - \boldsymbol{K}_k \boldsymbol{G}_k \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}} \big)^\top \\
&= \boldsymbol{P}_k^- - \boldsymbol{P}_k^- \Big(\boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1} \boldsymbol{H}_k + (\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}} \big((\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \boldsymbol{W}_{k+1}^{\mathrm{s}} \big)^\top \Big) \boldsymbol{P}_k^- \\
&= \boldsymbol{P}_k^- - \boldsymbol{P}_k^- \boldsymbol{W}_k \boldsymbol{P}_k^-.
\end{aligned}$$

□

B.3 Sampling via Matheron's Rule

The naive approach to sampling from a multivariate normal distribution (e.g., by Cholesky factorization or eigendecomposition) has cubic cost and requires storing the covariance matrix in memory, which is not possible for large state-space dimension. We alleviate this by applying Matheron's rule (Matheron, 1963; Wilson et al., 2020) to the (computation-aware) Kalman filter and RTS smoother recursions, making it possible to sample the filtering and smoothing posteriors by transforming samples from the prior.

Lemma B.6 (Matheron's Rule)**.** *Let* $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$*,* $\boldsymbol{A} \in \mathbb{R}^{N_{\boldsymbol{A}} \times D}$*,* $\boldsymbol{B} \in \mathbb{R}^{N_{\boldsymbol{B}} \times D}$*, and* $\boldsymbol{\beta} \in \operatorname{ran}(\boldsymbol{B}\boldsymbol{\Sigma})$*. Define*

$$\boldsymbol{\mathcal{M}}_{\boldsymbol{\Sigma},\boldsymbol{A},\boldsymbol{B},\boldsymbol{\beta}} \colon \mathbb{R}^D \to \mathbb{R}^{N_{\boldsymbol{A}}}, \boldsymbol{\xi} \mapsto \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\boldsymbol{\xi}, \boldsymbol{\Sigma})} \left[\boldsymbol{A}\tilde{\mathbf{x}} \mid \boldsymbol{B}\tilde{\mathbf{x}} = \boldsymbol{\beta} \right].$$

Then $(\boldsymbol{A}\mathbf{x} \mid \boldsymbol{B}\mathbf{x} = \boldsymbol{\beta}) \overset{d}{=} \boldsymbol{\mathcal{M}}_{\boldsymbol{\Sigma},\boldsymbol{A},\boldsymbol{B},\boldsymbol{\beta}}(\mathbf{x})$*.*

Proof. Let $\mathbf{a} \coloneqq \boldsymbol{A}\mathbf{x}$ and $\mathbf{b} \coloneqq \boldsymbol{B}\mathbf{x}$. Then

$$\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{A}\boldsymbol{\mu} \\ \boldsymbol{B}\boldsymbol{\mu} \end{pmatrix}, \begin{pmatrix} \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top & \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{B}^\top \\ \boldsymbol{B}\boldsymbol{\Sigma}\boldsymbol{A}^\top & \boldsymbol{B}\boldsymbol{\Sigma}\boldsymbol{B}^\top \end{pmatrix} \right)$$

and

$$\boldsymbol{\mathcal{M}}_{\boldsymbol{\Sigma},\boldsymbol{A},\boldsymbol{B},\boldsymbol{\beta}}(\boldsymbol{\xi}) = \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\boldsymbol{\xi}, \boldsymbol{\Sigma})} \left[\boldsymbol{A}\tilde{\mathbf{x}} \mid \boldsymbol{B}\tilde{\mathbf{x}} = \boldsymbol{\beta} \right] = \boldsymbol{A}\boldsymbol{\xi} + \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{B}^\top (\boldsymbol{B}\boldsymbol{\Sigma}\boldsymbol{B}^\top)^{-1} (\boldsymbol{\beta} - \boldsymbol{B}\boldsymbol{\xi}).$$

Hence,

$$\boldsymbol{\mathcal{M}}_{\boldsymbol{\Sigma},\boldsymbol{A},\boldsymbol{B},\boldsymbol{\beta}}(\mathbf{x}) = \underbrace{\boldsymbol{A}\mathbf{x}}_{=\mathbf{a}} + \underbrace{\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{B}^\top}_{=\operatorname{Cov}[\mathbf{a},\mathbf{b}]} (\underbrace{\boldsymbol{B}\boldsymbol{\Sigma}\boldsymbol{B}^\top}_{=\operatorname{Cov}[\mathbf{b},\mathbf{b}]})^{-1} (\boldsymbol{\beta} - \underbrace{\boldsymbol{B}\mathbf{x}}_{=\mathbf{b}})$$

and thus the statement follows from Wilson et al. (2020, Theorem 1). □

To proceed, we assume that it is feasible to obtain an (approximate) sample from the initial state $\mathbf{u}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$, as well as (approximate) samples from the dynamics and observational noise $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_{k-1})$, $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_k)$, $k = 1, \ldots, K$. This assumption is reasonable because the covariance matrices $\boldsymbol{\Sigma}_0, \boldsymbol{Q}_{k-1}, \boldsymbol{\Lambda}_k$ are often simple or highly structured; for example it is common for $\boldsymbol{\Lambda}_k$ to be diagonal. Moreover, for discretized spatiotemporal Gauss–Markov processes one can use function space approximations like random Fourier features (RFF) (Rahimi and Recht, 2007) to obtain approximate samples from $\mathbf{u}_0$ and $\mathbf{q}_{k-1}$ (see also Wilson et al., 2020). Finally, Krylov methods can be used to approximate matrix square roots of the covariances in a matrix-free fashion (see e.g., Pleiss et al., 2020). With these samples, Theorem B.7 shows how Matheron sampling can be implemented for the standard Kalman filter and RTS smoother, while Proposition B.8 gives an equivalent form of Matheron sampling for the smoother that circumvents inversion of state covariance matrices.

Each of these approaches can be applied to the modified state-space model used in the CAKF and the CAKS at low cost, again recycling computed values from the filtering pass in Algorithms 1 and 2. The resulting algorithm for sampling from the computation-aware posterior process $\{\mathbf{u}_k \mid \check{\mathbf{y}}_{1:K} = \check{\boldsymbol{y}}_{1:K}\}_{k=0}^K$ is detailed in Algorithm B.1. If it is stopped early before Line 9, then it can also be used to compute samples from the CAKF states $\{\mathbf{u}_k \mid \check{\mathbf{y}}_{1:k} = \check{\boldsymbol{y}}_{1:k}\}_{k=0}^K$. Also note that Algorithm B.1 allows us to sample from the full Bayesian posterior without running the CAKS, since all quantities used above have already been computed by the filter.

Theorem B.7. *Let* $(\mathbf{u}, \mathbf{y})$ *be the LGSSM from Definition B.1. Fix* $\boldsymbol{y}_k \in \mathbb{R}^{N_k}$ *for* $k = 1, \ldots, K$ *and define*

$$\begin{aligned} \mathbf{u}_k^{f-} &\coloneqq \boldsymbol{A}_{k-1}\mathbf{u}_{k-1}^f + \boldsymbol{b}_{k-1} + \mathbf{q}_{k-1} \\ \mathbf{y}_k^{f-} &\coloneqq \boldsymbol{H}_k\mathbf{u}_k^{f-} + \boldsymbol{c}_k + \boldsymbol{\epsilon}_k \\ \mathbf{u}_k^f &\coloneqq \mathbf{u}_k^{f-} + \boldsymbol{K}_k(\boldsymbol{y}_k - \mathbf{y}_k^{f-}) \end{aligned}$$

for $k = 1, \ldots, K$*, where* $\mathbf{u}_0^f \coloneqq \mathbf{u}_0$*, as well as*

$$\mathbf{u}_k^{\mathrm{s}} \coloneqq \mathbf{u}_k^f + \boldsymbol{K}_k^{\mathrm{s}}(\mathbf{u}_{k+1}^{\mathrm{s}} - \mathbf{u}_{k+1}^{f-})$$

for $k = K-1, \ldots, 0$*, where* $\mathbf{u}_K^{\mathrm{s}} \coloneqq \mathbf{u}_K^f$*. Then*

$$(\mathbf{u}_1^{\mathrm{s}}, \ldots, \mathbf{u}_K^{\mathrm{s}}) \overset{d}{=} \Big((\mathbf{u}_1, \ldots, \mathbf{u}_K) \mid \mathbf{y}_1 = \boldsymbol{y}_1, \ldots, \mathbf{y}_K = \boldsymbol{y}_K \Big).$$

Proof. Let $\tilde{\mathbf{q}}_k \coloneqq \mathbf{q}_k + \boldsymbol{b}_k$ and $\tilde{\boldsymbol{\epsilon}}_k \coloneqq \boldsymbol{\epsilon}_k + \boldsymbol{c}_k$. Then $\mathbf{x} \coloneqq (\mathbf{u}_0, \tilde{\mathbf{q}}_0, \dots, \tilde{\mathbf{q}}_{K-1}, \tilde{\boldsymbol{\epsilon}}_1, \dots, \tilde{\boldsymbol{\epsilon}}_K)$ is jointly Gaussian with pairwise independent components and covariance matrix $\tilde{\boldsymbol{\Sigma}}$.

$$\begin{aligned} \mathbf{u}_{k+1} &= \boldsymbol{A}_k \mathbf{u}_k + \tilde{\mathbf{q}}_k, \qquad \text{and} \\ \mathbf{y}_k &= \boldsymbol{H}_k \mathbf{u}_k + \tilde{\boldsymbol{\epsilon}}_k \end{aligned}$$

pointwise. Let $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{Y}}$ be the linear operators defined by

$$\begin{aligned} \boldsymbol{\mathcal{U}}(\mathbf{u}_0, \tilde{\mathbf{q}}_0, \dots, \tilde{\mathbf{q}}_{K-1}, \tilde{\boldsymbol{\epsilon}}_1, \dots, \tilde{\boldsymbol{\epsilon}}_K) &= (\mathbf{u}_1, \dots, \mathbf{u}_K), \qquad \text{and} \\ \boldsymbol{\mathcal{Y}}(\mathbf{u}_0, \tilde{\mathbf{q}}_0, \dots, \tilde{\mathbf{q}}_{K-1}, \tilde{\boldsymbol{\epsilon}}_1, \dots, \tilde{\boldsymbol{\epsilon}}_K) &= (\mathbf{y}_1, \dots, \mathbf{y}_K). \end{aligned}$$

By Theorems B.2 and B.4, the operator $\boldsymbol{\mathcal{M}}_{\tilde{\boldsymbol{\Sigma}}, \boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{Y}}, \boldsymbol{y}_{1:K}}$ from Lemma B.6 corresponding to this model is given by

$$\boldsymbol{\mathcal{M}}_{\tilde{\boldsymbol{\Sigma}}, \boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{Y}}, \boldsymbol{y}_{1:K}}(\boldsymbol{\xi}) = (\boldsymbol{m}^{\mathrm{s}}_1(\boldsymbol{\xi}), \dots, \boldsymbol{m}^{\mathrm{s}}_K(\boldsymbol{\xi})),$$

where, for $\boldsymbol{\xi} = (\boldsymbol{\xi}^{\mathbf{u}}_0, \boldsymbol{\xi}^{\tilde{\mathbf{q}}}_0, \dots, \boldsymbol{\xi}^{\tilde{\mathbf{q}}}_{K-1}, \boldsymbol{\xi}^{\tilde{\boldsymbol{\epsilon}}}_1, \dots, \boldsymbol{\xi}^{\tilde{\boldsymbol{\epsilon}}}_K)$,

$$\begin{aligned} \boldsymbol{m}_0(\boldsymbol{\xi}) &\coloneqq \boldsymbol{\xi}^{\mathbf{u}}_0, \\ \boldsymbol{m}^-_k(\boldsymbol{\xi}) &\coloneqq \boldsymbol{A}_{k-1} \boldsymbol{m}_{k-1}(\boldsymbol{\xi}) + \boldsymbol{\xi}^{\tilde{\mathbf{q}}}_{k-1}, \\ \boldsymbol{m}_k(\boldsymbol{\xi}) &\coloneqq \boldsymbol{m}^-_k(\boldsymbol{\xi}) + \boldsymbol{K}_k(\boldsymbol{y}_k - (\boldsymbol{H}_k \boldsymbol{m}^-_k(\boldsymbol{\xi}) + \boldsymbol{\xi}^{\tilde{\boldsymbol{\epsilon}}}_k)), \qquad \text{and} \\ \boldsymbol{m}^{\mathrm{s}}_k(\boldsymbol{\xi}) &\coloneqq \boldsymbol{m}_k(\boldsymbol{\xi}) + \boldsymbol{K}^{\mathrm{s}}_k(\boldsymbol{m}^{\mathrm{s}}_{k+1}(\boldsymbol{\xi}) - \boldsymbol{m}^-_{k+1}(\boldsymbol{\xi})), \end{aligned}$$

Note that $\boldsymbol{m}_0(\mathbf{x}) = \mathbf{u}_0$ and hence

$$\begin{aligned} \boldsymbol{m}^-_k(\mathbf{x}) &= \boldsymbol{A}_{k-1} \underbrace{\boldsymbol{m}_{k-1}(\mathbf{x})}_{=\mathbf{u}^f_k} + \tilde{\mathbf{q}}_{k-1} = \mathbf{u}^{f-}_k, \qquad \text{and} \\ \boldsymbol{m}_k(\mathbf{x}) &= \mathbf{u}^{f-}_k + \boldsymbol{K}_k(\boldsymbol{y}_k - \underbrace{(\boldsymbol{H}_k \boldsymbol{m}^-_k(\mathbf{x}) + \tilde{\mathbf{y}}_k)}_{=\mathbf{y}^{f-}_k}) = \mathbf{u}^f_k \end{aligned}$$

by induction on k. Moreover, $\boldsymbol{m}^{\mathrm{s}}_K(\mathbf{x}) = \boldsymbol{m}_K(\mathbf{x}) = \mathbf{u}^f_K = \mathbf{u}^{\mathrm{s}}_K$ and thus

$$\boldsymbol{m}^{\mathrm{s}}_k(\mathbf{x}) = \underbrace{\boldsymbol{m}_k(\mathbf{x})}_{=\mathbf{u}^f_k} + \boldsymbol{K}^{\mathrm{s}}_k(\underbrace{\boldsymbol{m}^{\mathrm{s}}_{k+1}(\mathbf{x})}_{=\mathbf{u}^{\mathrm{s}}_{k+1}} - \underbrace{\boldsymbol{m}^-_{k+1}(\mathbf{x})}_{=\mathbf{u}^{f-}_k}) = \mathbf{u}^{\mathrm{s}}_k$$

by induction on k. Finally, by Lemma B.6, we arrive at

$$(\mathbf{u}^{\mathrm{s}}_1, \dots, \mathbf{u}^{\mathrm{s}}_K) = \boldsymbol{\mathcal{M}}_{\tilde{\boldsymbol{\Sigma}}, \boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{Y}}, \boldsymbol{y}_{1:K}}(\mathbf{x}) \overset{d}{=} \Big(\boldsymbol{\mathcal{U}}(\mathbf{x}) \mid \boldsymbol{\mathcal{Y}}(\mathbf{x}) = \boldsymbol{y}_{1:K}\Big) = \Big((\mathbf{u}_1, \dots, \mathbf{u}_K) \mid \mathbf{y}_1 = \boldsymbol{y}_1, \dots, \mathbf{y}_K = \boldsymbol{y}_K\Big).$$

□

Proposition B.8 (Inverse-Free Posterior Sampling)**.** *Samples from the smoothing posterior can be equivalently computed by means of the recursion*

$$\mathbf{u}^{\mathrm{s}}_k = \mathbf{u}^{f-}_k + \boldsymbol{P}^-_k \mathbf{w}^{\mathrm{s}}_k,$$

where $\mathbf{w}^{\mathrm{s}}_K \coloneqq \boldsymbol{H}_K^\top \boldsymbol{G}_K^{-1}(\boldsymbol{y}_K - \mathbf{y}^{f-}_K)$*, and*

$$\begin{aligned} \mathbf{w}^{\mathrm{s}}_k &\coloneqq \boldsymbol{H}_k \boldsymbol{G}_k^{-1}(\boldsymbol{y}_k - \mathbf{y}^{f-}_k) + (\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \mathbf{w}^{\mathrm{s}}_{k+1} \\ &= \boldsymbol{H}_k \boldsymbol{G}_k^{-1}(\boldsymbol{y}_k - \mathbf{y}^{f-}_k) + (\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1} \boldsymbol{H}_k \boldsymbol{P}^-_k) \boldsymbol{A}_k^\top \mathbf{w}_{k+1} \end{aligned}$$

for $k = 0, \dots, K-1$*. Moreover,*

$$\mathbf{u}^{\mathrm{s}}_k = \mathbf{u}^f_k + \boldsymbol{P}_k \boldsymbol{A}_k^\top \mathbf{w}^{\mathrm{s}}_{k+1}$$

pointwise for $k = 0, \dots, K-1$*.*

Proof.

$$\mathbf{u}_K^{\mathrm{s}} = \mathbf{u}_K^f = \mathbf{u}_K^{f-} + \boldsymbol{K}_K(\boldsymbol{y}_K - \mathbf{y}_K^{f-}) = \mathbf{u}_K^{f-} + \boldsymbol{P}_K^- \underbrace{\boldsymbol{H}_K^\top \boldsymbol{G}_K^{-1}(\boldsymbol{y}_K - \mathbf{y}_K^{f-})}_{=\mathbf{w}_K^{\mathrm{s}}}$$

Now assume that $\mathbf{u}_{k+1}^{\mathrm{s}} = \mathbf{u}_{k+1}^{f-} + \boldsymbol{P}_{k+1}^- \mathbf{w}_{k+1}^{\mathrm{s}}$, which is equivalent to $\mathbf{u}_{k+1}^{\mathrm{s}} - \mathbf{u}_{k+1}^{f-} = \boldsymbol{P}_{k+1}^- \mathbf{w}_{k+1}^{\mathrm{s}}$. Then

$$\begin{aligned}
\mathbf{u}_k^{\mathrm{s}} &= \mathbf{u}_k^f + \boldsymbol{K}_k^{\mathrm{s}}(\mathbf{u}_{k+1}^{\mathrm{s}} - \mathbf{u}_{k+1}^{f-}) \\
&= \mathbf{u}_k^f + \boldsymbol{P}_k \boldsymbol{A}_k^\top (\boldsymbol{P}_{k+1}^-)^{-1} \boldsymbol{P}_{k+1}^- \mathbf{w}_{k+1}^{\mathrm{s}} \\
&= \mathbf{u}_k^f + \boldsymbol{P}_k \boldsymbol{A}_k^\top \mathbf{w}_{k+1}^{\mathrm{s}} \\
&= \mathbf{u}_k^{f-} + \boldsymbol{K}_k(\boldsymbol{y}_k - \mathbf{y}_k^{f-}) + \boldsymbol{P}_k \boldsymbol{A}_k^\top (\boldsymbol{P}_{k+1}^-)^{-1} \boldsymbol{P}_{k+1}^- \mathbf{w}_{k+1}^{\mathrm{s}} \\
&= \mathbf{u}_k^{f-} + \boldsymbol{K}_k(\boldsymbol{y}_k - \mathbf{y}_k^{f-}) + (\boldsymbol{P}_k^- - \boldsymbol{K}_k \boldsymbol{G}_k \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \mathbf{w}_{k+1}^{\mathrm{s}} \\
&= \mathbf{u}_k^{f-} + \boldsymbol{P}_k^- \underbrace{\left(\boldsymbol{H}_k^\top \boldsymbol{G}_k^{-1}(\boldsymbol{y}_k - \mathbf{y}_k^{f-}) + (\boldsymbol{I} - \boldsymbol{H}_k^\top \boldsymbol{K}_k^\top) \boldsymbol{A}_k^\top \mathbf{w}_k^{\mathrm{s}} \right)}_{=\mathbf{w}_k^{\mathrm{s}}}.
\end{aligned}$$

□

Algorithm B.1 CAKF/CAKS Sampler

fn SAMPLE($\{\cdots, \hat{\boldsymbol{M}}_k, \check{\boldsymbol{V}}_k, \hat{\boldsymbol{W}}_k, \dots\}_{k=0}^K$)
  $\hat{\mathbf{u}}_0^f \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$
  for $k = 1, \dots, K$ **do**
    $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_{k-1})$
    $\check{\boldsymbol{\epsilon}}_k \sim \mathcal{N}(\mathbf{0}, \check{\boldsymbol{\Lambda}}_k)$
    $\hat{\mathbf{u}}_k^{f-} \leftarrow \boldsymbol{A}_{k-1}[\hat{\mathbf{u}}_{k-1}^f] + \boldsymbol{b}_{k-1} + \mathbf{q}_{k-1}$
    $\hat{\mathbf{w}}_k \leftarrow \hat{\boldsymbol{W}}_k(\check{\boldsymbol{V}}_k^\top(\check{\boldsymbol{y}}_k - \check{\boldsymbol{H}}_k \hat{\mathbf{u}}_k^{f-} - \check{\boldsymbol{\epsilon}}_k))$
    $\hat{\mathbf{u}}_k^f \leftarrow \hat{\mathbf{u}}_k^{f-} + \hat{\boldsymbol{P}}_k^-[\hat{\mathbf{w}}_k]$
  $\hat{\mathbf{w}}_n^{\mathrm{s}} = \hat{\mathbf{w}}_K$
  for $k = K-1, \dots, 0$ **do**
    $\hat{\mathbf{w}}_k^{\mathrm{s}} \leftarrow \hat{\mathbf{w}}_k + (\boldsymbol{I} - \hat{\boldsymbol{W}}_k \hat{\boldsymbol{W}}_k^\top \hat{\boldsymbol{P}}_k^-) \boldsymbol{A}_k^\top [\hat{\mathbf{w}}_{k+1}^{\mathrm{s}}]$
  return $\{\hat{\mathbf{u}}_k^{\mathrm{s}} = \hat{\mathbf{u}}_k^f + \hat{\boldsymbol{P}}_k \boldsymbol{A}_k^\top [\hat{\mathbf{w}}_k^{\mathrm{s}}]\}_{k=0}^K$

B.4 Temporal Interpolation

In practice, we are often interested in interpolating a set of discrete-time measurements. To this end, we need to exchange the discrete-time dynamics model in Definition B.1 by a continuous-time dynamics model.

Definition B.9 (Continuous-Discrete LGSSM)**.** A *continuous-discrete linear-Gaussian state-space model* (CD-LGSSM) is a pair $(\{\mathbf{u}(t)\}_{t \in \mathbb{T}}, \{\mathbf{y}_k\}_{k=1}^K)$ of a continuous-time Gauss–Markov process $\mathbf{u}$ with transition kernels

$$\mathrm{P}(\mathbf{u}(t) \mid \mathbf{u}(s) = \boldsymbol{u}(s)) = \mathcal{N}(\boldsymbol{A}(t,s)\boldsymbol{u}(s) + \boldsymbol{b}(t,s), \boldsymbol{Q}(t,s))$$

and a discrete-time Gauss–Markov process $\{\mathbf{y}_k\}_{k=1}^K$ defined by

$$\mathbf{y}_k = \boldsymbol{H}_k \mathbf{u}(t_k^{\mathrm{train}}) + \boldsymbol{\epsilon}_k,$$

where $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_k)$ for $k = 1, \dots, K$ and $\mathbf{u}$ are pairwise independent.

We write $\boldsymbol{\mu}(t) \coloneqq \mathbb{E}[\mathbf{u}(t)]$ and $\boldsymbol{\Sigma}(t) \coloneqq \mathrm{Cov}[\mathbf{u}(t), \mathbf{u}(t)]$ for the mean and covariance functions of the latent continuous-time Gauss–Markov process.

In practice, we want to be able to access the interpolant efficiently (i.e., without revisiting all training data) and on-demand, as we often do not know the query locations in advance. It is well-known that this can be achieved by running the Kalman filter and RTS smoother on the discretized LGSSM corresponding to $\mathbf{u}_k \coloneqq \mathbf{u}(t_k^{\mathrm{train}})$ followed by an interpolation step on the filter and/or smoother states that only involves the neighboring training time points. The above is formalized in Corollary B.10.

Algorithm B.2 CAKF Interpolation

fn InterpolateFilter($t, \mathbf{u}, \{t_k^{\text{train}}, \hat{\boldsymbol{m}}_k, \hat{\boldsymbol{M}}_k\}_{k=1}^K$)
  Find k such that $t_k^{\text{train}} \le t < t_{k+1}^{\text{train}}$, where $k = 0$ if $t < t_1^{\text{train}}$ and $k = K$ if $t \ge t_K^{\text{train}}$.
  if $t = t_k^{\text{train}}$ **then**
    $\hat{\boldsymbol{m}}(t) \leftarrow \hat{\boldsymbol{m}}_k$
    $\hat{\boldsymbol{M}}(t) \leftarrow \hat{\boldsymbol{M}}_k$
  else if $k < 1$ **then**
    $\hat{\boldsymbol{m}}(t) \leftarrow \boldsymbol{\mu}(t)$
    $\hat{\boldsymbol{M}}(t) \leftarrow () \in \mathbb{R}^{D \times 0}$
  else
    $\hat{\boldsymbol{m}}(t) \leftarrow \boldsymbol{A}(t, t_k^{\text{train}})[\hat{\boldsymbol{m}}_k] + \boldsymbol{b}(t, t_k^{\text{train}})$
    $\hat{\boldsymbol{M}}(t) \leftarrow \boldsymbol{A}(t, t_k^{\text{train}})[\hat{\boldsymbol{M}}_k^+]$
  return $\hat{\boldsymbol{m}}(t), \hat{\boldsymbol{M}}(t)$ ▷ $\hat{\boldsymbol{P}}(t) = \boldsymbol{\Sigma}(t) - \hat{\boldsymbol{M}}(t)\hat{\boldsymbol{M}}(t)^\top$

Algorithm B.3 CAKS Interpolation

fn InterpolateSmoother($t, \mathbf{u}, \{t_k^{\text{train}}, \hat{\boldsymbol{w}}_k^{\text{s}}, \hat{\boldsymbol{W}}_k^{\text{s}}, \ldots\}_{k=1}^K$)
  Find k such that $t_k^{\text{train}} \le t < t_{k+1}^{\text{train}}$, where $k = 0$ if $t < t_1^{\text{train}}$ and $k = K$ if $t \ge t_K^{\text{train}}$.
  if $t = t_k^{\text{train}}$ **then**
    return $\hat{\boldsymbol{m}}_k^{\text{s}}, \hat{\boldsymbol{M}}_k^{\text{s}}$ ▷ $\hat{\boldsymbol{P}}^{\text{s}}(t) = \hat{\boldsymbol{P}}_k^{\text{s}} = \boldsymbol{\Sigma}_k - \hat{\boldsymbol{M}}_k^{\text{s}}(\hat{\boldsymbol{M}}_k^{\text{s}})^\top$
  $\hat{\boldsymbol{m}}(t), \hat{\boldsymbol{M}}(t) \leftarrow$ InterpolateFilter($t, \mathbf{u}, \{t_k^{\text{train}}, \ldots\}_{k=1}^K$)
  if $k < K$ **then**
    $\hat{\boldsymbol{P}}(t) \leftarrow \boldsymbol{\Sigma}(t) - \hat{\boldsymbol{M}}(t)\hat{\boldsymbol{M}}(t)^\top$
    $\hat{\boldsymbol{m}}^{\text{s}}(t) \leftarrow \hat{\boldsymbol{m}}(t) + \hat{\boldsymbol{P}}(t)\boldsymbol{A}(t_{k+1}^{\text{train}}, t)^\top[\hat{\boldsymbol{w}}_{k+1}^{\text{s}}]$
    $\hat{\boldsymbol{M}}^{\text{s}}(t) \leftarrow \begin{pmatrix}\hat{\boldsymbol{M}}(t) & \hat{\boldsymbol{P}}(t)\boldsymbol{A}(t_{k+1}^{\text{train}}, t)^\top[\hat{\boldsymbol{W}}_{k+1}^{\text{s}}]\end{pmatrix}$
  else
    $\hat{\boldsymbol{m}}^{\text{s}}(t) \leftarrow \hat{\boldsymbol{m}}(t)$
    $\hat{\boldsymbol{M}}^{\text{s}}(t) \leftarrow \hat{\boldsymbol{M}}(t)$
  return $\hat{\boldsymbol{m}}^{\text{s}}(t), \hat{\boldsymbol{M}}^{\text{s}}(t)$ ▷ $\hat{\boldsymbol{P}}^{\text{s}}(t) = \boldsymbol{\Sigma}(t) - \hat{\boldsymbol{M}}^{\text{s}}(t)\hat{\boldsymbol{M}}^{\text{s}}(t)^\top$

Corollary B.10 (Temporal Interpolation in Kalman Filter and RTS Smoother)**.** *Let $t \in \mathbb{T}$ and $0 \le k \le K$.*

(i) If $t_k^{\text{train}} \le t < t_{k+1}^{\text{train}}$ with $1 \le k < K$ or $t_k^{\text{train}} \le t$ with $k = K$, then

$$\mathbf{u}^f(t) \coloneqq (\mathbf{u}(t) \mid \mathbf{y}_{1:k} = \boldsymbol{y}_{1:k}) \sim \mathcal{N}(\boldsymbol{m}(t), \boldsymbol{P}(t))$$

with

$$\begin{aligned}\boldsymbol{m}(t) &\coloneqq \boldsymbol{A}(t, t_k^{\text{train}})\boldsymbol{m}_k + \boldsymbol{b}(t, t_k^{\text{train}}), \qquad \text{and}\\ \boldsymbol{P}(t) &\coloneqq \boldsymbol{A}(t, t_k^{\text{train}})\boldsymbol{P}_k\boldsymbol{A}(t, t_k^{\text{train}})^\top + \boldsymbol{Q}(t, t_k^{\text{train}})\\ &= \boldsymbol{\Sigma}(t) - \underbrace{\boldsymbol{A}(t, t_k^{\text{train}})\boldsymbol{M}_k}_{=:\boldsymbol{M}(t)}(\boldsymbol{A}(t, t_k^{\text{train}})\boldsymbol{M}_k)^\top.\end{aligned}$$

For $t < t_1^{\text{train}}$, we extend the definition by $\mathbf{u}^f(t) \coloneqq \mathbf{u}(t)$, i.e., $\boldsymbol{m}(t) \coloneqq \boldsymbol{\mu}(t)$, $\boldsymbol{P}(t) \coloneqq \boldsymbol{\Sigma}(t)$, and $\boldsymbol{M}(t) \in \mathbb{R}^{D \times 0}$.

(ii) If $t_k^{\text{train}} \le t < t_{k+1}^{\text{train}}$ with $1 \le k < K$ or $t < t_{k+1}^{\text{train}}$ with $k = 0$, then

$$\mathbf{u}^{\text{s}}(t) \coloneqq (\mathbf{u}(t) \mid \mathbf{y}_{1:K} = \boldsymbol{y}_{1:K}) \sim \mathcal{N}(\boldsymbol{m}^{\text{s}}(t), \boldsymbol{P}^{\text{s}}(t))$$

with

$$\begin{aligned}\boldsymbol{m}^{\text{s}}(t) &\coloneqq \boldsymbol{m}(t) + \boldsymbol{K}^{\text{s}}(t)(\boldsymbol{m}_{k+1}^{\text{s}} - \boldsymbol{m}(t))\\ &= \boldsymbol{m}(t) + \boldsymbol{P}(t)\boldsymbol{A}(t_{k+1}^{\text{train}}, t)^\top \boldsymbol{w}_{k+1}^{\text{s}}, \qquad \text{and}\\ \boldsymbol{P}^{\text{s}}(t) &\coloneqq \boldsymbol{P}^{\text{s}}(t) + \boldsymbol{K}^{\text{s}}(t)(\boldsymbol{P}_{k+1}^{\text{s}} - \boldsymbol{P}(t))\boldsymbol{K}^{\text{s}}(t)^\top\\ &= \boldsymbol{P}(t) - \boldsymbol{P}(t)\boldsymbol{A}(t, t_{k+1}^{\text{train}})^\top \boldsymbol{W}_{k+1}^{\text{s}}(\boldsymbol{P}(t)\boldsymbol{A}(t, t_{k+1}^{\text{train}})^\top \boldsymbol{W}_{k+1}^{\text{s}})^\top,\end{aligned}$$

where $\boldsymbol{K}^{\text{s}}(t) \coloneqq \boldsymbol{P}(t)\boldsymbol{A}(t, t_{k+1}^{\text{train}})^\top(\boldsymbol{P}_{k+1}^-)^{-1}$. Again, we extend the definition by $\mathbf{u}^{\text{s}}(t) \coloneqq \mathbf{u}^f(t)$ for $t > t_K^{\text{train}}$, i.e., $\boldsymbol{m}^{\text{s}}(t) \coloneqq \boldsymbol{m}(t)$ and $\boldsymbol{P}^{\text{s}}(t) \coloneqq \boldsymbol{P}(t)$.

Proof. This follows from Theorems B.2 and B.4 and Propositions B.3 and B.5. □

Corollary B.10 also shows that the efficient interpolation capabilities extend to the downdate-form versions of the Kalman filter and RTS smoother from Propositions B.3 and B.5 that form the basis of the CAKF and CAKS. Using this result, we can derive Algorithms B.2 and B.3. An efficient version of Algorithm B.1 that allows for sampling at intermediate points can be derived analogously.

Algorithm B.4 CAKF Update Step (Iterative Version)

1: **fn** UPDATE($\hat{\boldsymbol{m}}_k^-, \hat{\boldsymbol{M}}_k^-, \boldsymbol{\Sigma}_k, \boldsymbol{H}_k, \boldsymbol{\Lambda}_k, \boldsymbol{y}_k$)
2: $\quad \hat{\boldsymbol{P}}_k^- \leftarrow \boldsymbol{\Sigma}_k - \hat{\boldsymbol{M}}_k^- (\hat{\boldsymbol{M}}_k^-)^\top \in \mathbb{R}^{D \times D}$
3: $\quad \hat{\boldsymbol{G}}_k \leftarrow \boldsymbol{H}_k \hat{\boldsymbol{P}}_k^- \boldsymbol{H}_k^\top + \boldsymbol{\Lambda}_k \in \mathbb{R}^{N_k \times N_k}$
4: $\quad \hat{\boldsymbol{v}}_k^{(0)} \leftarrow \boldsymbol{0} \in \mathbb{R}^{N_k}$
5: $\quad \hat{\boldsymbol{V}}_k^{(0)} \leftarrow (\quad) \in \mathbb{R}^{N_k \times 0}$
6: $\quad \hat{\boldsymbol{r}}_k^{(0)} \leftarrow \boldsymbol{y}_k - \boldsymbol{H}_k[\hat{\boldsymbol{m}}_k^-]$
7: $\quad$ **while** $\neg$STOPPINGCRITERION$(i, \hat{\boldsymbol{r}}_k^{(i)}, \dots)$ **do**
8: $\quad\quad \hat{\boldsymbol{s}}_k^{(i)} \leftarrow$ POLICY$(i, \hat{\boldsymbol{r}}_k^{(i-1)}, \dots)$
9: $\quad\quad \hat{\boldsymbol{r}}_k^{(i)} \leftarrow \hat{\boldsymbol{r}}_k^{(0)} - \hat{\boldsymbol{G}}_k[\hat{\boldsymbol{v}}_k^{(i-1)}]$
10: $\quad\quad \alpha_k^{(i)} \leftarrow \langle \hat{\boldsymbol{s}}_k^{(i)}, \hat{\boldsymbol{r}}_k^{(i)} \rangle$
11: $\quad\quad \boldsymbol{d}_k^{(i)} \leftarrow (\boldsymbol{I} - \hat{\boldsymbol{V}}_k^{(i-1)} (\hat{\boldsymbol{V}}_k^{(i-1)})^\top \hat{\boldsymbol{G}}_k)[\hat{\boldsymbol{s}}_k^{(i)}]$
12: $\quad\quad \eta_k^{(i)} \leftarrow \langle \hat{\boldsymbol{s}}_k^{(i)}, \hat{\boldsymbol{G}}_k[\boldsymbol{d}_k^{(i)}] \rangle$
13: $\quad\quad \hat{\boldsymbol{v}}_k^{(i)} \leftarrow \hat{\boldsymbol{v}}_k^{(i-1)} + \frac{\alpha_k^{(i)}}{\eta_k^{(i)}} \boldsymbol{d}_k^{(i)}$
14: $\quad\quad \hat{\boldsymbol{V}}_k^{(i)} \leftarrow \begin{pmatrix} \hat{\boldsymbol{V}}_k^{(i-1)} & \frac{1}{\sqrt{\eta_k^{(i)}}} \boldsymbol{d}^{(i)} \end{pmatrix} \in \mathbb{R}^{N_k \times i}$
15: $\quad \hat{\boldsymbol{w}}_k \leftarrow \boldsymbol{H}_k^\top \hat{\boldsymbol{v}}_k^{(i)}$
16: $\quad \hat{\boldsymbol{W}}_k \leftarrow \boldsymbol{H}_k^\top \hat{\boldsymbol{V}}_k^{(i)}$
17: $\quad \hat{\boldsymbol{m}}_k \leftarrow \hat{\boldsymbol{m}}_k^- + \hat{\boldsymbol{P}}_k^-[\hat{\boldsymbol{w}}_k]$
18: $\quad \hat{\boldsymbol{M}}_k \leftarrow \begin{pmatrix} \hat{\boldsymbol{M}}_k^- & \hat{\boldsymbol{P}}_k^-[\hat{\boldsymbol{W}}_k] \end{pmatrix}$
19: $\quad$ **return** $(\hat{\boldsymbol{m}}_k, \hat{\boldsymbol{M}}_k)$

B.5 Iterative Version of the CAKF Update Step

Proposition B.11. *When an identical* POLICY *is used, Algorithms 2 and B.4 are equivalent (in exact precision).*

Proof. The principal difference between the two algorithms is that the quantities $\hat{\boldsymbol{w}}_k$ and $\hat{\boldsymbol{W}}_k$ are calculated differently. To show that these actually take the same values for the same policy, first note that in Algorithm 2 we have that $\hat{\boldsymbol{W}}_k \hat{\boldsymbol{W}}_k^\top = \check{\boldsymbol{H}}_k^\top \check{\boldsymbol{G}}_k^\dagger \check{\boldsymbol{H}}_k = \boldsymbol{H}_k^\top \hat{\boldsymbol{S}}_k (\hat{\boldsymbol{S}}_k^\top \hat{\boldsymbol{G}}_k \hat{\boldsymbol{S}}_k)^\dagger \hat{\boldsymbol{S}}_k^\top \boldsymbol{H}_k$. In Algorithm B.4 the matrix $\hat{\boldsymbol{V}}_k^{(\hat{N}_k)}$ has the same span as $\hat{\boldsymbol{S}}_k$, but is orthogonalised to remove the need for the matrix inversion in $\hat{\boldsymbol{W}}_k \hat{\boldsymbol{W}}_k^\top$. This essentially follows from the fact that Line 11 implements a version of the Gram-Schmidt procedure with an adjustment to enforce orthogonality with-respect to $\langle \cdot, \cdot \rangle_{\hat{\boldsymbol{G}}_k}$ rather than the standard Euclidean inner product.

To show this we proceed by induction. For the base step we need only show that $(\hat{\boldsymbol{V}}_k^{(1)})^\top \hat{\boldsymbol{G}}_k \hat{\boldsymbol{V}}_k^{(1)} = \boldsymbol{I}$; this follows from the fact that since $\hat{\boldsymbol{V}}_k^{(0)}$ is an empty matrix, $\boldsymbol{d}_k^{(1)} = \hat{\boldsymbol{s}}_k^{(1)}$ and therefore

$$
\begin{aligned}
(\boldsymbol{d}_k^{(1)})^\top \hat{\boldsymbol{G}}_k \boldsymbol{d}_k^{(1)} &= (\hat{\boldsymbol{s}}_k^{(1)})^\top \hat{\boldsymbol{G}}_k \boldsymbol{d}_k^{(1)} = \eta_k^{(1)} \\
\implies \left\| \frac{\boldsymbol{d}_k^{(1)}}{\sqrt{\eta_k^{(1)}}} \right\|_{\hat{\boldsymbol{G}}_k} &= 1.
\end{aligned}
$$

For the induction step suppose that $\hat{\boldsymbol{V}}_k^{(i-1)}$ is $\hat{\boldsymbol{G}}_k$-orthonormal. Let $\boldsymbol{z}^{(i)} = \frac{\boldsymbol{d}_k^{(i)}}{\sqrt{\eta_k^{(i)}}}$ and consider the matrix

$$
(\hat{\boldsymbol{V}}_k^{(i)})^\top \hat{\boldsymbol{G}}_k \hat{\boldsymbol{V}}_k^{(i)} = \begin{pmatrix} (\hat{\boldsymbol{V}}_k^{(i-1)})^\top \hat{\boldsymbol{G}}_k \hat{\boldsymbol{V}}_k^{(i-1)} & (\hat{\boldsymbol{V}}_k^{(i-1)})^\top \hat{\boldsymbol{G}}_k \boldsymbol{z}^{(i)} \\ (\boldsymbol{z}^{(i)})^\top \hat{\boldsymbol{G}}_k (\hat{\boldsymbol{V}}_k^{(i-1)}) & (\boldsymbol{z}^{(i)})^\top \hat{\boldsymbol{G}}_k \boldsymbol{z}^{(i)} \end{pmatrix}.
$$

It is straightforward to show that $(\boldsymbol{z}^{(i)})^\top \hat{\boldsymbol{G}}_k (\hat{\boldsymbol{V}}_k^{(i-1)}) = 0$, since

$$
(\boldsymbol{d}_k^{(i)})^\top \hat{\boldsymbol{G}}_k (\hat{\boldsymbol{V}}_k^{(i-1)}) = (\hat{\boldsymbol{s}}_k^{(i)})^\top (\boldsymbol{I} - \hat{\boldsymbol{G}}_k \hat{\boldsymbol{V}}_k^{(i-1)} (\hat{\boldsymbol{V}}_k^{(i-1)})^\top) \hat{\boldsymbol{G}}_k (\hat{\boldsymbol{V}}_k^{(i-1)}) \tag{B.5}
$$

$$= (\hat{\boldsymbol{s}}_k^{(i)})^\top \hat{\boldsymbol{G}}_k \hat{\boldsymbol{V}}_k^{(i-1)} - (\hat{\boldsymbol{s}}_k^{(i)})^\top \hat{\boldsymbol{G}}_k \hat{\boldsymbol{V}}_k^{(i-1)} \underbrace{(\hat{\boldsymbol{V}}_k^{(i-1)})^\top \hat{\boldsymbol{G}}_k (\hat{\boldsymbol{V}}_k^{(i-1)})}_{=\boldsymbol{I}} \tag{B.6}$$

$$= \boldsymbol{0} \tag{B.7}$$

by the inductive assumption. It remains to show that $(\boldsymbol{z}^{(i)})^\top \hat{\boldsymbol{G}} \boldsymbol{z}^{(i)} = 1$. This follows from observing that

$$\begin{aligned}
\|\boldsymbol{d}_k^{(i)}\|_{\hat{\boldsymbol{G}}_k}^2 &= (\hat{\boldsymbol{s}}_k^{(i)})^\top (\boldsymbol{I} - \hat{\boldsymbol{G}}_k \hat{\boldsymbol{V}}_k^{(i-1)} (\hat{\boldsymbol{V}}_k^{(i-1)})^\top) \hat{\boldsymbol{G}}_k \boldsymbol{d}_k^{(i)} \\
&= (\hat{\boldsymbol{s}}_k^{(i)})^\top \hat{\boldsymbol{G}}_k \boldsymbol{d}_k^{(i)} - (\hat{\boldsymbol{s}}_k^{(i)})^\top \hat{\boldsymbol{G}}_k \hat{\boldsymbol{V}}_k^{(i-1)} \underbrace{(\hat{\boldsymbol{V}}_k^{(i-1)})^\top \hat{\boldsymbol{G}}_k \boldsymbol{d}_k^{(i)}}_{=\boldsymbol{0}} \\
&= \eta_k^{(i)}
\end{aligned}$$

where equality with zero is from the calculation above. It follows that $\hat{\boldsymbol{S}}_k (\hat{\boldsymbol{S}}_k^\top \hat{\boldsymbol{G}} \hat{\boldsymbol{S}}_k) \hat{\boldsymbol{S}}_k^\top = (\hat{\boldsymbol{V}}_k^{(\tilde{N}_k)})^\top \hat{\boldsymbol{V}}_k^{(\tilde{N}_k)}$, which completes the proof. □

C SPACE-TIME SEPARABLE GAUSS–MARKOV PROCESSES

Assume we are given a spatiotemporal regression problem over the domain $\mathbb{Z} = [t_0, T] \times \mathbb{X}$ and a Gaussian process prior

$$\mathrm{f} \sim \mathcal{GP}(\mu, \Sigma) \tag{C.1}$$

for the latent function $f^\star \colon \mathbb{Z} \to \mathbb{R}$, where $\mu : \mathbb{Z} \to \mathbb{R}$ and $\Sigma : \mathbb{Z} \times \mathbb{Z} \to \mathbb{R}$. Our goal will be to translate this batch GP regression problem into state-space form where under suitable assumptions the state dynamics are Markovian, such that we can perform exact and importantly linear-time inference via Bayesian filtering and smoothing.

C.1 Spatiotemporal GP Regression in State-Space Form

As a first step, we augment the state with a sufficient number of $D' - 1$ time derivatives, i.e.,

$$\mathbf{f}(t, \boldsymbol{x}) = \begin{pmatrix} \mathbf{f}_0(t, \boldsymbol{x}) \\ \vdots \\ \mathbf{f}_{d'-1}(t, \boldsymbol{x}) \end{pmatrix} \coloneqq \begin{pmatrix} \mathbf{f}(t, \boldsymbol{x}) \\ \frac{\partial}{\partial t} \mathbf{f}(t, \boldsymbol{x}) \\ \vdots \\ \frac{\partial^{(D'-1)}}{\partial t^{(D'-1)}} \mathbf{f}(t, \boldsymbol{x}) \end{pmatrix} \in \mathbb{R}^{D'}, \tag{C.2}$$

and assume the resulting Gaussian process is space-time separable.

Definition C.1 (Space-Time Separable Gaussian Process)**.** A D'-output Gaussian process $\mathbf{f} \sim \mathcal{GP}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with index set $[t_0, T] \times \mathbb{X}$ is called *space-time separable* if $\boldsymbol{\mu}(t, \boldsymbol{x}) = \boldsymbol{\mu}^t(t) \cdot \mu^{\boldsymbol{x}}(\boldsymbol{x})$ and

$$\boldsymbol{\Sigma}((t_1, \boldsymbol{x}_1), (t_2, \boldsymbol{x}_2)) = \boldsymbol{\Sigma}^t(t_1, t_2) \cdot \Sigma^{\boldsymbol{x}}(\boldsymbol{x}_1, \boldsymbol{x}_2).$$

Then given that the temporal process $\mathbf{f}^t \sim \mathcal{GP}(\boldsymbol{\mu}^t, \boldsymbol{\Sigma}^t)$ is Markovian, we obtain the desired state-space representation, which can be computed exactly in closed form under suitable assumptions on the covariance function $\boldsymbol{\Sigma}^t$ (see Remark C.5). The following result formalizing this argument has been presented previously, but without an explicit proof (Särkkä and Hartikainen, 2012; Solin, 2016; Hamelijnck et al., 2021).

Lemma C.2. *Let $\mathbf{f} \sim \mathcal{GP}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be a space-time separable D'-output Gaussian process with index set $[t_0, T] \times \mathbb{X}$ such that $\mathbf{f}^t \sim \mathcal{GP}(\boldsymbol{\mu}^t, \boldsymbol{\Sigma}^t)$ is Markov with transition densities $p(\mathbf{f}^t(t) \mid \mathbf{f}^t(s)) = \mathcal{N}\big(\mathbf{f}^t(t); \boldsymbol{A}^t(t, s)\mathbf{f}^t(s) + \boldsymbol{b}^t(t, s), \boldsymbol{Q}^t(t, s)\big)$. Let $\boldsymbol{X} \in \mathbb{X}^{N_{\boldsymbol{X}}}$ and define*

$$\mathbf{u}(t) \coloneqq \mathbf{f}(t, \boldsymbol{X}) = \begin{pmatrix} \mathbf{f}_0(t, \boldsymbol{X}) \\ \vdots \\ \mathbf{f}_{d'-1}(t, \boldsymbol{X}) \end{pmatrix} \in \mathbb{R}^{D' \cdot N_{\boldsymbol{X}}} \tag{C.3}$$

for all $t \in [t_0, T]$. Then $\mathbf{u}$ is a Gauss–Markov process with transition densities

$$p(\mathbf{u}(t) \mid \mathbf{u}(s)) = \mathcal{N}(\mathbf{u}(t); \boldsymbol{A}(t, s)\mathbf{u}(s) + \boldsymbol{b}(t, s), \boldsymbol{Q}(t, s)),$$

where

$$\begin{aligned}\boldsymbol{A}(t,s) &\coloneqq \boldsymbol{A}^t(t,s) \otimes \boldsymbol{I}_{N_{\boldsymbol{X}}} \\ \boldsymbol{b}(t,s) &\coloneqq \boldsymbol{b}^t(t,s) \otimes \mu^{\boldsymbol{x}}(\boldsymbol{X}), \qquad \textit{and} \\ \boldsymbol{Q}(t,s) &\coloneqq \boldsymbol{Q}^t(t,s) \otimes \Sigma^{\boldsymbol{x}}(\boldsymbol{X},\boldsymbol{X}).\end{aligned}$$

Remark C.3. Abusing terminology, we refer to $\mathbf{f}$ as a *space-time separable Gauss–Markov process* if $\mathbf{f}$ is space-time separable and $\mathcal{GP}(\boldsymbol{\mu}^t, \boldsymbol{\Sigma}^t)$ is Markov.

To prove Lemma C.2, we will need the following intermediate result, which can be found in most standard textbooks, for example in Appendix B of Bishop (2006). We restate the result here for convenience:

Lemma C.4. *Let*

$$\begin{pmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix}\boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{21}^\top \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{pmatrix}\right).$$

Then

$$\mathbf{x}_2 \mid \mathbf{x}_1 \sim \mathcal{N}(\boldsymbol{A}\mathbf{x}_1 + \boldsymbol{b}, \boldsymbol{Q}),$$

where $\boldsymbol{A} = \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^\dagger$, $\boldsymbol{b} = (\boldsymbol{\mu}_2 - \boldsymbol{A}\boldsymbol{\mu}_1)$, *and* $\boldsymbol{Q} = \boldsymbol{\Sigma}_{22} - \boldsymbol{A}\boldsymbol{\Sigma}_{11}\boldsymbol{A}^\top$.

We can now use Lemma C.4 to show that every Gauss–Markov process has transition densities of the form required by Lemma C.2.

Proof of Lemma C.2. By definition, $\mathbf{u}$ is a $D' \cdot N_{\boldsymbol{X}}$-output Gaussian process with index set $[t_0, T]$, whose mean and covariance functions are given by

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{u}}(t) &\coloneqq \boldsymbol{\mu}^t(t) \otimes \mu^{\boldsymbol{x}}(\boldsymbol{X}), \qquad \text{and} \\ \boldsymbol{\Sigma}_{\mathbf{u}}(t_1,t_2) &\coloneqq \boldsymbol{\Sigma}^t(t_1,t_2) \otimes \Sigma^{\boldsymbol{x}}(\boldsymbol{X},\boldsymbol{X}),\end{aligned}$$

respectively. Let $t_0 \le t_1 < \cdots < t_K \le T$ and define

$$\begin{aligned}\boldsymbol{A}_k^t &\coloneqq \boldsymbol{A}^t(t_{k+1}, t_k), \\ \boldsymbol{b}_k^t &\coloneqq \boldsymbol{b}^t(t_{k+1}, t_k), \qquad \text{and} \\ \boldsymbol{Q}_k^t &\coloneqq \boldsymbol{Q}^t(t_{k+1}, t_k).\end{aligned}$$

We have

$$\begin{aligned}\boldsymbol{\mu}^t(t_{k+1}) &= \boldsymbol{A}_k^t\boldsymbol{\mu}^t(t_k) + \boldsymbol{b}_k^t, \\ \boldsymbol{\Sigma}^t(t_{k+1},t_{k+1}) &= \boldsymbol{A}_k^t\boldsymbol{\Sigma}^t(t_k,t_k)(\boldsymbol{A}_k^t)^\top + \boldsymbol{Q}_k^t, \qquad \text{and} \\ \boldsymbol{\Sigma}^t(t_k,t_{k+l}) &= \boldsymbol{\Sigma}^t(t_k,t_k)\prod_{j=0}^{l-1}(\boldsymbol{A}_{k+j}^t)^\top.\end{aligned}$$

It follows that

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{u}}(t_{k+1}) &= \boldsymbol{\mu}^t(t_{k+1}) \otimes \mu^{\boldsymbol{x}}(\boldsymbol{X}) \\ &= (\boldsymbol{A}_k^t\boldsymbol{\mu}^t(t_k) + \boldsymbol{b}_k^t) \otimes \mu^{\boldsymbol{x}}(\boldsymbol{X}) \\ &= \underbrace{(\boldsymbol{A}_k^t \otimes \boldsymbol{I}_n)}_{=:\boldsymbol{A}_k}(\boldsymbol{\mu}^t(t_k) \otimes \mu^{\boldsymbol{x}}(\boldsymbol{X})) + \underbrace{\boldsymbol{b}_k^t \otimes \mu^{\boldsymbol{x}}(\boldsymbol{X})}_{=:\boldsymbol{b}_k} \\ &= \boldsymbol{A}_k\boldsymbol{\mu}_{\mathbf{u}}(t_k) + \boldsymbol{b}_k,\end{aligned}$$

as well as

$$\begin{aligned}\boldsymbol{\Sigma}_{\mathbf{u}}(t_k,t_k) &= \boldsymbol{\Sigma}^t(t_k,t_k) \otimes \Sigma^{\boldsymbol{x}}(\boldsymbol{X},\boldsymbol{X}) \\ &= (\boldsymbol{A}_k^t\boldsymbol{\Sigma}^t(t_k,t_k)(\boldsymbol{A}_k^t)^\top + \boldsymbol{Q}_k^t) \otimes \Sigma^{\boldsymbol{x}}(\boldsymbol{X},\boldsymbol{X})\end{aligned}$$

$$= (\boldsymbol{A}_k^t \otimes \boldsymbol{I}_n)(\boldsymbol{\Sigma}^t(t_k, t_k) \otimes \Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X}))(\boldsymbol{A}_k^t \otimes \boldsymbol{I}_n)^\top + \underbrace{\boldsymbol{Q}_k^t \otimes \Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})}_{=:\boldsymbol{Q}_k}$$
$$= \boldsymbol{A}_k \boldsymbol{\Sigma}_{\mathbf{u}}(t_k, t_k) \boldsymbol{A}_k^\top + \boldsymbol{Q}_k,$$

and

$$\begin{aligned}
\boldsymbol{\Sigma}_{\mathbf{u}}(t_k, t_{k+l}) &= \boldsymbol{\Sigma}^t(t_k, t_{k+l}) \otimes \Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X}) \\
&= \left(\boldsymbol{\Sigma}^t(t_k, t_k) \prod_{j=0}^{l-1} (\boldsymbol{A}_{k+j}^t)^\top \right) \otimes \Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X}) \\
&= (\boldsymbol{\Sigma}^t(t_k, t_k) \otimes \Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})) \prod_{j=0}^{l-1} (\boldsymbol{A}_{k+j}^t \otimes \boldsymbol{I})^\top \\
&= \boldsymbol{\Sigma}_{\mathbf{u}}(t_k, t_k) \prod_{j=0}^{l-1} \boldsymbol{A}_{k+j}^\top.
\end{aligned}$$

Moreover, by Lemma C.4, we have

$$p(\mathbf{u}(t_{k+1}) \mid \mathbf{u}(t_k)) = \mathcal{N}(\mathbf{u}(t_{k+1}); \boldsymbol{A}_k \mathbf{u}(t_k) + \boldsymbol{b}_k, \boldsymbol{Q}_k).$$

All in all, this shows that

$$p(\mathbf{u}(t_1), \ldots, \mathbf{u}(t_K)) = p(\mathbf{u}(t_1)) \prod_{k=2}^{K} p(\mathbf{u}(t_k) \mid \mathbf{u}(t_{k-1})).$$

The statement then follows from Le Gall (2016, "Consequences of the definition" below Definition 6.2). □

Remark C.5 (Converting Spatiotemporal GP Priors to Space-Time Separable Gauss–Markov Processes). Not every Gaussian process prior $\mathrm{f} \sim \mathcal{GP}(\mu, \Sigma)$ induces a space-time separable Gauss–Markov process, even if both μ and Σ are separable, such that $\mu(\boldsymbol{z}) = \mu^t(t)\mu^{\boldsymbol{x}}(\boldsymbol{x})$ and $\Sigma(\boldsymbol{z}, \boldsymbol{z}') = \Sigma^t(t, t')\Sigma^{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{x}')$, e.g if Σ^t is an exponentiated quadratic kernel. However, if Σ^t is stationary and the spectral density of Σ^t is a rational function of the form

$$S^t(\omega) = \frac{(\text{constant})}{(\text{polynomial in } \omega^2)} \tag{C.4}$$

then a corresponding STSGMP exists. This is the case for example if Σ^t is a Matérn(ν) kernel with differentiability parameter p such that $\nu = p + \frac{1}{2}$. See Hartikainen and Särkkä (2010, Sec. 4), Särkkä et al. (2013), and Solin (2016, Sec. 4.3) for details.

C.2 Pointwise Error Bound

Having formalized assumptions under which the (spatiotemporal) batch GP regression problem can be equivalently formulated in state-space form and thus solved via Bayesian filtering and smoothing, we now aim to give a relative error bound for the approximate posterior mean computed by the CAKS in terms of its approximate marginal variance.

C.2.1 (Iteratively Approximated) Batch Gaussian Process Regression

The CAKF and CAKS can be viewed as performing exact inference in an approximate observation model. Therefore we can analyze its error via the corresponding iterative approximation for the batch GP regression problem as introduced by Wenger et al. (2022).

Definition C.6 (Iteratively Approximated Batch GP Regression)**.** Let $\mathbb{Z}$ be a non-empty set, $\mathrm{f} \sim \mathcal{GP}(\mu, \Sigma)$ a Gaussian process prior for the latent function $f^\star \in \mathbb{H}_\Sigma$ assumed to lie in the RKHS induced by Σ. Define the noise scale $\sigma \geq 0$, the covariance function $\Sigma^\sigma(\boldsymbol{z}, \boldsymbol{z}') \coloneqq \Sigma(\boldsymbol{z}, \boldsymbol{z}') + \sigma^2 \delta(\boldsymbol{z}, \boldsymbol{z}')$ of the observed process (Kanagawa et al.,

2018, Eqn. (32)) and let $y^\star(\cdot) \in \mathbb{H}_{\Sigma^\sigma}$ be the function generating the data[4]. Assume we've observed training data $\boldsymbol{y}^{\text{train}} = y^\star(\boldsymbol{Z}^{\text{train}}) = (y^\star(\boldsymbol{z}_1^{\text{train}}), \dots, y^\star(\boldsymbol{z}_N^{\text{train}}))^\top \in \mathbb{R}^N$ at inputs $\boldsymbol{Z}^{\text{train}} = (\boldsymbol{z}_1^{\text{train}}, \dots, \boldsymbol{z}_N^{\text{train}})^\top \in \mathbb{Z}^N$ and let $\boldsymbol{S} \in \mathbb{R}^{N \times \check{N}}$ be a matrix with linearly independent columns. Following Wenger et al. (2022), define the *iteratively approximated batch GP posterior* as $(\mathrm{f} \mid \boldsymbol{S}^\top \boldsymbol{y}^{\text{train}}) \sim \mathcal{GP}\left(\hat{\mu}^{y^\star}, \hat{\Sigma}\right)$, with

$$\begin{aligned} \hat{\mu}^{y^\star}(\boldsymbol{z}) &= \mu(\boldsymbol{z}) + \Sigma(\boldsymbol{z}, \boldsymbol{Z}^{\text{train}}) \boldsymbol{C} (\boldsymbol{y}^{\text{train}} - \mu(\boldsymbol{Z}^{\text{train}})), \\ \hat{\Sigma}(\boldsymbol{z}, \boldsymbol{z}') &= \Sigma(\boldsymbol{z}, \boldsymbol{z}') - \Sigma(\boldsymbol{z}, \boldsymbol{Z}^{\text{train}}) \boldsymbol{C} \Sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{z}'), \end{aligned} \tag{C.5}$$

where $\boldsymbol{C} = \boldsymbol{S}(\boldsymbol{S}^\top(\Sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{Z}^{\text{train}}) + \sigma^2 \boldsymbol{I})\boldsymbol{S}^\top)^\dagger \boldsymbol{S}^\top$

Lemma C.7 (Iteratively Approximated GP as Exact Inference Given a Modified Observation Model)**.** *Given a Gaussian process prior* $\mathrm{f} \sim \mathcal{GP}(\mu, \Sigma)$ *and training data* $(\boldsymbol{Z}^{\text{train}}, \boldsymbol{y}^{\text{train}})$ *the iteratively approximated batch GP posterior* $(\mathrm{f} \mid \boldsymbol{S}^\top \boldsymbol{y}^{\text{train}}) \sim \mathcal{GP}\left(\hat{\mu}^{y^\star}, \hat{\Sigma}\right)$ *(see Definition C.6) is equivalent to an exact batch GP posterior* $(\mathrm{f} \mid \check{\boldsymbol{y}}^{\text{train}})$ *given observations* $\check{\boldsymbol{y}}^{\text{train}} = \boldsymbol{S}^\top \boldsymbol{y}^{\text{train}}$ *observed according to the modified likelihood* $\check{\mathbf{y}}^{\text{train}} \mid \mathrm{f}(\boldsymbol{Z}^{\text{train}}) \sim \mathcal{N}\left(\boldsymbol{S}^\top \mathrm{f}(\boldsymbol{Z}^{\text{train}}), \sigma^2 \boldsymbol{S}^\top \boldsymbol{S}\right)$.

Proof. By basic properties of Gaussian distributions, we have for arbitrary $\boldsymbol{Z} \in \mathbb{Z}^{N_{\boldsymbol{Z}}}$ that

$$\begin{pmatrix} \check{\boldsymbol{y}}^{\text{train}} \\ \mathrm{f}(\boldsymbol{Z}) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{S}^\top \mu(\boldsymbol{Z}^{\text{train}}) \\ \mu(\boldsymbol{Z}) \end{pmatrix}, \begin{pmatrix} \boldsymbol{S}^\top \Sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{Z}^{\text{train}}) \boldsymbol{S} + \sigma^2 \boldsymbol{S}^\top \boldsymbol{S} & \boldsymbol{S}^\top \Sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{Z}) \\ \Sigma(\boldsymbol{Z}, \boldsymbol{Z}^{\text{train}}) \boldsymbol{S} & \Sigma(\boldsymbol{Z}, \boldsymbol{Z}) \end{pmatrix} \right)$$

is jointly Gaussian. Therefore by Lemma C.4 we have that $\mathrm{f}(\boldsymbol{Z}) \mid \check{\boldsymbol{y}}^{\text{train}} \sim \mathcal{N}\left(\hat{\mu}^{y^\star}(\boldsymbol{Z}), \hat{\Sigma}(\boldsymbol{Z}, \boldsymbol{Z})\right)$ where

$$\begin{aligned} \hat{\mu}^{y^\star}(\boldsymbol{Z}) &= \mu(\boldsymbol{Z}) + \Sigma(\boldsymbol{Z}, \boldsymbol{Z}^{\text{train}}) \boldsymbol{S} (\boldsymbol{S}^\top (\Sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{Z}^{\text{train}}) + \sigma^2 \boldsymbol{I}) \boldsymbol{S}^\top)^\dagger (\check{\boldsymbol{y}}^{\text{train}} - \boldsymbol{S}^\top \mu(\boldsymbol{Z}^{\text{train}})), \\ \hat{\Sigma}(\boldsymbol{Z}, \boldsymbol{Z}) &= \Sigma(\boldsymbol{Z}, \boldsymbol{Z}) - \Sigma(\boldsymbol{Z}, \boldsymbol{Z}^{\text{train}}) \boldsymbol{S} (\boldsymbol{S}^\top (\Sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{Z}^{\text{train}}) + \sigma^2 \boldsymbol{I}) \boldsymbol{S}^\top)^\dagger \boldsymbol{S}^\top \Sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{Z}) \end{aligned}$$

which is equivalent to the form of iteratively approximated GP posterior in Definition C.6. This proves the claim. □

The iteratively approximated posterior mean satisfies a pointwise worst-case error bound in a unit ball in the underlying RKHS, where the bound is given by the approximate standard deviation. Therefore, the output of the approximate method directly provides an error bound on its prediction error, that includes any error introduced through approximation.

Theorem C.8 (Worst-Case Error of (Iteratively Approximated) Batch GP Regression (Wenger et al., 2022))**.** *Consider the (iteratively approximated) GP posterior* $(\mathrm{f} \mid \boldsymbol{S}^\top \boldsymbol{y}^{\text{train}}) \sim \mathcal{GP}\left(\hat{\mu}^{y^\star}, \hat{\Sigma}\right)$ *given in Definition C.6. The pointwise worst-case error of the (approximate) posterior mean* $\hat{\mu}^{y^\star}$ *for an arbitrary data-generating function* $y^\star \in \mathbb{H}_{K^\sigma}$ *such that* $\|y^\star\|_{\mathbb{H}_{\Sigma^\sigma}} \leq 1$ *is given by*

$$\sup_{\substack{y \in \mathbb{H}_{\Sigma^\sigma} \\ \|y\|_{\mathbb{H}_{\Sigma^\sigma}} \leq 1}} |y(\boldsymbol{z}) - \hat{\mu}^y(\boldsymbol{z})| = \sqrt{\hat{\Sigma}(\boldsymbol{z}, \boldsymbol{z}) + \sigma^2}. \tag{C.6}$$

for any $\boldsymbol{z} \in \mathbb{Z} \setminus \boldsymbol{Z}^{\text{train}}$ *not in the training data. In the absence of observation noise, i.e.,* $\sigma^2 = 0$*, this holds for all* $\boldsymbol{z} \in \mathbb{Z}$*. Note that this result also trivially extends to the exact batch GP posterior by choosing* $\boldsymbol{S} = \boldsymbol{I}_{N \times N}$*.*

Proof. This result and its proof are identical to Theorem 2 in Wenger et al. (2022). We give a proof in our notation for completeness. Let $\boldsymbol{z}_0^{\text{train}} = \boldsymbol{z}$, $c_0 = 1$ and $c_j = -(\boldsymbol{C}\Sigma^\sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{z}))_j$ for $j = 1, \dots, N$. Then by Lemma 3.9 of Kanagawa et al. (2018), it holds that

$$\left(\sup_{\substack{y \in \mathbb{H}_{\Sigma^\sigma} \\ \|y\|_{\mathbb{H}_{\Sigma^\sigma}} \leq 1}} |y(\boldsymbol{z}) - \hat{\mu}^y(\boldsymbol{z})| \right)^2 = \left(\sup_{\substack{y \in \mathbb{H}_{\Sigma^\sigma} \\ \|y\|_{\mathbb{H}_{\Sigma^\sigma}} \leq 1}} \sum_{j=0}^{N} c_j y(\boldsymbol{z}_j) \right)^2 = \left\| \sum_{j=0}^{N} c_j \Sigma(\cdot, \boldsymbol{z}_j) \right\|_{\mathbb{H}_{\Sigma^\sigma}}^2$$

[4] By Section 6 of Aronszajn (1950), functions $y \in \mathbb{H}_{\Sigma^\sigma}$ can be written as a sum $y(\cdot) = f(\cdot) + \varepsilon(\cdot)$ of functions $f \in \mathbb{H}_\Sigma$ and $\varepsilon \in \mathbb{H}_{\sigma^2 \delta}$.

$$= \left\| \Sigma^\sigma(\cdot, \boldsymbol{z}_0^{\text{train}}) - \sum_{j=1}^{N} c_j \Sigma^\sigma(\cdot, \boldsymbol{z}_j) \right\|^2_{\mathbb{H}_{\Sigma^\sigma}} = \left\| \Sigma^\sigma(\cdot, \boldsymbol{z}) - \Sigma^\sigma(\cdot, \boldsymbol{Z}^{\text{train}}) \boldsymbol{C} \Sigma^\sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{z}) \right\|^2_{\mathbb{H}_{\Sigma^\sigma}}$$
$$= \langle \Sigma^\sigma(\cdot, \boldsymbol{z}), \Sigma^\sigma(\cdot, \boldsymbol{z}) \rangle_{\mathbb{H}_{K^\sigma}} - 2 \langle \Sigma^\sigma(\cdot, \boldsymbol{z}), \Sigma^\sigma(\cdot, \boldsymbol{Z}^{\text{train}}) \boldsymbol{C} \Sigma^\sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{z}) \rangle_{\mathbb{H}_{K^\sigma}}$$
$$+ \langle \Sigma^\sigma(\cdot, \boldsymbol{Z}^{\text{train}}) \boldsymbol{C} \Sigma^\sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{z}), \Sigma^\sigma(\cdot, \boldsymbol{Z}^{\text{train}}) \boldsymbol{C} \Sigma^\sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{z}) \rangle_{\mathbb{H}_{K^\sigma}}$$

By the reproducing property it holds that

$$= \Sigma^\sigma(\boldsymbol{z}, \boldsymbol{z}) - 2\Sigma^\sigma(\boldsymbol{z}, \boldsymbol{Z}^{\text{train}}) \boldsymbol{C} \Sigma^\sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{z}) + \Sigma^\sigma(\boldsymbol{z}, \boldsymbol{Z}^{\text{train}}) \boldsymbol{C} \Sigma^\sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{Z}^{\text{train}}) \boldsymbol{C} \Sigma^\sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{z})$$
$$= \Sigma^\sigma(\boldsymbol{z}, \boldsymbol{z}) - \Sigma^\sigma(\boldsymbol{z}, \boldsymbol{Z}^{\text{train}}) \boldsymbol{C} \Sigma^\sigma(\boldsymbol{Z}^{\text{train}}, \boldsymbol{z})$$
$$=: \hat{\Sigma}^\sigma(\boldsymbol{z}, \boldsymbol{z})$$

Now if $\boldsymbol{z} \notin \boldsymbol{Z}^{\text{train}}$ or $\sigma^2 = 0$ it holds that $\Sigma^\sigma(\boldsymbol{z}, \boldsymbol{Z}^{\text{train}}) = \Sigma(\boldsymbol{z}, \boldsymbol{Z}^{\text{train}})$ and therefore $\hat{\Sigma}^\sigma(\boldsymbol{z}, \boldsymbol{z}) = \hat{\Sigma}(\boldsymbol{z}, \boldsymbol{z}) + \sigma^2$. □

C.2.2 Computation-aware Filtering and Smoothing

Having obtained an error bound for the iteratively approximated batch GP posterior, we now aim to show that the CAKF and CAKS compute precisely the same posterior marginals and thus satisfy the same error bound. We do so by leveraging Lemma C.2 describing how to translate between a (batch) GP regression problem and an equivalent state-space formulation under suitable assumptions on the model.

Proposition C.9 (Connecting (Computation-Aware) Batch Spatio-temporal GP Regression and Filtering and Smoothing)**.** *Consider the following spatiotemporal regression problem over the domain $\mathbb{Z} = [t_0, T] \times \mathbb{X}$. Define a space-time separable Gauss–Markov process $\mathbf{f} \sim \mathcal{GP}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ such that its first component $\mathrm{f} \coloneqq \mathbf{f}_0 \sim \mathcal{GP}(\mu, \Sigma)$ defines a Gaussian process prior for the latent function $f^\star \in \mathbb{H}_\Sigma$, where $\mu(t, \boldsymbol{x}) = \boldsymbol{\mu}_0^t(t) \mu^{\boldsymbol{x}}(\boldsymbol{x})$ and $\Sigma((t, \boldsymbol{x}), (t', \boldsymbol{x}')) = \boldsymbol{\Sigma}_0^t(t, t') \Sigma^{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{x}')$. Assume we are given a training dataset consisting of $N = \sum_{k=1}^K N_k$ inputs $\boldsymbol{Z}^{\text{train}} = ((t_1^{\text{train}}, \boldsymbol{x}_{1,1}^{\text{train}}), \ldots, (t_1^{\text{train}}, \boldsymbol{x}_{1,N_1}^{\text{train}}), \ldots, (t_K^{\text{train}}, \boldsymbol{x}_{K,1}^{\text{train}}), \ldots, (t_K^{\text{train}}, \boldsymbol{x}_{K,N_K}^{\text{train}})) \in \mathbb{Z}^N$ and targets $\boldsymbol{y}^{\text{train}} \in (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K) \in \mathbb{R}^N$. Then for any test input $\boldsymbol{z} = (t, \boldsymbol{x}) \in \mathbb{Z}$ the computation-aware smoother computes the mean $\hat{\mu}^{y^\star}(\boldsymbol{z})$ and variance $\hat{\Sigma}(\boldsymbol{z}, \boldsymbol{z})$ of the marginal distribution of the iteratively approximated batch GP posterior $(\mathrm{f} \mid \boldsymbol{S}^\top \boldsymbol{y}^{\text{train}}) \sim \mathcal{GP}\left(\hat{\mu}^{y^\star}, \hat{\Sigma}\right)$ with*

$$\boldsymbol{S} = \begin{pmatrix} \hat{\boldsymbol{S}}_1 & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \hat{\boldsymbol{S}}_K \end{pmatrix} \in \mathbb{R}^{N \times \sum_{k=1}^K \check{N}_k} \tag{C.7}$$

evaluated at the given test input $\boldsymbol{z}$, i.e.

$$\hat{\boldsymbol{m}}^{\mathrm{s}}(\boldsymbol{z} \mid y^\star)_0 = \hat{\mu}^{y^\star}(\boldsymbol{z}), \quad \textit{and}$$
$$\hat{\boldsymbol{P}}^{\mathrm{s}}(\boldsymbol{z})_{0,0} = \hat{\Sigma}(\boldsymbol{z}, \boldsymbol{z}).$$

If $t \geq t_K^{\text{train}}$ it suffices to run the computation-aware filter.

Proof. Let $\boldsymbol{X} \in \mathbb{X}^{N_{\boldsymbol{X}}}$ be a vector containing all (unique) spatial training points and the spatial test point $\boldsymbol{x}$ as its zeroth component. By Lemma C.2 a space-time separable Gauss–Markov process $\mathbf{f}$ evaluated at the spatial inputs $\boldsymbol{X}$, i.e., $\mathbf{u}(t) \coloneqq \mathbf{f}(t, \boldsymbol{X}) \in \mathbb{R}^{N_{\boldsymbol{X}} D'}$ and $\mathbf{u}_0(t) = \mathrm{f}(t, x)$, admits a state-space representation with dynamics

$$\mathbf{u}(t) = \boldsymbol{A}(t, s) \mathbf{u}(s) + \boldsymbol{b}(t, s) + \mathbf{q}(t, s) \tag{C.8}$$

where $\mathbf{q}(t, s) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}(t, s))$ and the observation model is by assumption given by

$$\check{\mathbf{y}}_k = \hat{\boldsymbol{S}}_k^\top \mathbf{f}_0(t_k^{\text{train}}, \boldsymbol{X}_k^{\text{train}}) + \check{\boldsymbol{\epsilon}}_k = \hat{\boldsymbol{S}}_k^\top \boldsymbol{H}_k \mathbf{u}(t_k^{\text{train}}) + \check{\boldsymbol{\epsilon}}_k \in \mathbb{R}^{\check{N}_k} \tag{C.9}$$

where $\boldsymbol{H}_k \in \mathbb{R}^{N_k \times D}$ is implicitly defined by $\boldsymbol{H}_k \mathbf{u}(t_k^{\text{train}}) = \mathbf{f}_0(t_k^{\text{train}}, \boldsymbol{X}_k^{\text{train}})$.

Now, in a linear Gaussian state-space model as defined by Equation (C.8) and Equation (C.9), the vanilla Kalman filter and smoother (Särkkä, 2006, Alg. 3.14 & 3.17) compute the posterior marginal

$$(\mathbf{f}(t, \boldsymbol{X}) \mid \check{\mathbf{y}}_1 = \check{\boldsymbol{y}}_1, \dots, \check{\mathbf{y}}_K = \check{\boldsymbol{y}}_K) \sim \mathcal{N}\left(\hat{\boldsymbol{\mu}}^{y^\star}(t, \boldsymbol{X}), \hat{\boldsymbol{\Sigma}}((t, \boldsymbol{X}), (t, \boldsymbol{X}))\right) \tag{C.10}$$

at an arbitrary timepoint t exactly, and if $t \geq t_K^{\text{train}}$, then it suffices to run the filter (Särkkä and Solin, 2019, Alg. 10.15 & 10.18). By construction, the output of the computation-aware filter and smoother for given $\hat{\boldsymbol{S}}_1, \dots, \hat{\boldsymbol{S}}_K$ (assuming no truncation) are equivalent to applying the vanilla filter and smoother to the state-space model defined by Equation (C.8) and Equation (C.9), i.e., $\hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z} \mid y^\star) = \hat{\boldsymbol{\mu}}^{y^\star}(\boldsymbol{z})$ and $\hat{\boldsymbol{P}}^{\text{s}}(\boldsymbol{z}) = \hat{\boldsymbol{\Sigma}}(\boldsymbol{z}, \boldsymbol{z})$. By Lemma C.7, the zeroth components of the CAKS moments $(\hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z}))_0, (\hat{\boldsymbol{P}}^{\text{s}}(\boldsymbol{z}))_{0,0}$ are hence equal to the marginal moments of the iteratively approximated batch GP posterior $(\mathrm{f} \mid \boldsymbol{S}^\top \boldsymbol{y}^{\text{train}}) \sim \mathcal{GP}\left(\hat{\mu}^{y^\star}, \hat{\Sigma}\right)$ evaluated at the test point $\boldsymbol{z} = (t, \boldsymbol{x})$. This completes the proof. □

Theorem 1 (Pointwise Worst-Case Prediction Error)**.** *Let $\mathbb{Z} = [t_0, T] \times \mathbb{X}$ and define a space-time separable Gauss–Markov process $\mathbf{f} \sim \mathcal{GP}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ such that its first component $\mathrm{f} \coloneqq \mathbf{f}_0 \sim \mathcal{GP}(\mu, \Sigma)$ defines a prior for the latent function $f^\star \in \mathbb{H}_\Sigma$ generating the data, assumed to be an element of the RKHS defined by $\mathbb{H}_\Sigma$. Given observation noise $\sigma^2 \geq 0$, let $y^\star(\cdot) \in \mathbb{H}_{\Sigma^\sigma}$ be the observed process with $\Sigma^\sigma(\boldsymbol{z}, \boldsymbol{z}') \coloneqq \Sigma(\boldsymbol{z}, \boldsymbol{z}') + \sigma^2 \delta(\boldsymbol{z}, \boldsymbol{z}')$. Given training inputs $\boldsymbol{Z}^{\text{train}} \in \mathbb{Z}^N$ and targets $\boldsymbol{y}^{\text{train}} = y^\star(\boldsymbol{Z}^{\text{train}}) \in \mathbb{R}^N$, let $\hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z} \mid y^\star)$ and $\hat{\boldsymbol{P}}^{\text{s}}(\boldsymbol{z})$ be the mean and variance of the CAKS for an arbitrary test input $\boldsymbol{z} = (t, \boldsymbol{x}) \in \mathbb{Z} \setminus \boldsymbol{Z}^{\text{train}}$. Then*

$$\sup_{y \in \mathbb{H}_{\Sigma^\sigma} \setminus \{0\}} \frac{|y(\boldsymbol{z}) - \hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z} \mid y)_0|}{\|y\|_{\mathbb{H}_{\Sigma^\sigma}}} = \sqrt{\hat{\boldsymbol{P}}^{\text{s}}(\boldsymbol{z})_{0,0} + \sigma^2}. \tag{5.1}$$

If $\sigma^2 = 0$, this also holds for training inputs $\boldsymbol{z} \in \boldsymbol{Z}^{\text{train}}$.

Proof. By Proposition C.9 the mean $\hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z} \mid y^\star)_0$ and variance $\hat{\boldsymbol{P}}^{\text{s}}(\boldsymbol{z}, \boldsymbol{z})_{0,0}$ computed by the computation-aware filter and smoother for the test input $\boldsymbol{z} = (t, \boldsymbol{x})$ are equivalent to the marginal posterior mean and variance of an iteratively approximated batch GP posterior with the induced (space-time separable) prior $\mathrm{f} \coloneqq \mathbf{f}_0 \sim \mathcal{GP}(\mu, \Sigma)$ and actions $\boldsymbol{S}$ defined as in Equation (C.7). Therefore by Theorem C.8 it holds that

$$\sup_{\substack{\tilde{y} \in \mathbb{H}_{\Sigma^\sigma} \\ \|\tilde{y}\|_{\mathbb{H}_{\Sigma^\sigma}} \leq 1}} |\tilde{y}(\boldsymbol{z}) - \hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z} \mid \tilde{y})_0| = \sqrt{\hat{\boldsymbol{P}}^{\text{s}}(\boldsymbol{z}, \boldsymbol{z})_{0,0} + \sigma^2}$$

Recognizing that the supremum is achieved on the boundary, i.e., where $\|\tilde{y}\|_{\mathbb{H}_{\Sigma^\sigma}} = 1$, we can equivalently consider $\tilde{y}(\cdot) = \frac{y(\cdot)}{\|y\|_{\mathbb{H}_{\Sigma^\sigma}}}$ for $y \in \mathbb{H}_{\Sigma^\sigma} \setminus \{0\}$ arbitrary. Then it holds that $\hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z} \mid \tilde{y})_0 = \frac{\hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z} \mid y)_0}{\|y\|_{\mathbb{H}_{\Sigma^\sigma}}}$ and therefore we have

$$\sup_{y \in \mathbb{H}_{\Sigma^\sigma} \setminus \{0\}} \frac{|y(\boldsymbol{z}) - \hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z} \mid y)_0|}{\|y\|_{\mathbb{H}_{\Sigma^\sigma}}} = \sup_{\substack{\tilde{y} \in \mathbb{H}_{\Sigma^\sigma} \\ \|\tilde{y}\|_{\mathbb{H}_{\Sigma^\sigma}} = 1}} |\tilde{y}(\boldsymbol{z}) - \hat{\boldsymbol{m}}^{\text{s}}(\boldsymbol{z} \mid \tilde{y})_0| = \sqrt{\hat{\boldsymbol{P}}^{\text{s}}(\boldsymbol{z}, \boldsymbol{z})_{0,0} + \sigma^2}$$

This proves the claim. □

D EXPERIMENTS

D.1 Metrics

To assess the predictive performance of the models considered for our experiments in Section 7, we mainly use two different metrics:

Mean Squared Error To assess the quality of the predictive mean $\tilde{\boldsymbol{m}}_k$ as a point estimate for the target state $\boldsymbol{u}_k^\star$, we use the mean squared error (MSE)

$$\text{MSE}(\boldsymbol{u}_k^\star, \tilde{\boldsymbol{m}}_k) = \frac{1}{D} \|\boldsymbol{u}_k^\star - \tilde{\boldsymbol{m}}_k\|_2^2.$$

Average Negative Log Density All inference algorithms considered in this work provide Gaussian conditional distributions $\mathcal{N}\left(\tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{P}}_k\right)$ as estimates of the unknown target state $\boldsymbol{u}_k^\star$. Hence, we can use the average negative log density (NLD)

$$\begin{aligned}\operatorname{AvgNLD}(\boldsymbol{u}_k^\star \mid \tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{P}}_k) &= -\frac{1}{D}\sum_{d=1}^{D} \log \mathcal{N}\left(\boldsymbol{u}_{k,d}^\star; \tilde{\boldsymbol{m}}_{k,d}, \tilde{\boldsymbol{P}}_{k,d,d}\right) \\ &= \frac{1}{2D}\sum_{d=1}^{D} \frac{(\boldsymbol{u}_{k,d}^\star - \tilde{\boldsymbol{m}}_{k,d})^2}{\tilde{\boldsymbol{P}}_{k,d,d}} + \log(2\pi\tilde{\boldsymbol{P}}_{k,d,d})\end{aligned}$$

corresponding to this Gaussian distribution as a measure of the quality of the predictive uncertainty. We use the average NLD as opposed to the "full" NLD, i.e., $-\log \mathcal{N}\left(\boldsymbol{u}_k^\star; \tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{P}}_k\right)$, since computation of the latter is infeasible for high-dimensional state spaces.

For both metrics, we choose $(\tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{P}}_k) \in \{(\hat{\boldsymbol{m}}_k, \hat{\boldsymbol{P}}_k), (\hat{\boldsymbol{m}}_k^{\mathrm{s}}, \hat{\boldsymbol{P}}_k^{\mathrm{s}}), (\boldsymbol{m}_k, \boldsymbol{P}_k), (\boldsymbol{m}_k^{\mathrm{s}}, \boldsymbol{P}_k^{\mathrm{s}}), \ldots\}$ as appropriate. We aggregate the metrics across entire trajectories by averaging. In Figures 4 and D.3, we also marginalize over the train and test parts of the states before computing the metrics to obtain a more fine-grained view of the performance of the algorithms.

D.2 Experiment Details

We provide some additional details for the experiments conducted in Section 7 in this section.

D.2.1 Comparison to Other Methods

The domain of the problem is $[0, T] \times \mathbb{X}$ with $\mathbb{X} = [0, 20]^{D_{\mathbb{X}}}$.

Model The temporal and spatial covariance functions are given by $\Sigma^t = \sigma^2 \cdot \text{Matérn}(3/2, \ell_t)$ and $\Sigma^{\boldsymbol{x}} = \text{Matérn}(3/2, \ell_{\boldsymbol{x}})$ with lengthscales $\ell_t = \ell_{\boldsymbol{x}} = 0.5$ and output scale $\sigma = 1$.

Data We discretize the model on regular grids $\boldsymbol{X} \in \mathbb{X}^{N_{\boldsymbol{X}}}$ and $\boldsymbol{t} \in [0, T]^{N_{\boldsymbol{t}}}$ of $N_{\boldsymbol{X}} = 100^{D_{\mathbb{X}}}$ points in space and $N_{\boldsymbol{t}} = 20 \cdot T$ points in time. Afterwards, we sample a ground-truth trajectory $\{\boldsymbol{u}_k^\star\}_{k=1}^{N_{\boldsymbol{t}}}$ from the resulting state-space model. Drawing samples from the SSM requires access to (left) square roots of the initial covariance matrix and the transition noise covariance matrices of the discretized model. As shown in Appendix C.1, these matrices are Kronecker products of a $\mathbb{R}^{2\times 2}$ matrices with the matrix $\Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{N_{\boldsymbol{X}} \times N_{\boldsymbol{X}}}$. Hence, we can compute the required (left) square roots by computing a (left) square root of $\Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})$. Since a (left) square root of $\Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})$ is not available in closed form, we compute it by means of an eigendecomposition (on the GPU). However, this severely limits the number of spatial points $N_{\boldsymbol{X}}$ that can be considered in the experiments. We pick random subsets $\boldsymbol{t}^{\text{train}} \subset \boldsymbol{t}$ and $\boldsymbol{X}^{\text{train}} \subset \boldsymbol{X}$ of $K = N_{\boldsymbol{t}}/10$ temporal and $N_k = 20^{D_{\mathbb{X}}}$ spatial points as training locations $\boldsymbol{Z} = \boldsymbol{t}^{\text{train}} \times \boldsymbol{X}^{\text{train}}$ and perturb the corresponding entries of the ground-truth states $\boldsymbol{u}_k^\star$ with additive i.i.d. Gaussian measurement noise with standard deviation $\lambda = 0.1$.

Evaluation We repeat the data sampling procedure outlined above five times with different random seeds and evaluate the performance of each inference algorithm independently on each of the resulting datasets. The solid lines in Figures 2, D.1 and D.2 are obtained by taking the median over the resulting metrics (and the wall-time in case of Figure 2) across the different runs while the individual values are scattered in the background with reduced opacity. For the CAKF/CAKS and the ensemble Kalman filters, we study the effect of the computational budget on the approximation by varying the rank parameter r. For a fair comparison, we fix both the number of iterations and the maximal rank after truncation in the CAKF/CAKS to the same values used in the ensemble Kalman filters, i.e., $\check{N}_k^{\max} = r_k^{\max} = r$.

Comparison to EnKF and ETKF In this experiment, we consider a two spatial dimensions, i.e., $D_{\mathbb{X}} = 2$, and set $T = 5$. This results in a state-space dimension of $D = 20\,000$, $N_{\boldsymbol{t}} = 100$ total time steps taken, and $N_k = 400$ spatial observations made at each of the $K = 10$ training time points.

The ensemble in the EnKF is initialized by drawing r samples from the initial state $\mathbf{u}_1$ of the model. In the predict step, we draw one new ensemble member from the transition model starting at the corresponding ensemble member in the previous ensemble. The ETKF-S uses the same sample-based initialization and prediction steps as

Table D.1: Total number of spatial points $N_{\boldsymbol{X}} = |\boldsymbol{X}|$, state-space dimension D, number of spatial training points per time step $N_k = |\boldsymbol{X}_k^{\text{train}}|$, and total number of training points $N = \sum_{k=1}^{K} N_k$ for the ERA5 experiment.

Downsampling factor	$N_{\boldsymbol{X}}$	D	N_k	N
$(1/24)^2$	1860	3720	1440	69 120
$(1/12)^2$	7320	14 640	5580	267 840
$(1/6)^2$	29 040	58 080	21 960	1 054 080
$(1/3)^2$	115 680	231 360	87 120	4 181 760

the EnKF. Both algorithms leverage the (left) square root of $\Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})$ computed during data generation at every step, and the wall time duration of its computation is added to the wall time cost of both algorithms in Figure 2. The ETKF-L uses the Lanczos process to compute a low-rank approximation to the initial covariance during initialization and to the predictive covariance during the prediction step. The random number generators in the ensemble Kalman filters is seeded with a deterministic transformation of the data seed.

Comparison to Kalman filter and RTS smoother In this experiment, we consider a one spatial dimension, i.e., $D_{\mathbb{X}} = 1$, and set $T = 50$. This results in a state-space dimension of $D = 200$, $N_{\boldsymbol{t}} = 1000$ total time steps taken, and $N_k = 20$ spatial observations made at each of the $K = 100$ training time points. Moreover, we fix one random set of training time points that is shared across runs to make the trajectories comparable.

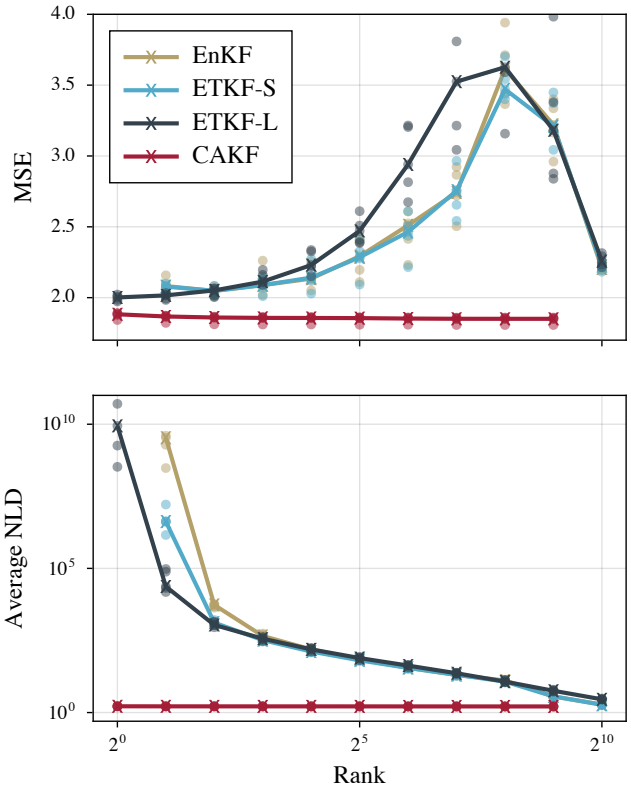


Figure D.1: Comparison of the CAKF, the EnKF, and two variants of the ETKF on on-model data while varying the rank parameters that govern the computational budget of the algorithms.

D.2.2 Impact of Truncation

The temporal domain of the problem is $[0, 1]$, while the spatial domain is $[0, \pi]$.

Data We generate the synthetic data on a regular grid of size 11×16 (in time and space) and the plots are generated on a 51×158 regular grid. The training data is corrupted by i.i.d. zero-mean Gaussian measurement noise with standard deviation $\lambda = 0.1$.

Model The temporal and spatial covariance functions are given by $\Sigma^t = \sigma^2 \cdot \text{Matérn}(3/2, \ell_t)$ and $\Sigma^{\boldsymbol{x}} = \text{Matérn}(5/2, \ell_{\boldsymbol{x}})$ with lengthscales $\ell_t = 0.5$ and $\ell_{\boldsymbol{x}} = 2$, respectively, and an output scale $\sigma = 1$. The resulting state-space dimension of the spatially discretized model is $D = 348$.

D.2.3 Large-Scale Climate Dataset

The domain of the problem is $[0\,\text{h}, 48\,\text{h}] \times \mathbb{S}^2(r_\oplus)$, where $\mathbb{S}^2(r)$ denotes the two-dimensional sphere with radius $r \in \mathbb{R}_{\geq 0}$, and $r_\oplus \approx 6371\,\text{km}$ is the average Earth radius.

Model The temporal covariance function is given by $\Sigma^t = \sigma^2 \cdot \text{Matérn}(3/2, \ell_t)$ with lengthscale $\ell_t = 3\,\text{h}$ and output scale $\sigma = 10\,^\circ\text{C}$. The spatial covariance function is chosen as an extrinsic $\text{Matérn}(3/2, \ell_{\boldsymbol{x}})$ covariance function on $\mathbb{S}^2(r_\oplus)$, i.e., a $\text{Matérn}(3/2, \ell_{\boldsymbol{x}})$ covariance function on $\mathbb{R}^3$ concatenated with a coordinate transformation from geographic coordinates to $\mathbb{R}^3$ in both arguments. For any given spatial downsampling factor, its lengthscale $\ell_{\boldsymbol{x}}$ is set to the geodesic distance of the training points at the equator. We assume the data is corrupted by iid Gaussian noise with standard deviation $0.1\,^\circ\text{C}$. These hyperparameters were chosen a priori and not tuned for the given training data.

We provide the total number of spatial points, state-space dimension, number of spatial training points per time step, and total number of training points for every downsampling factor of the ERA5 experiment in Table D.1.

D.3 Policy Choice

In general, the choice of optimal policy may be highly problem-dependent, but some natural choices present themselves.

Coordinate Actions The simplest choice of policy produces a sequence of unit vectors with all zero entries, except for a single coordinate $j(i)$, i.e., $\hat{\boldsymbol{s}}_k^{(i)} = \boldsymbol{e}_{j(i)}$. Choosing $j(i) = i$ simply corresponds to sequential conditioning on a subset of the components of $\boldsymbol{y}_k$, which in the spatiotemporal regression setting correspond to a subset of spatial locations. When the data has spatial structure, e.g., when it is placed on a grid as in Section 7.3, this structure can be leveraged by choosing a space-filling sequence of points. Berberidis and Giannakis (2017) similar to our work explored low-dimensional projections to accelerate Kalman filtering. They propose several effective policy choices, among them a coordinate policy, which is based on a computable measure of the amount of information in each component of $\boldsymbol{y}_k$, allowing one to select informative points sequentially while omitting uninformative data points.

Randomized Actions Berberidis and Giannakis (2017) also proposed using randomized actions inspired by *sketching* techniques in randomized numerical linear algebra (Martinsson and Tropp, 2020). For example, a common choice are actions with i.i.d. sampled entries, e.g., $\hat{\boldsymbol{s}}_k^{(i)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

Bayesian Experimental Design Another choice is to use Bayesian experimental design (see e.g., Berger, 1980). This has been explored before in the context of probabilistic linear solvers (Cockayne et al., 2019a) and was found to suffer from slow convergence since these are *a-priori* optimal and therefore do not adapt well to the specific problem. Furthermore, such optimal actions are not always tractable to compute.

CG/Lanczos Actions Finally, a choice that has been repeatedly proposed in the literature on probabilistic linear solvers (Wenger et al., 2022; Cockayne et al., 2019a; Hennig, 2015; Wenger and Hennig, 2020) is the Lanczos/CG algorithm (Saad, 2003, Section 6.6). In this approach, we obtain the vectors $\hat{\boldsymbol{s}}_k^{(1)}, \ldots, \hat{\boldsymbol{s}}_k^{(\tilde{N}_k)}$ by appling the Lanczos procedure to the matrix $\hat{\boldsymbol{G}}_k = \boldsymbol{H}_k \hat{\boldsymbol{P}}_k^- \boldsymbol{H}_k^\top + \boldsymbol{\Lambda}$, with initial vector $\hat{\boldsymbol{r}}_k^{(0)} = \boldsymbol{y}_k - \boldsymbol{H}_k \hat{\boldsymbol{m}}_k^-$. This has the effect of ensuring that the residuals $\hat{\boldsymbol{r}}_k^{(i)} \to 0$ at an exponential rate in i (see e.g., Liesen and Strakos, 2012, Corollary 5.6.7). As shown by Wenger et al. (2022, Cor. S2), this choice is equivalent to directly selecting the current residual as the next action, i.e., $\hat{\boldsymbol{s}}_k^{(i)} = \hat{\boldsymbol{r}}_k^{(i)}$.

D.3.1 Empirical Comparison of Policies

To empirically compare some of the proposed policy choices for the CAKF and CAKS, we rerun the experiment on the ERA5 climate dataset with a downsampling factor of 12 ($D = 14\,640$, orange line in Figure 4) using three different policies selected from the above choices. We compare coordinate actions with coordinates corresponding to spatial locations chosen according to an (approximately) space-filling design, random actions obtained by drawing independent samples from a standard normal distribution, i.e., $\hat{\boldsymbol{s}}_k^{(i)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and finally CG/Lanczos actions given by $\hat{\boldsymbol{s}}_k^{(i)} = \hat{\boldsymbol{r}}_k^{(i)}$. The corresponding work-precision diagrams are shown in Figure D.3. It shows that CG actions are preferable to the other actions considered in terms of the MSE and average NLD achieved on both train and test sets. Note that the blue lines in Figure D.3 coincide with the orange lines in Figure 4. In particular, the exponential convergence rate of CG can be seen in the top-left panel (the CG MSE terminates just below 10^{-4} for 2^8 iterations per time step as can be seen in Figure 4).

D.4 On Comparison with the EnKF

While the ensemble Kalman filter (EnKF) (Evensen, 1994) and variants such as the ensemble adjustment Kalman filter (EAKF) (Anderson, 2001) and the ensemble transform Kalman filter (ETKF) (Bishop et al., 2001) can sometimes be used to solve filtering problems with high-dimensional state spaces, they typically make strong assumptions about the state-space models that are not fulfilled in the setting of this paper. To be precise,

ensemble Kalman filters often assume that we can sample from the initial state $\mathbf{u}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and the process noise $\mathbf{q}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_k)$ in the dynamics model (Burgers et al., 1998, Eqn. 18) and/or that it is feasible to compute (left) square roots of the respective covariances $\boldsymbol{\Sigma}_0$ and $\boldsymbol{Q}_k$.

In the data assimilation context, it is common that either:

- $\boldsymbol{Q}_k = \mathbf{0}$
- $\boldsymbol{Q}_k$ is diagonal, or
- $\boldsymbol{Q}_k$ is low-rank with given left square root.

In separable spatiotemporal GP regression, we have $\boldsymbol{Q}_k = \boldsymbol{Q}_k^t \otimes \Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})$, where $\Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})$ is a dense kernel Gram matrix, whose (left) square root is (generally) not available without allocating $\Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})$ in memory. This means that most ensemble Kalman filters would still require computing a Cholesky factorisation of $\Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})$, and would thus have $\mathcal{O}(N_{\boldsymbol{X}}^3)$ time and $\mathcal{O}(N_{\boldsymbol{X}}^2)$ memory complexities, which are precisely the complexities that the CAKF/S are constructed to avoid. We encounter analogous issues when considering $\boldsymbol{\Sigma}_0$.

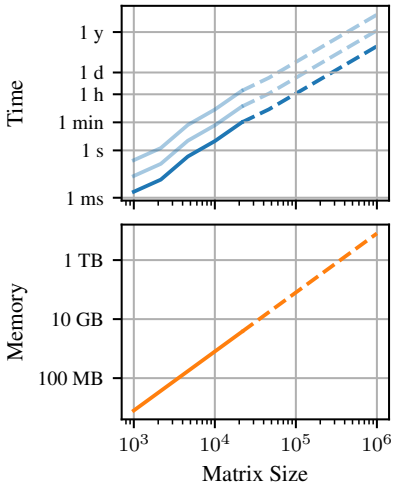
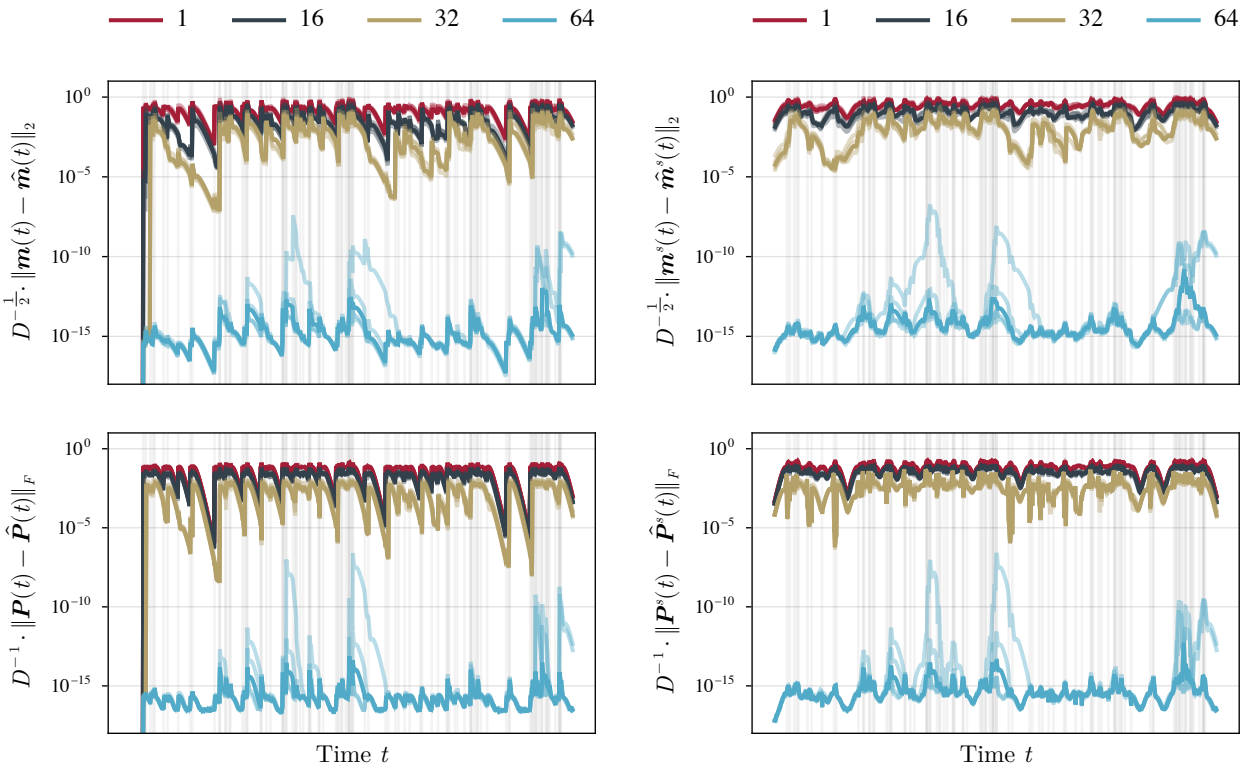


Figure D.4: Computational time and memory cost of a Cholesky decomposition in double precision as a function of matrix size.

To make this point more clearly, Figure D.4 shows how time and memory requirements to compute a Cholesky decomposition of $\Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})$ scale with matrix size $N_{\boldsymbol{X}}$. The solid lines correspond to what decompositions we are able to compute on our machines for the different problem sizes in Table D.1, and the dashed lines are extrapolations past the problem sizes where the decomposition crashed due to memory allocation errors. The largest problem size we apply our methods to has $N_{\boldsymbol{X}} = 115\,680$ spatial observations, which is comfortably outside the range where a Cholesky decomposition can be computed on all but the most specialised hardware; just to allocate this matrix would require in excess of 100 GB of working memory.

An exception to the above is the seemingly seldom-used ETKF-L algorithm described in (Tippett et al., 2003, Sec. 3(b)) as well as in Section 7.1.1 and Appendix D.2.1. The ETKF-L approximates the required (left) square roots with the Lanczos method without allocating $\Sigma^{\boldsymbol{x}}(\boldsymbol{X}, \boldsymbol{X})$ in memory.



(a) Filter (b) Smoother

Figure D.2: Error dynamics of the CAKF and CAKS with varying rank parameter $\check{N}_k^{\max} = r_k^{\max} = 1, 16, 32, 64$ compared to the Kalman filter and RTS smoother on on-model data. The time points at which data is observed are marked by the vertical grid lines.

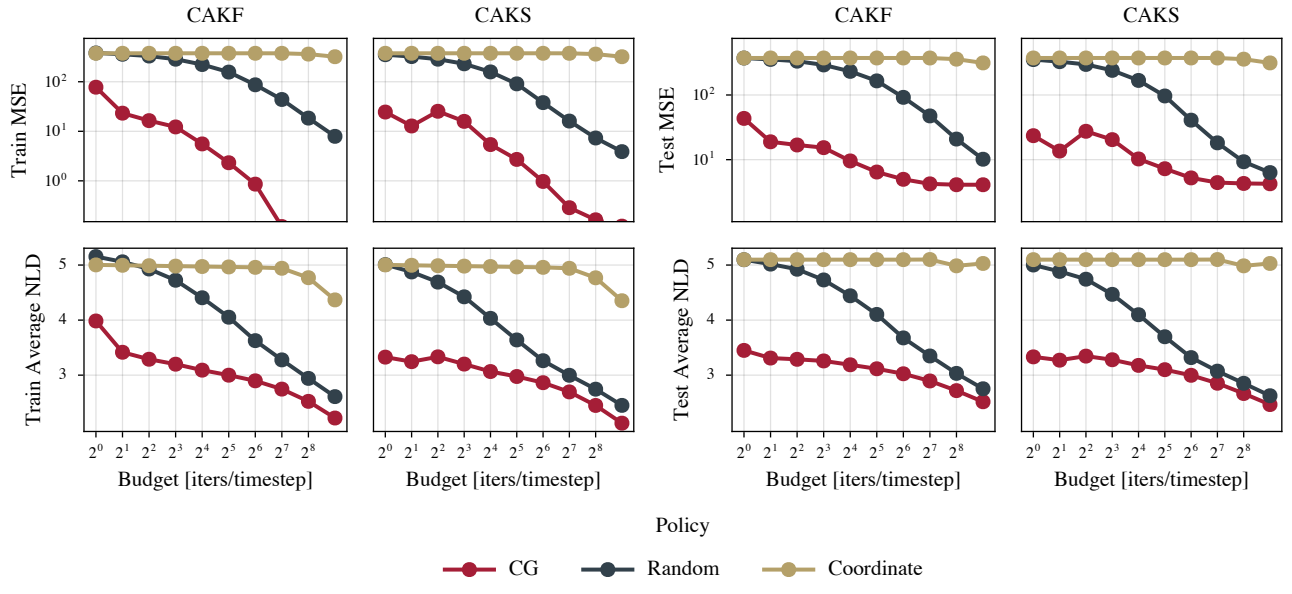


Figure D.3: *Comparison of different policies for the CAKF and CAKS on the ERA5 climate dataset.* The work-precision diagrams measuring MSE and average NLD on the train and test set universally show that CG actions achieve lower error as a function of the budget when compared to either coordinate or random actions.