Appendix

A Spectral Analysis and LTI-SDE

We consider the Matérn kernel family,

$$\kappa_\nu(t,t') = a\frac{\left(\frac{\sqrt{2\nu}}{\rho}\Delta\right)^\nu}{\Gamma(\nu)2^{\nu-1}} K_\nu\left(\frac{\sqrt{2\nu}}{\rho}\Delta\right) \tag{1}$$

where $\Delta = |t-t'|$, $\Gamma(\cdot)$ is the Gamma function, $a>0$ and $\rho>0$ are the amplitude and length-scale parameters, respectively, K_ν is the modified Bessel function of the second kind, $\nu>0$ controls the smoothness. Since κ_ν is a stationary kernel, *i.e.,* $\kappa_\nu(t,t') = \kappa_\nu(t-t')$, according to the Wiener-Khinchin theorem [Chatfield, 2003], if

$$f(t) \sim \mathcal{GP}(0, \kappa_\nu(t,t')),$$

the energy spectrum density of $f(t)$ can be obtained by the Fourier transform of $\kappa_\nu(\Delta)$,

$$S(\omega) = a\frac{2\sqrt{\pi}\Gamma(\frac{1}{2}+\nu)}{\Gamma(\nu)}\alpha^{2\nu}\left(\alpha^2+\omega^2\right)^{-(\nu+\frac{1}{2})} \tag{2}$$

where ω is the frequency, and $\alpha = \frac{\sqrt{2\nu}}{\rho}$. We consider the commonly used choice $\nu = p+\frac{1}{2}$ where $p \in \{0,1,2,\ldots\}$. Then we can observe that

$$S(\omega) = \frac{\sigma^2}{(\alpha^2+\omega^2)^{p+1}} = \frac{\sigma^2}{(\alpha+i\omega)^{p+1}(\alpha-i\omega)^{p+1}} \tag{3}$$

where $\sigma^2 = a\frac{2\sqrt{\pi}\Gamma(p+1)}{\Gamma(p+\frac{1}{2})}\alpha^{2p+1}$, and i indicates an imaginary number. We expand the polynomial

$$(\alpha+i\omega)^{p+1} = \sum\nolimits_{k=0}^{p} c_k (i\omega)^k + (i\omega)^{p+1} \tag{4}$$

where $\{c_k | 0 \le k \le p\}$ are the coefficients. From (3) and (4), we can construct an equivalent system to generate the signal $f(t)$. That is, in the frequency domain, the system output's Fourier transform $\widehat{f}(\omega)$ is given by

$$\sum\nolimits_{k=1}^{p} c_k (i\omega)^k \widehat{f}(\omega) + (i\omega)^{p+1}\widehat{f}(\omega) = \widehat{\beta}(\omega) \tag{5}$$

where $\widehat{\beta}$ is the Fourier transform of a white noise process $\beta(t)$ with spectral density (or diffusion) σ^2. The reason is that by construction, $\widehat{f}(\omega) = \frac{\widehat{\beta}(\omega)}{(\alpha+i\omega)^{p+1}}$, which gives exactly the same spectral density as in (3), $S(\omega) = |\widehat{f}(\omega)|^2$. We then conduct inverse Fourier transform on both sides of (5) to obtain the representation in the time domain,

$$\sum\nolimits_{k=1}^{p} c_k \frac{\mathrm{d}^k f}{\mathrm{d}t^k} + \frac{\mathrm{d}^{p+1} f}{\mathrm{d}t^{p+1}} = \beta(t), \tag{6}$$

which is an SDE. Note that $\beta(t)$ has the density σ^2. We can further construct a new state $\mathbf{z} = (f, f^{(1)}, \ldots, f^{(p)})^\top$ (where each $f^{(k)} \triangleq \mathrm{d}^k f/\mathrm{d}t^k$) and convert (6) into a linear time-invariant (LTI) SDE,

$$\frac{\mathrm{d}\mathbf{z}}{\mathrm{d}t} = \mathbf{A}\mathbf{z} + \boldsymbol{\eta}\cdot\beta(t) \tag{7}$$

where

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -c_0 & \ldots & -c_{p-1} & -c_p \end{pmatrix}, \quad \boldsymbol{\eta} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

For a concrete example, if we take $p = 1$ (and so $\nu = \frac{3}{2}$), then $\mathbf{A} = [0, 1; -\alpha^2, -2\alpha]$, $\boldsymbol{\eta} = [0; 1]$, and $\sigma^2 = 4a\alpha^3$.

The LTI-SDE is particularly useful in that its finite set of states follow a Gauss-Markov chain, namely the state-space prior. Specifically, given arbitrary $t_1 < \ldots < t_L$, we have

$$p(\mathbf{z}(t_1), \ldots, \mathbf{z}(t_L)) = p(\mathbf{z}(t_1)) \prod\nolimits_{k=1}^{L-1} p(\mathbf{z}(t_{k+1})|\mathbf{z}(t_k))$$

where $p(\mathbf{z}(t_1)) = \mathcal{N}(\mathbf{z}(t_1)|\mathbf{0}, \mathbf{P}_\infty)$, $p(\mathbf{z}(t_{k+1})|\mathbf{z}(t_k)) = \mathcal{N}(\mathbf{z}(t_{k+1})|\mathbf{F}_k\mathbf{z}(t_k), \mathbf{Q}_k)$, $\mathbf{P}_\infty$ is the stationary covariance matrix computed by solving the matrix Riccati equation [Lancaster and Rodman, 1995], $\mathbf{F}_n = \exp(\Delta_k \cdot \mathbf{A})$ where $\Delta_k = t_{k+1} - t_k$, and $\mathbf{Q}_k = \mathbf{P}_\infty - \mathbf{A}_k\mathbf{P}_\infty\mathbf{A}_k^\top$. Therefore, we do not need the full covariance matrix as in the standard GP prior, and the computation is much more efficient. The chain structure is also convenient to handle streaming data as we will explain later.

Note that for other type of kernel functions, such as the square exponential (SE) kernel, we can approximate the inverse spectral density $1/S(\omega)$ with a polynomial of ω^2 with negative roots, and follow the same way to construct an LTI-SDE and state-space prior.

B RTS Smoother

Consider a standard state-space model with state $\mathbf{x}_n$ and observation $\mathbf{y}_n$ at each time step n. The prior distribution is a Gauss-Markov chain,

$$p(\mathbf{x}_{n+1}|\mathbf{x}_n) = \mathcal{N}(\mathbf{x}_{n+1}|\mathbf{A}_n\mathbf{x}_n, \mathbf{Q}_n),$$
$$p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0|\mathbf{m}_0, \mathbf{P}_0).$$

Suppose we have a Gaussian observation likelihood,

$$p(\mathbf{y}_n|\mathbf{x}_n) = \mathcal{N}(\mathbf{y}_n|\mathbf{H}_n\mathbf{x}_n, \mathbf{W}_n).$$

Then upon receiving each $\mathbf{y}_n$, we can use Kalman filtering to obtain the exact running posterior,

$$p(\mathbf{x}_n|\mathbf{y}_{1:n}) = \mathcal{N}(\mathbf{x}_n|\mathbf{m}_k, \mathbf{P}_k)$$

which is a Gaussian. After all the data has been processed — suppose it ends after step N — we can use Rauch–Tung–Striebel (RTS) smoother [Särkkä, 2013] to efficiently compute the full posterior of each state from backward, which does not need to re-access any data: $p(\mathbf{x}_n|\mathbf{y}_{1:N}) = \mathcal{N}(\mathbf{x}_n|\mathbf{m}_n^s, \mathbf{P}_n^s)$, where

$$\begin{aligned}
\mathbf{m}_{n+1}^- &= \mathbf{A}_n\mathbf{m}_n, \;\; \mathbf{P}_{n+1}^- = \mathbf{A}_n\mathbf{P}_n\mathbf{A}_n^\top + \mathbf{Q}_n, \\
\mathbf{G}_n &= \mathbf{P}_n\mathbf{A}_n^\top[\mathbf{P}_{n+1}^-]^{-1}, \\
\mathbf{m}_n^s &= \mathbf{m}_n + \mathbf{G}_n\left(\mathbf{m}_{n+1}^s - \mathbf{m}_{n+1}^-\right), \\
\mathbf{P}_n^s &= \mathbf{P}_n + \mathbf{G}_n[\mathbf{P}_{n+1}^s - \mathbf{P}_{n+1}^-]\mathbf{G}_n^\top.
\end{aligned} \tag{8}$$

As we can see, the computation only needs the running posterior $p(\mathbf{x}_n|\mathbf{y}_{1:n}) = \mathcal{N}(\cdot|\mathbf{m}_n, \mathbf{P}_n)$ and the full posterior of the next state $p(\mathbf{x}_{n+1}|\mathbf{y}_{1:N}) = \mathcal{N}(\cdot|\mathbf{m}_{n+1}, \mathbf{P}_{n+1})$. It does not need to revisit previous observations $\mathbf{y}_{1:N}$

C Details about Online Trajectory Inference

In this section, we provide the details about how to update the running posterior according to equation (8) and (9) (in the main paper) with the conditional EP (CEP) framework [Wang and Zhe, 2019].

C.1 EP and CEP framework

We first give a brief introduction to the EP and CEP framework. Consider a general probabilistic model with latent parameters $\boldsymbol{\theta}$. Given the observed data $\mathcal{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, the joint probability distribution is

$$p(\boldsymbol{\theta}, \mathcal{D}) = p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\mathbf{y}_n|\boldsymbol{\theta}). \tag{9}$$

Our goal is compute the posterior $p(\boldsymbol{\theta}|\mathcal{D})$. However, it is usually infeasible to compute the exact the marginal distribution $p(\mathcal{D})$, because of the complexity of the likelihood and/or prior. EP therefore seeks to approximate each term in the joint probability by an exponential-family term,

$$p(y_n|\boldsymbol{\theta}) \approx c_n f_n(\boldsymbol{\theta}), \quad p(\boldsymbol{\theta}) \approx c_0 f_0(\boldsymbol{\theta}) \tag{10}$$

where c_n and c_0 are constants to ensure the normalization consistency (they will get canceled in the inference, so we do not need to calculate them), and

$$f_n(\boldsymbol{\theta}) \propto \exp(\boldsymbol{\lambda}_n^\top \boldsymbol{\phi}(\boldsymbol{\theta}))(0 \leq n \leq N)$$

where $\boldsymbol{\lambda}_n$ is the natural parameter and $\boldsymbol{\phi}(\boldsymbol{\theta})$ is sufficient statistics. For example, if we choose a Gaussian term, $f_n = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$, then the sufficient statistics is $\boldsymbol{\phi}(\boldsymbol{\theta}) = \{\boldsymbol{\theta}, \boldsymbol{\theta}\boldsymbol{\theta}^\top\}$. The moment is the expectation of the sufficient statistics.

We therefore approximate the joint probability with

$$p(\boldsymbol{\theta}, \mathcal{D}) = p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\mathbf{y}_n|\boldsymbol{\theta}) \approx f_0(\boldsymbol{\theta}) \prod_{n=1}^{N} f_n(\boldsymbol{\theta}) \cdot \text{const} \tag{11}$$

Because the exponential family is closed under product operations, we can immediately obtain a closed-form approximate posterior $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D})$ by merging the approximation terms in the RHS of (11), which is still a distribution in the exponential family.

Then the task amounts to optimizing those approximation terms $\{f_n(\boldsymbol{\theta})|0 \leq n \leq N\}$. EP repeatedly conducts four steps to optimize each f_n.

- **Step 1.** We obtain the calibrated distribution that integrates the context information of f_n,

  $$q^{\backslash n}(\boldsymbol{\theta}) \propto \frac{q(\boldsymbol{\theta})}{f_n(\boldsymbol{\theta})}$$

  where $q(\boldsymbol{\theta})$ is the current posterior approximation.
- **Step 2.** We construct a tilted distribution to combine the true likelihood,

  $$\widetilde{p}(\boldsymbol{\theta}) \propto q^{\backslash n}(\boldsymbol{\theta}) \cdot p(\mathbf{y}_n|\boldsymbol{\theta})$$

  Note that if $n = 0$, we have $\widetilde{p}(\boldsymbol{\theta}) \propto q^{\backslash n}(\boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})$.
- **Step 3.** We project the tilted distribution back to the exponential family,

  $$q^*(\boldsymbol{\theta}) = \underset{q}{\operatorname{argmin}} \quad \mathrm{KL}(\widetilde{p}\|q)$$

  where q belongs to the exponential family. This can be done by moment matching,

  $$\mathbb{E}_{q^*}[\boldsymbol{\phi}(\boldsymbol{\theta})] = \mathbb{E}_{\widetilde{p}}[\boldsymbol{\phi}(\boldsymbol{\theta})]. \tag{12}$$

  That is, we compute the expected moment under $\widetilde{p}$, with which to obtain the parameters of q^*. For example, if $q^*(\boldsymbol{\theta})$ is a Gaussian distribution, then we need to compute $\mathbb{E}_{\widetilde{p}}[\boldsymbol{\theta}]$ and $\mathbb{E}_{\widetilde{p}}[\boldsymbol{\theta}\boldsymbol{\theta}^\top]$, with which to obtain the mean and covariance for $q^*(\boldsymbol{\theta})$. Hence we obtain $q^*(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbb{E}_{\widetilde{p}}[\boldsymbol{\theta}], \mathbb{E}_{\widetilde{p}}[\boldsymbol{\theta}\boldsymbol{\theta}^\top] - \mathbb{E}_{\widetilde{p}}[\boldsymbol{\theta}]\mathbb{E}_{\widetilde{p}}[\boldsymbol{\theta}]^\top)$
- **Step 4.** We update the approximation term by

  $$f_n(\boldsymbol{\theta}) \approx \frac{q^*(\boldsymbol{\theta})}{q^{\backslash}(\boldsymbol{\theta})}. \tag{13}$$

In practice, EP often updates all the f_n's in parallel, and uses damping to avoid divergence. It iteratively runs the four steps until convergence. In essence, this is a fixed point iteration to optimize a free energy function (a mini-max problem) [Minka, 2001].

The critical step in EP is the moment matching (12). However, in many cases, it is analytically intractable to compute the moment under the tilted distribution $\widetilde{p}$, due to the complexity of the likelihood. To address this problem, CEP considers the commonly used case that each f_n has a factorized structure,

$$f_n(\boldsymbol{\theta}) = \prod_m f_{nm}(\boldsymbol{\theta}_m) \tag{14}$$

where each f_{nm} is also in the exponential family, and $\{\boldsymbol{\theta}_m\}$ are mutually disjoint. Then at the moment matching step, we need to compute the moment of each $\boldsymbol{\theta}_m$ under $\widetilde{p}$, *i.e.,* $\mathbb{E}_{\widetilde{p}}[\boldsymbol{\phi}(\boldsymbol{\theta}_m)]$. The first key idea of CEP is to use the nested structure,

$$\mathbb{E}_{\widetilde{p}}[\boldsymbol{\phi}(\boldsymbol{\theta}_m)] = \mathbb{E}_{\widetilde{p}(\boldsymbol{\theta}_{\backslash m})}\mathbb{E}_{\widetilde{p}(\boldsymbol{\theta}_m|\boldsymbol{\theta}_{\backslash m})}[\boldsymbol{\phi}(\boldsymbol{\theta}_m)] \tag{15}$$

where $\boldsymbol{\theta}_{\backslash m} = \boldsymbol{\theta}\backslash\boldsymbol{\theta}_m$. Therefore, we can first compute the inner expectation, *i.e.,* conditional moment,

$$\mathbb{E}_{\widetilde{p}(\boldsymbol{\theta}_m|\boldsymbol{\theta}_{\backslash m})}[\boldsymbol{\phi}(\boldsymbol{\theta}_m)] = \mathbf{g}(\boldsymbol{\theta}_{\backslash m}), \tag{16}$$

and then seek for computing the outer expectation, $\mathbb{E}_{\widetilde{p}(\boldsymbol{\theta}_{\backslash m})}[\mathbf{g}(\boldsymbol{\theta}_{\backslash m})]$. The inner expectation is often easy to compute (*e.g.,* with our CP/Tucker likelihood). When f_n is factorized individually over each element of $\boldsymbol{\theta}$, this can always be efficiently and accurately calculated by quadrature. However, the outer expectation is still difficult to obtain because $\widetilde{p}(\boldsymbol{\theta}_{\backslash m})$ is intractable. The second key idea of CEP is that since the moment matching is also between $q(\boldsymbol{\theta}_{\backslash m})$ and $\widetilde{p}(\boldsymbol{\theta}_{\backslash m})$, we can use the current marginal posterior to approximate the marginal titled distribution and then compute the outer expectation,

$$\mathbb{E}_{\widetilde{p}(\boldsymbol{\theta}_{\backslash m})}[\mathbf{g}(\boldsymbol{\theta}_{\backslash m})] \approx \mathbb{E}_{q(\boldsymbol{\theta}_{\backslash m})}[\mathbf{g}(\boldsymbol{\theta}_{\backslash m})]. \tag{17}$$

If it is still analytically intractable, we can use the delta method [Oehlert, 1992] to approximate the expectation. That is, we use a Taylor expansion of $\mathbf{g}(\cdot)$ at the mean of $\boldsymbol{\theta}_{\backslash m}$. Take the first-order expansion as an example,

$$\mathbf{g}(\boldsymbol{\theta}_{\backslash m}) \approx \mathbf{g}\left(\mathbb{E}_{q(\boldsymbol{\theta}_{\backslash m})}[\boldsymbol{\theta}_{\backslash m}]\right) + \mathbf{J}\left(\boldsymbol{\theta}_{\backslash m} - \mathbb{E}_{q(\boldsymbol{\theta}_{\backslash m})}[\boldsymbol{\theta}_{\backslash m}]\right)$$

where $\mathbf{J}$ is the Jacobian of $\mathbf{g}$ at $\mathbb{E}_{q(\boldsymbol{\theta}_{\backslash m})}[\boldsymbol{\theta}_{\backslash m}]$. Then we take the expectation on the Taylor approximation instead,

$$\mathbb{E}_{q(\boldsymbol{\theta}_{\backslash m})}\left[\mathbf{g}(\boldsymbol{\theta}_{\backslash m})\right] \approx \mathbf{g}\left(\mathbb{E}_{q(\boldsymbol{\theta}_{\backslash m})}[\boldsymbol{\theta}_{\backslash m}]\right). \tag{18}$$

The above computation are very conveniently to implement. Once we obtain the conditional moment $\mathbf{g}(\boldsymbol{\theta}_{\backslash m})$, we simply replace the $\boldsymbol{\theta}_{\backslash m}$ by its expectation under current posterior approximation q, *i.e.,* $\mathbb{E}_{q(\boldsymbol{\theta}_{\backslash m})}[\boldsymbol{\theta}_{\backslash m}]$, to obtain the matched moment $\mathbf{g}(\mathbb{E}_{q(\boldsymbol{\theta}_{\backslash m})}[\boldsymbol{\theta}_{\backslash m}])$, with which to construct q^* in Step 3 of EP (see (12)). The remaining steps are the same.

C.2 Running Posterior Update

Now we use the CEP framework to update the running posterior $p(\Theta_{n+1}, \tau|\mathcal{D}_{t_{n+1}})$ in equation (8) in main paper via the approximation (equation (9) in the main paper) . To simplify the notation, let us define $\mathbf{v}^m_{l_m} \triangleq \mathbf{u}^m_{\ell_m}(t_{n+1})$, and hence for each $(\boldsymbol{\ell}, y) \in \mathcal{B}_{n+1}$, we approximate

$$\mathcal{N}\left(y|\mathbf{1}^\top\left(\mathbf{v}^1_{\ell_1} \circ \ldots \circ \mathbf{v}^M_{\ell_M}\right), \tau^{-1}\right) \approx \prod_{m=1}^M \mathcal{N}(\mathbf{v}^m_{\ell_m}|\gamma^m_{\ell_m}, \boldsymbol{\Sigma}^m_{\ell_m})\text{Gam}(\tau|\alpha_{\boldsymbol{\ell}}, \omega_{\boldsymbol{\ell}}). \tag{19}$$

If we substitute (**??**) into (**??**), we can immediately obtain a Gaussian posterior approximation of each $\mathbf{v}^m_{\ell_m}$ and a Gamma posterior approximation of the noise inverse variance τ. Then dividing the current posterior approximation with the R.H.S. of (19), we can obtain the calibrated distribution,

$$\begin{aligned} q^{\backslash\boldsymbol{\ell}}(\mathbf{v}^m_{\ell_m}) &= \mathcal{N}(\mathbf{v}^m_{\ell_m}|\boldsymbol{\beta}^m_{\ell_m}, \boldsymbol{\Omega}^m_{\ell_m}), \\ q^{\backslash\boldsymbol{\ell}}(\tau) &= \text{Gam}(\alpha^{\backslash\boldsymbol{\ell}}, \omega^{\backslash\boldsymbol{\ell}}) \end{aligned} \tag{20}$$

where $1 \le m \le M$. Next, we construct a tilted distribution,

$$\widetilde{p}(\mathbf{v}^1_{\ell_1}, \ldots, \mathbf{v}^M_{\ell_M}, \tau) \propto q^{\backslash\boldsymbol{\ell}}(\tau) \cdot \prod_{m=1}^M q^{\backslash\boldsymbol{\ell}}(\mathbf{v}^m_{\ell_m}) \cdot \mathcal{N}\left(y|\mathbf{1}^\top\left(\mathbf{v}^1_{\ell_1} \circ \ldots \circ \mathbf{v}^M_{\ell_M}\right), \tau^{-1}\right). \tag{21}$$

To update each $\mathcal{N}(\mathbf{v}^m_{\ell_m}|\gamma^m_{\ell_m}, \boldsymbol{\Sigma}^m_{\ell_m})$ in (19), we first look into the conditional tilted distribution,

$$\widetilde{p}(\mathbf{v}^m_{\ell_m}|\mathcal{V}^{\backslash m}_{\boldsymbol{\ell}}, \tau) \propto \mathcal{N}(\mathbf{v}^m_{\boldsymbol{\ell}_m}|\boldsymbol{\beta}^m_{\ell_m}, \boldsymbol{\Omega}^m_{\ell_m}) \cdot \mathcal{N}\left(y|\left(\mathbf{v}^m_{\ell_m}\right)^\top \mathbf{v}^{\backslash m}_{\boldsymbol{\ell}}, \tau^{-1}\right) \tag{22}$$

where $\mathcal{V}_{\boldsymbol{\ell}}^{\backslash m}$ is $\{\mathbf{v}_{\ell_j}^{j}|1 \le j \le M, j \neq m\}$, and

$$\mathbf{v}_{\boldsymbol{\ell}}^{\backslash m} = \mathbf{v}_{\ell_1}^{1} \circ \ldots \circ \mathbf{v}_{\ell_{m-1}}^{m-1} \circ \mathbf{v}_{\ell_{m+1}}^{m+1} \circ \ldots \circ \mathbf{v}_{\ell_M}^{M}.$$

The conditional tilted distribution is obviously Gaussian, and the conditional moment is straightforward to obtain,

$$\mathbf{S}(\mathbf{v}_{\ell_m}^{m}|\mathcal{V}_{\boldsymbol{\ell}}^{\backslash m}, \tau) = \left[{\boldsymbol{\Omega}_{\ell_m}^{m}}^{-1} + \tau \mathbf{v}_{\boldsymbol{\ell}}^{\backslash m} \left(\mathbf{v}_{\boldsymbol{\ell}}^{\backslash m}\right)^{\top} \right]^{-1}, \tag{23}$$

$$\mathbb{E}[\mathbf{v}_{\ell_m}^{m}|\mathcal{V}_{\boldsymbol{\ell}}^{\backslash m}, \tau] = \mathbf{S}(\mathbf{v}_{\ell_m}^{m}|\mathcal{V}_{\boldsymbol{\ell}}^{\backslash m}, \tau) \cdot \left({\boldsymbol{\Omega}_{\boldsymbol{\ell}_m}^{m}}^{-1} \boldsymbol{\beta}_{\ell_m}^{m} + \tau y \mathbf{v}_{\boldsymbol{\ell}}^{\backslash m} \right), \tag{24}$$

where $\mathbf{S}$ denotes the conditional covariance. Next, according to (18), we simply replace τ, $\mathbf{v}_{\boldsymbol{\ell}}^{\backslash m}$, and $\mathbf{v}_{\boldsymbol{\ell}}^{\backslash m}\left(\mathbf{v}_{\boldsymbol{\ell}}^{\backslash m}\right)^{\top}$ by their expectation under the current posterior q in (23) and (24), to obtain the moments, *i.e.,* the mean and covariance matrix, with which we can construct q^* in Step 3 of the EP framework. The computation of $\mathbb{E}_q[\tau]$ is straightforward, and

$$\begin{aligned}
&\mathbb{E}_q[\mathbf{v}_{\boldsymbol{\ell}}^{\backslash m}] = \mathbb{E}_q[\mathbf{v}_{\ell_1}^{1}] \circ \ldots \circ \mathbb{E}_q[\mathbf{v}_{\ell_{m-1}}^{m-1}] \circ \mathbb{E}_q[\mathbf{v}_{\ell_{m+1}}^{m+1}] \circ \ldots \circ \mathbb{E}_q[\mathbf{v}_{\ell_M}^{M}], \\
&\mathbb{E}_q[\mathbf{v}_{\boldsymbol{\ell}}^{\backslash m}\left(\mathbf{v}_{\boldsymbol{\ell}}^{\backslash m}\right)^{\top}] = \mathbb{E}_q[\mathbf{v}_{\ell_1}^{1}\left(\mathbf{v}_{\ell_1}^{1}\right)^{\top}] \circ \ldots \circ \mathbb{E}_q[\mathbf{v}_{\ell_{m-1}}^{m-1}\left(\mathbf{v}_{\ell_{m-1}}^{m-1}\right)^{\top}] \\
&\circ \mathbb{E}_q[\mathbf{v}_{\ell_{m+1}}^{m+1}\left(\mathbf{v}_{\ell_{m+1}}^{m+1}\right)^{\top}] \circ \ldots \circ \mathbb{E}_q[\mathbf{v}_{\ell_M}^{M}\left(\mathbf{v}_{\ell_M}^{M}\right)^{\top}].
\end{aligned}$$

Similarly, to update $\text{Gam}(\alpha_{\boldsymbol{\ell}}, \omega_{\boldsymbol{\ell}})$ in (19), we first observe that the conditional titled distribution is also a Gamma distribution,

$$\widetilde{p}(\tau|\mathcal{V}_{\boldsymbol{\ell}}) \propto \text{Gam}(\tau|\widetilde{\alpha}, \widetilde{\omega}) \propto \text{Gam}(\tau|\alpha^{\backslash \boldsymbol{\ell}}, \omega^{\backslash \boldsymbol{\ell}})\mathcal{N}(y|\mathbf{1}^{\top}\mathbf{v}_{\boldsymbol{\ell}}, \tau^{-1}) \tag{25}$$

where $\mathbf{v}_{\boldsymbol{\ell}} = \mathbf{v}_{\ell_1}^{1} \circ \ldots \circ \mathbf{v}_{\ell_M}^{M}$, and

$$\begin{aligned}
\widetilde{\alpha} &= \alpha^{\backslash \boldsymbol{\ell}} + \frac{1}{2}, \\
\widetilde{\omega} &= \omega^{\backslash \boldsymbol{\ell}} + \frac{1}{2}y^2 + \frac{1}{2}\mathbf{1}^{\top}\mathbf{v}_{\boldsymbol{\ell}}\mathbf{v}_{\boldsymbol{\ell}}^{\top}\mathbf{1} - y\mathbf{1}^{\top}\mathbf{v}.
\end{aligned} \tag{26}$$

Since the conditional moments (the expectation of τ and $\log \tau$) are functions of α and ω, when using the delta method to approximate the expected conditional moment, it is equivalent to approximating the expectation of $\widetilde{\alpha}$ and $\widetilde{\omega}$ first, and then use the expected $\widetilde{\alpha}$ and $\widetilde{\omega}$ to recover the moments. As a result, we can simply replace $\mathbf{v}_{\boldsymbol{\ell}}$ and $\mathbf{v}_{\boldsymbol{\ell}}\mathbf{v}_{\boldsymbol{\ell}}^{\top}$ in (26) by their expectation under the current posterior, and we obtain the approximation of $\mathbb{E}_q[\widetilde{\alpha}]$ and $\mathbb{E}_q[\widetilde{\omega}]$. With these approximated expectation, we then construct $q^*(\tau) = \text{Gam}(\tau|\mathbb{E}_q[\alpha], \mathbb{E}_q[\omega])$ at Step 3 in EP. The remaining steps are straightforward. The running posterior update with the Tucker form likelihood follows a similar way.

D More Results on Simulation Study

D.1 Accuracy of Trajectory Recovery

We provide the quantitative result in recovering the factor trajectories. Note that there is only one competing method, NONFAT, which can also estimate factor trajectories. We therefore ran our method and NONFAT on the synthetic dataset. We then randomly sampled 500 time points in the domain and evaluate the RMSE of the learned factor trajectories for each method. As shown in Table 1, the RMSE of NONFAT on recovering $u_1^1(t)$ and $u_1^2(t)$ is close to SFTL, showing NONFAT achieved the same (or very close) quality in recovering these two trajectories. However, on $u_2^1(t)$ and $u_2^2(t)$, the RMSE of NONFAT is much larger, showing that NONFAT have failed to capture the other two trajectories. By contrast, SFTL consistently well recovered them.

D.2 Sensitive Analysis on Kernel Parameters

To examine the sensitivity to the kernel parameters, we used the synthetic dataset, and randomly sampled 100 entries and new timestamps for evaluation. We then examined the length-scale ρ and

	$u_1^1(t)$	$u_2^1(t)$	$u_1^2(t)$	$u_2^2(t)$
SFTL	0.073	0.082	0.103	0.054
NONFAT	0.085	0.442	0.096	0.443

Table 1: RMSE in recovering trajectories on the simulation data.

	ρ	0.1	0.3	0.5	0.7	0.9
Matérn-1/2	SFTL-CP	0.091	0.064	0.059	0.056	0.057
	SFTL-Tucker	0.060	0.055	0.056	0.056	0.057
Matérn-3/2	SFTL-CP	0.062	0.061	0.074	0.093	0.112
	SFTL-Tucker	0.061	0.059	0.078	0.101	0.129

(a) Prediction RMSE with $a = 0.3$ and varying ρ.

	a	0.1	0.3	0.5	0.7	0.9
Matérn-1/2	SFTL-CP	0.056	0.064	0.057	0.059	0.063
	SFTL-Tucker	0.065	0.055	0.054	0.055	0.055
Matérn-3/2	SFTL-CP	0.072	0.061	0.063	0.060	0.059
	SFTL-Tucker	0.098	0.059	0.064	0.062	0.061

(b) Prediction RMSE with $\rho = 0.3$ and varying a.

Table 2: Sensitive analysis of amplitude a and length-scale ρ on synthetic data.

amplitude a, for two commonly-used Matérn kernels: Matérn-1/2 and Matérn-3/2. The study was performed on SFTL based on both the CP and Tucker forms. The results are reported in Table 2. Overall, the predictive performance of SFTL is less sensitive to the amplitude parameter a than to the length-scale parameter ρ. But when we use Matérn-1/2, the performance of both SFTL-CP and SFTL-Tucker is quite stable to the length-scale parameter ρ. When we use Matérn-3/2, the choice of the length-scale is critical.

E Real-World Dataset Information and Competing Methods

We tested all the methods in the following four real-world datasets.

- *FitRecord*[1], workout logs of EndoMondo users' health status in outdoor exercises. We extracted a three-mode tensor among 500 users, 20 sports types, and 50 altitudes. The entry values are heart rates. There are 50K observed entry values along with the timestamps.
- *ServerRoom*[2], temperature logs of Poznan Supercomputing and Networking Center. We extracted a three-mode tensor between 3 air conditioning modes (24°, 27° and 30°), 3 power usage levels (50%, 75%, 100%) and 34 locations. We collected 10K entry values and their timestamps.
- *BeijingAir-2*[3], air pollution measurement in Beijing from year 2014 to 2017. We extracted a two-mode tensor (monitoring site, pollutant), of size 12×6, and collected 20K observed entry values (concentration) and their timestamps.
- *BeijingAir-3*, extracted from the same data source as *BeijingAir-2*, a three-mode tensor among 12 monitoring sites, 12 wind speeds and 6 wind directions. The entry value is the PM2.5 concentration. There are 15K observed entry values at different timestamps.

We first compared with the following state-of-the-art streaming tensor decomposition methods based on the CP or Tucker model. (1) POST [Du et al., 2018], probabilistic streaming CP decomposition via mean-field streaming variational Bayes [Broderick et al., 2013] (2) BASS-Tucker [Fang et al., 2021] Bayesian streaming Tucker decomposition, which online estimates a sparse tensor-core via a spike-and-slab prior to enhance the interpretability. We also implemented (3) ADF-CP, streaming CP

[1] https://sites.google.com/eng.ucsd.edu/fitrec-project/home
[2] https://zenodo.org/record/3610078#%23.Y8SYt3bMJGi
[3] https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data

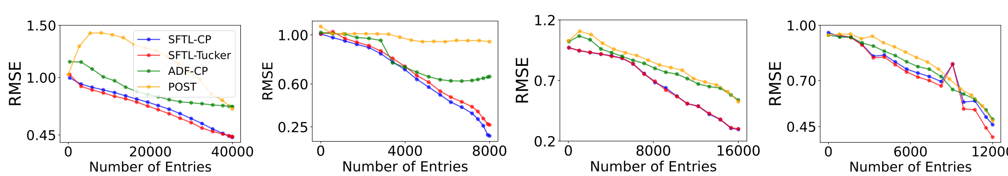
decomposition by combining the assumed density filtering and conditional moment matching [Wang and Zhe, 2019].

Next, we tested the state-of-the-art static decomposition algorithms, which have to go through the data many times. (4) P-Tucker [Oh et al., 2018], an efficient Tucker decomposition algorithm that performs parallel row-wise updates. (5) CP-ALS and (6) Tucker-ALS [Bader and Kolda, 2008], CP/Tucker decomposition via alternating least square (ALS) updates. The methods (1-6) are not specifically designed for temporal decomposition and cannot utilize the timestamps of the observed entries. In order to incorporate the time information for a fair comparison, we augment the tensor with a time mode, and convert the ordered, unique timestamps into increasing time steps.

We then compare with the most recent continuous-time temporal decomposition methods. Note that none of these methods can handle data streams. They have to iteratively access the data to update the model parameters and factor estimates. (7) CT-CP [Zhang et al., 2021], continuous-time CP decomposition, which uses polynomial splines to model a time-varying coefficient $\boldsymbol{\lambda}$ for each latent factor, (8) CT-GP, continuous-time GP decomposition, which extends [Zhe et al., 2016] to use GPs to learn the tensor entry value as a function of the latent factors and time $y_{\boldsymbol{\ell}}(t) = g(\mathbf{u}^1_{\ell_1}, \ldots, \mathbf{u}^K_{\ell_K}, t) \sim \mathcal{GP}(0, \kappa(\cdot,\cdot))$, (9) BCTT [Fang et al., 2022], Bayesian continuous-time Tucker decomposition, which estimates the tensor-core as a time-varying function, (10) THIS-ODE [Li et al., 2022], which uses a neural ODE [Chen et al., 2018] to model the entry value as a function of the latent factors and time, $\frac{\mathrm{d}y_{\boldsymbol{\ell}}(t)}{\mathrm{d}t} = \mathrm{NN}(\mathbf{u}^1_{\ell_1}, \ldots, \mathbf{u}^K_{\ell_K}, t)$ where NN is short for neural networks. (11) NONFAT [Wang and Zhe, 2022], nonparametric factor trajectory learning, the only existing work that also estimates factor trajectories for temporal tensor decomposition. It uses a bi-level GP to estimate the trajectories in the frequency domain and applies inverse Fourier transform to return to the time domain.

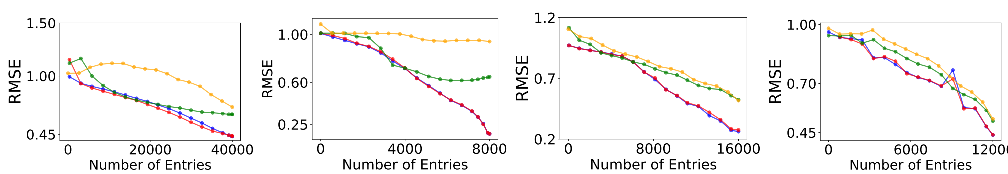
F More Results about Prediction Accuracy

We report for $R = 2$, $R = 3$ and $R = 7$, the final prediction error (after the data has been processed) of all the methods in Table 3, Table 4, and Table 5, respectively. We report for $R = 2$, $R = 3$ and $R = 7$, the online predictive performance of the streaming decomposition approaches in Fig. 1, Fig. 2, and Fig. 3, respectively.



(a) *FitRecord* (b) *ServerRoom* (c) *BeijingAir-2* (d) *BeijingAir-3*

Figure 1: Online prediction error with the number of processed entries ($R = 2$)



(a) *FitRecord* (b) *ServerRoom* (c) *BeijingAir-2* (d) *BeijingAir-3*

Figure 2: Online prediction error with the number of processed entries ($R = 3$)

G Running time

As compared with static (non-streaming) methods, such as BCTT, our method is faster and more efficient. That is because whenever new data comes in, the static methods have to retrain the model from scratch and iteratively access the whole data accumulated so far, while our method only performs incremental updates and never needs to revisit the past data. To demonstrate this point, we compare

	RMSE	*FitRecord*	*ServerRoom*	*BeijingAir-2*	*BeijingAir-3*
Static	PTucker	0.606 ± 0.015	0.757 ± 0.36	0.509 ± 0.01	0.442 ± 0.142
	Tucker-ALS	0.914 ± 0.01	0.991 ± 0.016	0.586 ± 0.016	0.896 ± 0.032
	CP-ALS	0.926 ± 0.013	0.997 ± 0.016	0.647 ± 0.041	0.918 ± 0.031
	CT-CP	0.675 ± 0.009	0.412 ± 0.024	0.642 ± 0.007	0.832 ± 0.035
	CT-GP	0.611 ± 0.009	0.218 ± 0.021	0.723 ± 0.01	0.88 ± 0.026
	BCTT	0.604 ± 0.019	0.715 ± 0.352	0.504 ± 0.01	0.799 ± 0.027
	NONFAT	0.543 ± 0.002	$\mathbf{0.132 \pm 0.002}$	0.425 ± 0.002	0.878 ± 0.014
	THIS-ODE	0.544 ± 0.005	0.142 ± 0.004	0.553 ± 0.015	0.876 ± 0.027
Stream	POST	0.705 ± 0.013	0.767 ± 0.155	0.539 ± 0.01	0.695 ± 0.135
	ADF-CP	0.669 ± 0.033	0.764 ± 0.114	0.583 ± 0.07	0.54 ± 0.045
	BASS-Tucker	1 ± 0.016	1 ± 0.016	1.043 ± 0.05	0.982 ± 0.058
	SFTL-CP	$\mathbf{0.437 \pm 0.014}$	0.18 ± 0.019	$\mathbf{0.323 \pm 0.019}$	0.462 ± 0.009
	SFTL-Tucker	0.446 ± 0.024	0.276 ± 0.031	0.344 ± 0.031	$\mathbf{0.417 \pm 0.035}$
	MAE				
Static	PTucker	0.416 ± 0.005	0.388 ± 0.152	0.336 ± 0.004	0.271 ± 0.053
	Tucker-ALS	0.676 ± 0.008	0.744 ± 0.01	0.408 ± 0.008	0.669 ± 0.02
	CP-ALS	0.686 ± 0.011	0.748 ± 0.009	0.454 ± 0.057	0.691 ± 0.016
	CT-CP	0.466 ± 0.005	0.295 ± 0.029	0.49 ± 0.006	0.642 ± 0.02
	CT-GP	0.424 ± 0.006	0.155 ± 0.012	0.517 ± 0.01	0.626 ± 0.01
	BCTT	0.419 ± 0.015	0.534 ± 0.263	0.343 ± 0.003	0.579 ± 0.018
	NONFAT	0.373 ± 0.001	$\mathbf{0.083 \pm 0.001}$	0.282 ± 0.002	0.622 ± 0.006
	THIS-ODE	0.377 ± 0.003	0.097 ± 0.003	0.355 ± 0.008	0.606 ± 0.015
Stream	POST	0.485 ± 0.008	0.564 ± 0.091	0.368 ± 0.008	0.517 ± 0.123
	ADF-CP	0.462 ± 0.022	0.574 ± 0.073	0.401 ± 0.029	0.415 ± 0.038
	BASS	0.777 ± 0.039	0.749 ± 0.01	0.871 ± 0.125	0.727 ± 0.029
	SFTL-CP	$\mathbf{0.248 \pm 0.005}$	0.126 ± 0.007	$\mathbf{0.199 \pm 0.005}$	0.311 ± 0.004
	SFTL-Tucker	0.25 ± 0.01	0.203 ± 0.032	0.218 ± 0.02	$\mathbf{0.261 \pm 0.023}$

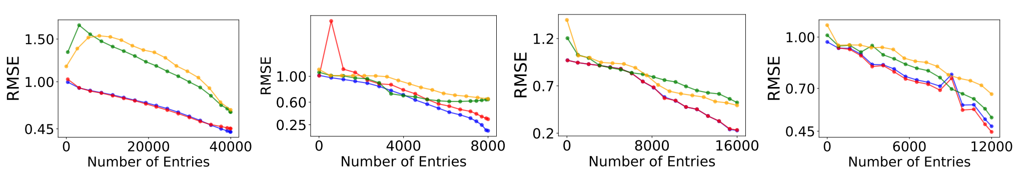
Table 3: Final prediction error with $R = 2$. The results were averaged from five runs.

	RMSE	*FitRecord*	*ServerRoom*	*BeijingAir-2*	*BeijingAir-3*
Static	PTucker	0.603 ± 0.045	0.677 ± 0.129	0.464 ± 0.012	0.421 ± 0.074
	Tucker-ALS	0.885 ± 0.007	0.989 ± 0.014	0.559 ± 0.017	0.863 ± 0.032
	CP-ALS	0.907 ± 0.015	0.993 ± 0.016	0.594 ± 0.031	0.901 ± 0.03
	CT-CP	0.666 ± 0.008	0.5 ± 0.2	0.641 ± 0.006	0.819 ± 0.019
	CT-GP	0.606 ± 0.008	0.217 ± 0.025	0.749 ± 0.014	0.895 ± 0.054
	BCTT	0.576 ± 0.015	0.358 ± 0.082	0.454 ± 0.011	0.829 ± 0.028
	NONFAT	0.517 ± 0.002	$\mathbf{0.129 \pm 0.002}$	0.408 ± 0.005	0.877 ± 0.014
	THIS-ODE	0.528 ± 0.005	0.132 ± 0.002	0.544 ± 0.014	0.878 ± 0.026
Stream	POST	0.706 ± 0.034	0.741 ± 0.161	0.518 ± 0.016	0.622 ± 0.123
	ADF-CP	0.641 ± 0.009	0.652 ± 0.012	0.542 ± 0.012	0.518 ± 0.003
	BASS-Tucker	1.008 ± 0.017	1 ± 0.016	1.035 ± 0.038	0.99 ± 0.034
	SFTL-CP	0.434 ± 0.014	0.178 ± 0.006	$\mathbf{0.288 \pm 0.017}$	0.454 ± 0.011
	SFTL-Tucker	$\mathbf{0.418 \pm 0.01}$	0.289 ± 0.096	0.314 ± 0.049	$\mathbf{0.41 \pm 0.013}$
	MAE				
Static	PTucker	0.392 ± 0.009	0.323 ± 0.053	0.307 ± 0.005	0.197 ± 0.029
	Tucker-ALS	0.648 ± 0.012	0.743 ± 0.008	0.39 ± 0.008	0.651 ± 0.018
	CP-ALS	0.666 ± 0.013	0.746 ± 0.01	0.415 ± 0.022	0.676 ± 0.021
	CT-CP	0.462 ± 0.005	0.348 ± 0.141	0.489 ± 0.006	0.632 ± 0.015
	CT-GP	0.419 ± 0.005	0.158 ± 0.022	0.544 ± 0.012	0.627 ± 0.015
	BCTT	0.392 ± 0.004	0.267 ± 0.067	0.299 ± 0.006	0.607 ± 0.027
	NONFAT	0.355 ± 0.001	$\mathbf{0.078 \pm 0.001}$	0.265 ± 0.003	0.622 ± 0.006
	THIS-ODE	0.363 ± 0.004	0.083 ± 0.002	0.348 ± 0.006	0.603 ± 0.009
Stream	POST	0.482 ± 0.022	0.54 ± 0.102	0.351 ± 0.009	0.442 ± 0.109
	ADF-CP	0.445 ± 0.006	0.5 ± 0.009	0.381 ± 0.006	0.393 ± 0.009
	BASS	0.822 ± 0.024	0.749 ± 0.009	0.919 ± 0.041	0.73 ± 0.018
	SFTL-CP	0.246 ± 0.005	0.121 ± 0.003	$\mathbf{0.176 \pm 0.006}$	0.305 ± 0.006
	SFTL-Tucker	$\mathbf{0.24 \pm 0.002}$	0.18 ± 0.042	0.196 ± 0.03	$\mathbf{0.263 \pm 0.011}$

Table 4: Final prediction error with $R = 3$. The results were averaged from five runs.

	RMSE	*FitRecord*	*ServerRoom*	*BeijingAir-2*	*BeijingAir-3*
Static	PTucker	0.603 ± 0.045	0.677 ± 0.129	0.464 ± 0.012	0.421 ± 0.074
	Tucker-ALS	0.826 ± 0.003	0.983 ± 0.016	0.586 ± 0.018	0.825 ± 0.026
	CP-ALS	0.878 ± 0.012	0.994 ± 0.013	0.897 ± 0.215	0.863 ± 0.024
	CT-CP	0.663 ± 0.008	0.384 ± 0.008	0.64 ± 0.007	0.818 ± 0.019
	CT-GP	0.603 ± 0.006	0.381 ± 0.303	0.766 ± 0.016	0.904 ± 0.046
	BCTT	0.498 ± 0.011	0.194 ± 0.017	0.368 ± 0.01	0.813 ± 0.028
	NONFAT	0.497 ± 0.003	$\mathbf{0.128 \pm 0.002}$	0.394 ± 0.004	0.88 ± 0.013
	THIS-ODE	0.138 ± 0.003	0.554 ± 0.016	0.878 ± 0.027	
Stream	POST	0.675 ± 0.012	0.707 ± 0.14	0.519 ± 0.017	0.738 ± 0.068
	ADF-CP	0.652 ± 0.01	0.646 ± 0.008	0.548 ± 0.012	0.552 ± 0.026
	BASS-Tucker	0.604 ± 0.043	0.493 ± 0.071	0.391 ± 0.005	0.634 ± 0.083
	SFTL-CP	$\mathbf{0.424 \pm 0.006}$	0.166 ± 0.013	0.256 ± 0.013	0.481 ± 0.006
	SFTL-Tucker	0.448 ± 0.009	0.406 ± 0.052	$\mathbf{0.249 \pm 0.017}$	$\mathbf{0.432 \pm 0.019}$
	MAE				
Static	PTucker	0.353 ± 0.005	0.305 ± 0.042	0.248 ± 0.004	0.32 ± 0.038
	Tucker-ALS	0.6 ± 0.002	0.737 ± 0.009	0.392 ± 0.011	0.619 ± 0.015
	CP-ALS	0.64 ± 0.009	0.745 ± 0.008	0.593 ± 0.121	0.637 ± 0.015
	CT-CP	0.459 ± 0.005	0.27 ± 0.003	0.488 ± 0.005	0.626 ± 0.012
	CT-GP	0.412 ± 0.004	0.282 ± 0.23	0.557 ± 0.009	0.628 ± 0.01
	BCTT	0.342 ± 0.005	0.157 ± 0.015	0.234 ± 0.005	0.581 ± 0.022
	NONFAT	0.335 ± 0.002	$\mathbf{0.077 \pm 0.002}$	0.256 ± 0.003	0.627 ± 0.005
	THIS-ODE	0.362 ± 0.002	0.089 ± 0.002	0.357 ± 0.007	0.603 ± 0.013
Stream	POST	0.461 ± 0.008	0.518 ± 0.087	0.357 ± 0.011	0.558 ± 0.058
	ADF-CP	0.451 ± 0.006	0.489 ± 0.009	0.384 ± 0.014	0.411 ± 0.025
	BASS	0.745 ± 0.026	0.749 ± 0.01	0.903 ± 0.044	0.721 ± 0.038
	SFTL-CP	$\mathbf{0.243 \pm 0.003}$	0.111 ± 0.008	0.159 ± 0.004	0.323 ± 0.003
	SFTL-Tucker	0.253 ± 0.004	0.273 ± 0.033	$\mathbf{0.144 \pm 0.008}$	$\mathbf{0.273 \pm 0.016}$

Table 5: Final prediction error with $R = 7$. The results were averaged from five runs.



(a) *FitRecord* (b) *ServerRoom* (c) *BeijingAir-2* (d) *BeijingAir-3*

Figure 3: Online prediction error with the number of processed entries ($R = 7$)

the training time of our method with BCTT on *BeijingAir2* dataset. All the methods ran on a Linux workstation. From Table 6, we can see a large speed-up of our method with both the CP and Tucker form. The higher the rank (R), the more significant the speed-up.

	$R = 2$	$R = 3$	$R = 5$	$R = 7$
SFTL-CP	27.1	27.2	28.5	29.1
SFTL-Tucker	32.3	35.6	43.2	59.3
BCTT	49.5	56.1	72.1	136.7

Table 6: Running time in seconds on *BeijingAir2* dataset.

H Limitation and Discussion

The state-space prior used our method arises from the LTI-SDE (7), an equivalent representation of the GP prior over time functions using a type of Matérn kernels. While elegant and useful, building equivalent SDEs to a specific GP prior might restrict the expressivity of our model. To overcome this limitation, we plan to construct an SDE prior directly, *e.g.,* a linear SDE to model how the factor trajectory varies along the time. Then we consider converting the SDE into a state-space prior. In doing so, we can further improve the flexibility of our model to capture more complex temporal evolution, *e.g.,* non-stationary and highly fluctuating.

References

Brett W Bader and Tamara G Kolda. Efficient matlab computations with sparse and factored tensors. SIAM Journal on Scientific Computing, 30(1):205–231, 2008.

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michael I Jordan. Streaming variational bayes. Advances in neural information processing systems, 26, 2013.

Chris Chatfield. The analysis of time series: an introduction. Chapman and hall/CRC, 2003.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. Advances in neural information processing systems, 31, 2018.

Yishuai Du, Yimin Zheng, Kuang-chih Lee, and Shandian Zhe. Probabilistic streaming tensor decomposition. In 2018 IEEE International Conference on Data Mining (ICDM), pages 99–108. IEEE, 2018.

Shikai Fang, Robert M Kirby, and Shandian Zhe. Bayesian streaming sparse tucker decomposition. In Uncertainty in Artificial Intelligence, pages 558–567. PMLR, 2021.

Shikai Fang, Akil Narayan, Robert Kirby, and Shandian Zhe. Bayesian continuous-time tucker decomposition. In International Conference on Machine Learning, pages 6235–6245. PMLR, 2022.

Peter Lancaster and Leiba Rodman. Algebraic riccati equations. Clarendon press, 1995.

Shibo Li, Robert Kirby, and Shandian Zhe. Decomposing temporal high-order interactions via latent odes. In International Conference on Machine Learning, pages 12797–12812. PMLR, 2022.

Thomas P Minka. Expectation propagation for approximate bayesian inference. In Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence, pages 362–369, 2001.

Gary W Oehlert. A note on the delta method. The American Statistician, 46(1):27–29, 1992.

Sejoon Oh, Namyong Park, Sael Lee, and Uksong Kang. Scalable tucker factorization for sparse tensors-algorithms and discoveries. In 2018 IEEE 34th International Conference on Data Engineering (ICDE), pages 1120–1131. IEEE, 2018.

Simo Särkkä. Bayesian filtering and smoothing. Number 3. Cambridge University Press, 2013.

Zheng Wang and Shandian Zhe. Conditional expectation propagation. In UAI, page 6, 2019.

Zheng Wang and Shandian Zhe. Nonparametric factor trajectory learning for dynamic tensor decomposition. In International Conference on Machine Learning, pages 23459–23469. PMLR, 2022.

Yanqing Zhang, Xuan Bi, Niansheng Tang, and Annie Qu. Dynamic tensor recommender systems. Journal of Machine Learning Research, 22(65):1–35, 2021.

Shandian Zhe, Yuan Qi, Youngja Park, Zenglin Xu, Ian Molloy, and Suresh Chari. Dintucker: Scaling up gaussian process models on large multidimensional arrays. In Thirtieth AAAI conference on artificial intelligence, 2016.