Appendices

Appendix A provides derivations supporting Section 3 in the main paper. In Appendix B, we explain our experimental setup, including dataset preparation and model implementation, in more detail. Finally, Appendix C provides additional results supporting our claims regarding the scalability of our method, together with additional results from the experiments presented in Section 4.

A ST-DGMRF derivations

In this section we provide detailed derivations of the ST-DGMRF joint distribution, for both first-order transition models (Section A.1) and higher-order transition models (Section A.2).

A.1 Joint distribution

The LDS (see Section 2.2 and 3.1 in the main paper) defines a joint distribution over system states $\mathbf{x}_{0:K}$ that factorizes as

$$p(\mathbf{x}_{0:K}) = \mathcal{N}(\mathbf{x}_0 \mid \boldsymbol{\mu}_0, \mathbf{Q}_0^{-1}) \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_k \mid \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{c}_k, \mathbf{Q}_k^{-1}), \tag{1}$$

with $\mathbf{x}_k, \boldsymbol{\mu}_0, \mathbf{c}_k \in \mathbb{R}^N$ and $\mathbf{F}_k, \mathbf{Q}_k \in \mathbb{R}^{N \times N}$. As a product of Gaussian distributions, $p(\mathbf{x}_{0:K})$ can be written as a joint Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Omega}^{-1})$ with mean $\boldsymbol{\mu} \in \mathbb{R}^{(K+1)N}$ and precision (inverse covariance) matrix $\boldsymbol{\Omega} \in \mathbb{R}^{(K+1)N \times (K+1)N}$. Here, we derive expressions for $\boldsymbol{\mu}$ and $\boldsymbol{\Omega}$ in terms of $\boldsymbol{\mu}_0, \mathbf{c}_k, \mathbf{F}_k, \mathbf{Q}_k$.

First, note that Eq. (1) can be written as a set of linear equations

$$\begin{aligned}
\mathbf{x}_0 &= \boldsymbol{\mu}_0 + \boldsymbol{\epsilon}_0 & \boldsymbol{\epsilon}_0 &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_0^{-1}) \\
\mathbf{x}_1 &= \mathbf{F}_1 \mathbf{x}_0 + \mathbf{c}_1 + \boldsymbol{\epsilon}_1 & \boldsymbol{\epsilon}_1 &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_1^{-1}) \\
\mathbf{x}_2 &= \mathbf{F}_2 \mathbf{x}_1 + \mathbf{c}_2 + \boldsymbol{\epsilon}_2 & \boldsymbol{\epsilon}_2 &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_2^{-1}) \\
&\dots \\
\mathbf{x}_K &= \mathbf{F}_K \mathbf{x}_{K-1} + \mathbf{c}_K + \boldsymbol{\epsilon}_K & \boldsymbol{\epsilon}_K &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_K^{-1}).
\end{aligned}$$

Moving all $\mathbf{x}_k$-terms to the left-hand side, we can rewrite this as a matrix-vector multiplication

$$\underbrace{\begin{bmatrix} \mathbf{I} & & & & \\ -\mathbf{F}_1 & \mathbf{I} & & & \\ & -\mathbf{F}_2 & \mathbf{I} & & \\ & & \dots & \dots & \\ & & & -\mathbf{F}_K & \mathbf{I} \end{bmatrix}}_{=\mathbf{F}} \cdot \underbrace{\begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_K \end{bmatrix}}_{=\mathbf{x}} = \underbrace{\begin{bmatrix} \boldsymbol{\mu}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_K \end{bmatrix}}_{=\mathbf{c}} + \underbrace{\begin{bmatrix} \boldsymbol{\epsilon}_0 \\ \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \\ \vdots \\ \boldsymbol{\epsilon}_K \end{bmatrix}}_{=\boldsymbol{\epsilon}}, \tag{2}$$

with block-matrix $\mathbf{F} \in \mathbb{R}^{(K+1)N \times (K+1)N}$ and vectorized $\mathbf{x} = \text{vec}(\mathbf{x}_0, \dots, \mathbf{x}_K) \in \mathbb{R}^{(K+1)N}$, $\mathbf{c} = \text{vec}(\boldsymbol{\mu}_0, \mathbf{c}_1, \dots, \mathbf{c}_K) \in \mathbb{R}^{(K+1)N}$ and $\boldsymbol{\epsilon} = \text{vec}(\boldsymbol{\epsilon}_0, \dots, \boldsymbol{\epsilon}_K) \in \mathbb{R}^{(K+1)N}$. Empty positions in $\mathbf{F}$ represent zero-blocks.

Now, we can express $\mathbf{x}$ as an affine transformation of $\boldsymbol{\epsilon}$

$$\mathbf{x} = \mathbf{F}^{-1}\mathbf{c} + \mathbf{F}^{-1}\boldsymbol{\epsilon}, \tag{3}$$

where $\mathbf{F}^{-1}$ exists because $\det(\mathbf{F}) = 1$. Since $\boldsymbol{\epsilon}$ is distributed as $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$ with $\mathbf{Q} = \text{diag}(\mathbf{Q}_0, \mathbf{Q}_1, \dots, \mathbf{Q}_K)$, and $\mathbf{c}$ is deterministic, we can use the affine property of Gaussian distributions to obtain the joint distribution

$$\mathbf{x} \sim \mathcal{N}(\mathbf{F}^{-1}\mathbf{c}, \mathbf{F}^{-1}\mathbf{Q}^{-1}\mathbf{F}^{-T}). \tag{4}$$

Thus, the joint precision matrix $\boldsymbol{\Omega}$ factorizes as

$$\boldsymbol{\Omega} = (\mathbf{F}^{-1}\mathbf{Q}^{-1}\mathbf{F}^{-T})^{-1} = \mathbf{F}^T\mathbf{Q}\mathbf{F} \tag{5}$$

and has a block-tridiagonal structure

$$\mathbf{\Omega} = \begin{bmatrix} \mathbf{Q}_0 + \mathbf{F}_1^T\mathbf{Q}_1\mathbf{F}_1 & -\mathbf{F}_1^T\mathbf{Q}_1 & & & \\ -\mathbf{Q}_1\mathbf{F}_1 & \mathbf{Q}_1 + \mathbf{F}_2^T\mathbf{Q}_2\mathbf{F}_2 & -\mathbf{F}_2^T\mathbf{Q}_2 & & \\ & \dots & \dots & \dots & \\ & & -\mathbf{Q}_{K-1}\mathbf{F}_{K-1} & \mathbf{Q}_{K-1} + \mathbf{F}_K^T\mathbf{Q}_K\mathbf{F}_T & -\mathbf{F}_K^T\mathbf{Q}_K \\ & & & -\mathbf{Q}_K\mathbf{F}_K & \mathbf{Q}_K \end{bmatrix}. \tag{6}$$

Note that for matrix-vector multiplications of the form $\mathbf{\Omega x}$, the sparse structure of $\mathbf{F}$ and $\mathbf{Q}$ can be leveraged by performing three consecutive matrix-vector multiplications, instead of first forming the full precision matrix and then computing the matrix vector product. This reduces both computations and memory requirements.

To compute the joint mean $\boldsymbol{\mu} = \mathbf{F}^{-1}\mathbf{c}$ without expensive matrix inversion, the components $\boldsymbol{\mu}_k$ have to be computed iteratively as

$$\boldsymbol{\mu}_k = \mathbf{F}_k\boldsymbol{\mu}_{k-1} + \mathbf{c}_k. \tag{7}$$

In contrast, the information vector $\boldsymbol{\eta} = \mathbf{\Omega}\boldsymbol{\mu}$ can be expressed compactly as

$$\boldsymbol{\eta} = \mathbf{F}^T\mathbf{Q}\mathbf{F}\mathbf{F}^{-1}\mathbf{c} = \mathbf{F}^T\mathbf{Q}\mathbf{c}, \tag{8}$$

which can be computed efficiently using sparse and parallel matrix-vector multiplications on a GPU. We make use of this property in the DGMRF formulation and in the conjugate gradient method.

A.2 Extension to higher-order Markov processes

We can easily adjust the joint distribution to accommodate higher-order processes with dependencies on multiple past time steps.

For a p-th order Markov process, the dynamics are defined by equations

$$\begin{aligned} \mathbf{x}_0 &= \boldsymbol{\mu}_0 + \boldsymbol{\epsilon}_0 && \boldsymbol{\epsilon}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_0^{-1}) \\ \mathbf{x}_1 &= \mathbf{F}_{1,1}\mathbf{x}_0 + \mathbf{c}_1 + \boldsymbol{\epsilon}_1 && \boldsymbol{\epsilon}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_1^{-1}) \\ &\dots \\ \mathbf{x}_k &= \mathbf{F}_{k,1}\mathbf{x}_{k-1} + \mathbf{F}_{k,2}\mathbf{x}_{k-2} + \dots + \mathbf{F}_{k,p}\mathbf{x}_{k-p} + \mathbf{c}_k + \boldsymbol{\epsilon}_k && \boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k^{-1}) \\ &\dots \\ \mathbf{x}_K &= \mathbf{F}_{K,1}\mathbf{x}_{K-1} + \mathbf{F}_{K,2}\mathbf{x}_{K-2} + \dots + \mathbf{F}_{K,p}\mathbf{x}_{K-p} + \mathbf{c}_K + \boldsymbol{\epsilon}_K && \boldsymbol{\epsilon}_K \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_K^{-1}). \end{aligned}$$

Following the same steps as before, this results in a linear system

$$\underbrace{\begin{bmatrix} \mathbf{I} & & & & & & \\ -\mathbf{F}_{1,1} & \mathbf{I} & & & & & \\ -\mathbf{F}_{2,2} & -\mathbf{F}_{2,1} & \mathbf{I} & & & & \\ \dots & \dots & \dots & \dots & & & \\ -\mathbf{F}_{p,p} & -\mathbf{F}_{p,p-1} & \dots & -\mathbf{F}_{p,1} & \mathbf{I} & & \\ & \dots & \dots & \dots & \dots & \dots & \\ & & -\mathbf{F}_{K,p} & \dots & -\mathbf{F}_{K,2} & -\mathbf{F}_{K,1} & \mathbf{I} \end{bmatrix}}_{=\mathbf{F}} \cdot \underbrace{\begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_K \end{bmatrix}}_{=\mathbf{x}} = \underbrace{\begin{bmatrix} \boldsymbol{\mu}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_K \end{bmatrix}}_{=\mathbf{c}} + \underbrace{\begin{bmatrix} \boldsymbol{\epsilon}_0 \\ \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \\ \vdots \\ \boldsymbol{\epsilon}_K \end{bmatrix}}_{=\boldsymbol{\epsilon}}. \tag{9}$$

This means that the matrix $\mathbf{F}$ is extended by adding the higher-order transition matrices $(\mathbf{F}_{\tau,\tau}, \dots, \mathbf{F}_{K,\tau})$ to the τ-th lower block diagonal of $\mathbf{F}$ for all $\tau = 1, \dots, p$. The expressions for $\mathbf{\Omega}$, $\boldsymbol{\mu}$ and $\boldsymbol{\eta}$ remain the same (using the extended $\mathbf{F}$), resulting in a block p-diagonal precision matrix.

B Experimental details

B.1 Advection-diffusion process

The advection-diffusion dataset is a random sample from a ST-DGMRF for which the transition matrices are defined according to an advection-diffusion process

$$\frac{\partial \rho(t,s)}{\partial t} = D\nabla^2\rho(t,s) - \nabla \cdot (\mathbf{v}\rho(t,s)) \tag{10}$$

with constant diffusion coefficient D and velocity vector $\mathbf{v} = [u, v]^T$. The process is discretized on a 30×30 lattice with grid cell size $\Delta x = \Delta y = 1$ and periodic boundary conditions. The spatial discretization results in a system of ordinary differential equations

$$\frac{\partial \boldsymbol{\rho}(t)}{\partial t} = \mathbf{M}\boldsymbol{\rho}(t), \tag{11}$$

where $\boldsymbol{\rho}(t)$ is a vector containing the system states of all grid cells. Using a finite difference discretization, matrix $\mathbf{M}$ is defined as

$$\mathbf{M}_{ij} = \begin{cases} D - \frac{1}{2}\mathbf{n}_{ij}^T\mathbf{v} & \text{if } d(i,j) = 1 \\ -4D & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where $d(i, j)$ denotes the distance between cell i and j, and $\mathbf{n}_{ij}$ denotes the unit vector pointing from lattice cell i to its neighbor j. For example, for cell $i = (s_x, s_y)$ and cell $j = (s_x, s_y - 1)$ it is $\mathbf{n}_{ij} = [0, -1]^T$, and thus $\mathbf{n}_{ij}^T\mathbf{v} = -v$.

Eq. 11 is converted into a discrete-time dynamical system by approximating

$$\boldsymbol{\rho}_{t+\Delta t} = \exp(\Delta t \cdot \mathbf{M})\boldsymbol{\rho}_t \approx \left(\sum_{k=0}^{3} \frac{1}{k!}(\Delta t)^k (\mathbf{M})^k\right)\boldsymbol{\rho}_t = \mathbf{F}_{\text{adv-diff}}\boldsymbol{\rho}_t \tag{13}$$

using a third-order Taylor series expansion. For simplicity, we use time resolution $\Delta t = 1$ resulting in

$$\mathbf{F}_{\text{adv-diff}} = \mathbf{I} + \mathbf{M} + \frac{1}{2}\mathbf{M}^2 + \frac{1}{6}\mathbf{M}^3. \tag{14}$$

B.1.1 Process simulation

We sample the initial state $\boldsymbol{\rho}_0$ from a GMRF with $\boldsymbol{\mu}_0 = \mathbf{0}$ and precision matrix $\mathbf{Q}_0 = \mathbf{S}_0^T\mathbf{S}_0$ with $\mathbf{S}_0 = (\mathbf{D} - \mathbf{A})$, where $\mathbf{A}$ is the adjacency matrix of the 4-nearest neighbour graph $\mathcal{G}_{\text{lattice}}$, and $\mathbf{D} = 4 \cdot \mathbf{I}$ is the corresponding degree matrix. This corresponds to a 1-layer DGMRF with parameters $\alpha = 1, \beta = -1, \gamma = 1$ and $b = 0$.

Starting from $\boldsymbol{\rho}_0$, we iteratively sample the next system state according to

$$\boldsymbol{\rho}_k = \mathbf{F}_k\boldsymbol{\rho}_{k-1} + \boldsymbol{\epsilon}_k \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k^{-1}) \tag{15}$$

with time-invariant transition matrix $\mathbf{F}_k = (\mathbf{F}_{\text{adv-diff}})^4$, where $\mathbf{F}_{\text{adv-diff}}$ is defined according to Eq. (14). This effectively aggregates four simulation steps into one, i.e. $\Delta t_k = (t_{k+1} - t_k) = 4$, resulting in larger differences between consecutive system states. For the noise terms $\boldsymbol{\epsilon}_k$, we use a time-invariant precision matrix $\mathbf{Q}_k = \mathbf{S}_k^T\mathbf{S}_k$ where $\mathbf{S}_k = (10 \cdot \mathbf{I} - \mathbf{A})$. This corresponds to a 1-layer DGMRF with parameters $\alpha = \frac{10}{4}, \beta = -1, \gamma = 1$ and $b = 0$.

We simulate for $K = 20$ time steps, using $D = 0.01$ and $\mathbf{v} = [-0.3, 0.3]^T$, and generate observations by masking out grid cells within a square of width $w \in \{6, \ldots, 12\}$ for 10 consecutive time steps and applying white noise with standard deviation $\sigma = 0.01$.

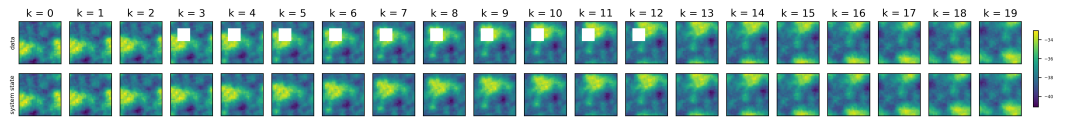

Figure 1: Advection-diffusion dataset with ground truth system states (bottom) and corresponding observations using masks of width $w = 9$ (top).

B.1.2 ST-DGMRF parameterization

We consider two ST-DGMRF variants that capture different amounts of prior knowledge. In both cases, spatial and temporal layers are defined based on $\mathcal{G}_{\text{lattice}}$, and temporal bias terms are, similar to spatial bias terms, defined as $\mathbf{b}_f^{(l)} = b_f^{(l)}\mathbf{1}$.

Variant 1 If prior knowledge is available in the form of a parameterized transition model, the ST-DGMRF transition matrices can be parameterized accordingly. Here, we consider temporal layers $\mathbf{F}_k^{(l)}$ that are a simplified first-order approximation to the true transition matrix $\mathbf{F}_{\text{adv-diff}}$ used to generate the data (see Eq. (14)), i.e. $\mathbf{F}_k^{(l)} = \mathbf{I} + \mathbf{M}^{(l)}$, with time-invariant learnable diffusion coefficients $D^{(l)}$ and velocity vectors $\mathbf{v}^{(l)}$. To ensure that the diffusion coefficient is non-negative, we model it as $D^{(l)} = (d^{(l)})^2$. This leaves us with four learnable parameters $d^{(l)}, u^{(l)}, v^{(l)}$ and $b_f^{(l)}$ per temporal layer.

Variant 2 If only partial knowledge about the underlying dynamics is available, the unknown parts can, for example, be replaced by a small neural network. Here, we consider temporal layers of the form $\mathbf{F}_k^{(l)} = \mathbf{I} + \mathbf{M}^{(l)}$ with

$$\mathbf{M}_{ij}^{(l)} = \begin{cases} (d^{(l)})^2 + \phi_{ij,1}^{(l)} & \text{if } j \in n(i) \\ -4(d^{(l)})^2 + \sum_{j \in n(i)} \phi_{ij,2}^{(l)} & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases} \tag{16}$$

where we define $\phi_{ij,1}^{(l)}, \phi_{ij,2}^{(l)} = f_{MLP}^{(l)}(\mathbf{n}_{ij})$ where $f_{MLP}^{(l)} : \mathbb{R}^2 \to \mathbb{R}^2$ is a multilayer perceptron (MLP) with one hidden layer of width 16 with ReLU non-linearity, and Tanh output non-linearity. Again, we define the transition model to be time-invariant and share MLP parameters across time and space. This amounts to 83 learnable parameters per temporal layer.

Log-determinant computations In our experiments, the spatial base graph $\mathcal{G}_{\text{lattice}}$ is small enough to pre-compute eigenvalues exactly and use the eigenvalue method for log-determinant computations proposed in [4].

Variational distribution For the variational distribution, we also consider two variants, one without temporal dependencies (equivalent to the DGMRF baseline) and one with a single temporal layer with time-invariant *diffusion* transition matrices $\tilde{\mathbf{F}}_k = \lambda \mathbf{I} + \omega(\mathbf{A} - \mathbf{D})$. Note that for $\omega = 0$, this reduces to a simple auto-regressive process.

Observation model All ST-DGMRF variants assume a temporally and spatially invariant observation noise level of $\sigma = 0.01$. The observation matrices $\mathbf{H}_k$ are defined as selection matrices matching the training masks during the learning phase and the training plus validation masks during the testing phase.

B.2 Air quality data

The air quality dataset is based on hourly PM2.5 measurements obtained from [7]. We consider 246 sensors within the metropolitan area of Beijing, China, for which we extracted time series of $K = 400$ hours between 13 March 2015 at 12pm and 30 March 2015 at 3am. Relevant weather covariates (surface temperature, as well as u and v wind components at 10 meters above ground level) were extracted from the ERA5 reanalysis dataset [3].

B.2.1 Data preprocessing

We define both the spatial and the temporal base graph based on the Delaunay triangulation of sensor locations, $\mathcal{G}_{\text{Delaunay}}$, where we disregard edges between sensors that are more than 160 kilometers apart. Edge weights are defined as the inverse distance between sensors, normalized to range between 0 and 1. The raw PM2.5 measurements are log-transformed and standardized to zero mean and unit variance. Finally, we remove clear outliers where the transformed values jump up and down by more than a threshold of $\delta = 2.0$ within three consecutive time steps. The ERA5 covariates are normalized to range between -1 and 1.

To mimic a realistic setting of repeatedly occurring partial network failures, we define our test set by masking out all measurements within a predefined spatial block (containing 50% of all sensors) within 10 randomly placed windows of 20 time steps (see Figure 2). Note that these windows may overlap, resulting in fewer periods of missing data with variable length. The masked out measurements are used for the final model evaluation.

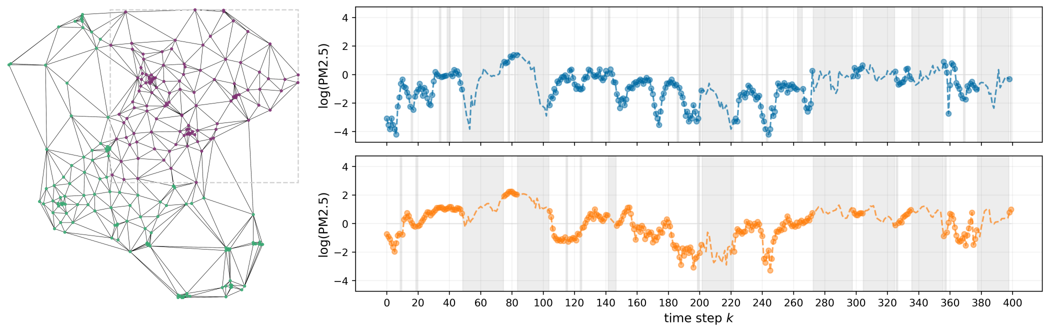

Figure 2: Left: air quality sensor network. Ca. 50% of the nodes are masked out (purple nodes within the gray box) during 10 randomly placed (partially overlapping) windows of 20 time steps. Right: associated log-transformed and normalized PM2.5 measurements for two sensors falling within the masked area. Time points that have either missing data or fall within a masked time window are shaded in gray.

B.2.2 ST-DGMRF parameterization

As with the advection-diffusion dataset, we consider two ST-DGMRF variants with different types of temporal layers. In both cases, spatial and temporal layers are defined based on the Delaunay triangulation described in Section B.2.1, and temporal bias terms $\mathbf{b}_f$ are defined in terms of a neural network mapping local weather covariates to temporally and spatially varying biases. We use a simple MLP with one hidden layer of width 16 with ReLU activations and no output non-linearity. The MLP parameters are shared over both space and time.

Variant 1 This variant accounts for directional transport processes, adopting a transition model similar to the neural network model used in the advection-diffusion experiments. In particular, we consider temporal layers of the form $\mathbf{F}_k^{(l)} = \mathbf{I} + \mathbf{M}_k^{(l)}$ with

$$\left(\mathbf{M}_k^{(l)}\right)_{ij} = \begin{cases} (d^{(l)})^2 + \phi_{k,ij}^{(l)} & \text{if } j \in n(i) \\ -4(d^{(l)})^2 + \sum_{j \in n(i)} \psi_{k,ij}^{(l)} & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

where we define $\phi_{k,ij}^{(l)}, \psi_{k,ij}^{(l)} = f_{MLP}^{(l)}\left(\mathbf{n}_{ij}, w_{ij}, (\mathbf{u}_k)_i\right)$ where $f_{MLP}^{(l)} : \mathbb{R}^6 \to \mathbb{R}^2$ is a MLP with one hidden layer of width 16 with ReLU activations, and Tanh output non-linearity. w_{ij} are the edge weights of the base graph (see Section B.2.1), and $(\mathbf{u}_k)_i$ is the vector of weather covariates for node i at time k. Since we use these time-dependent covariates as input to the MLP, the resulting transition model is not time-invariant anymore. However, the parameters of the MLP remain shared across time and space. As before, diffusion parameter $d^{(l)}$ is assumed to be spatially and temporally invariant.

Variant 2 The second variant uses highly simplified *diffusion* temporal layers of the form $\mathbf{F}_k^{(l)} = \lambda^{(l)}\mathbf{I} + \omega^{(l)}(\mathbf{A} - \mathbf{D})$ with spatially and temporally invariant parameters $\lambda^{(l)}$ and $\omega^{(l)}$.

Log-determinant computations Again, the spatial base graph $\mathcal{G}_{\text{Delaunay}}$ is small enough to pre-compute eigenvalues exactly and use the eigenvalue method for log-determinant computations [4].

Variational distribution As with the advection-diffusion dataset, we consider two variants for the variational distribution, one without temporal dependencies and one with a single temporal *diffusion* layer.

Observation model All ST-DGMRF variants assume a temporally and spatially invariant observation noise level of $\sigma = 0.01$. The observation matrices $\mathbf{H}_k$ are defined as selection matrices matching the training masks during the learning phase and the training plus validation masks during the testing phase.

B.3 Baseline models

B.3.1 DGMRF

We apply the DGMRF for general graphs introduced by [4] to each time frame of the time series, not accounting for temporal dependencies. The DGMRF parameters are not shared across time, allowing for dynamically changing spatial covariance patterns. We use one spatial layer in the variational distribution, as proposed in [4], and run a hyperparameter search over $L_{\text{spatial}} \in \{1, 2, 3\}$ with $L_{\text{spatial}} = 2$ performing best.

Including covariates In our experiments on the air quality dataset, for which we have access to relevant covariates, we follow [6] and add linear effects to the measurement model. Note that the vector of coefficients if shared across both space and time.

B.3.2 ARMA

We implemented ARMA(p, q) models with $p = 1$ and $q = 1$ for the advection-diffusion data, and with $p = 2$ and $q = 2$ for the air quality data, using the Python `statsmodels` package. For each node in the test set, maximum likelihood parameter estimation is performed based on the observed time points. Given the estimated model coefficients, we obtain posterior mean and variance estimates using the standard Kalman smoother [5]. As the maximum likelihood estimates are deterministic, we do not provide standard deviations of the evaluation metrics for these models.

B.3.3 ST-AR

The spatiotemporal autoregressive (ST-AR) model takes the form $\mathbf{x}_k = \alpha \cdot \mathbf{x}_{k-1} + \boldsymbol{\epsilon}_k$, with initial state $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and unconstrained spatial error terms $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$. We fix $\boldsymbol{\Sigma}_0 = 10 \cdot \mathbf{I}$ to encode high uncertainty about the initial state $\mathbf{x}_0$, and fit $\alpha, \boldsymbol{\mu}_0$ and $\mathbf{Q}^{-1}$ to the data using closed-form EM updates. The EM algorithm is initialized with $\alpha = 1, \boldsymbol{\mu}_0 = \mathbf{0}$ and $\mathbf{Q}^{-1} = \text{diag}(\mathbf{q})$ where elements $\mathbf{q}_i$ are drawn randomly from the interval $[5, 6]$. After convergence of the EM-algorithm, the final state estimates are obtained with the Kalman smoother [5].

B.3.4 EnKS

We consider an Ensemble Kalman Smoother (EnKS) variant for which the transition model matches the true data-generating process of the advection-diffusion dataset, as well as an EnKS variant for which we use a state augmentation approach to estimate unknown parameters $\mathbf{v}$ and $d = \sqrt{D}$ jointly with the system states. For both variants, we use 10^4 ensemble members (the maximum feasible on our machine). We fix the initial state distribution to $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, 10 \cdot \mathbf{I})$, and sample transition noise terms as $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, 0.1 \cdot \mathbf{I})$.

For the state augmentation approach, we define the initial distribution over velocities $\mathbf{v}$ as $\mathcal{N}(\boldsymbol{\mu}_v, 0.1 \cdot \mathbf{I})$, where $\boldsymbol{\mu}_v$ is randomly drawn from $[-1, 1]$ for each repeated run of the EnKS. Similarly, the initial distribution for diffusion parameter d is defined as $\mathcal{N}(\mu_d, 0.01)$ where μ_d is randomly drawn from $[0, 0.2]$ for each repeated run of the EnKS. Finally, the transition noise terms for parameters $\mathbf{v}$ and d are sampled from $\mathcal{N}(\mathbf{0}, 0.01 \cdot \mathbf{I})$.

B.3.5 MLP

For the air quality dataset, the MLP baseline maps local weather covariates $(\mathbf{u}_k)_i \in \mathbb{R}^3$ to log-transformed PM2.5 measurements. We use one hidden layer of width 16 with ReLU activations and no output non-linearity. The MLP parameters are shared over both space and time.

B.4 Regularized Conjugate Gradients

We use a regularized variant of the conjugate gradient (CG) method [1] to avoid slow convergence in the case of ill-conditioned matrices. Instead of directly solving a potentially ill-conditioned linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, the idea is to iteratively solve a sequence of regularized (i.e. well conditioned) linear systems

$$(\nu \mathbf{I} + \mathbf{A})\mathbf{x} = \nu \mathbf{x}^{(i)} + \mathbf{b}. \tag{18}$$

At each iteration, the solution from the previous iteration $\mathbf{x}^{(i)}$ is used to obtain the next solution $\mathbf{x}^{(i+1)}$. Eventually, this sequence will converge towards the true solution $\mathbf{x}^*$ of the original system $\mathbf{A}\mathbf{x} = \mathbf{b}$.

We start with $\nu = 10$ and decrease it every 10 iterations by factor 10. In each iteration, the standard CG method is employed to iteratively solve the regularized linear system until the residual norm drops below a threshold of 10^{-7} or a maximum of 200 inner CG iterations is reached. This inner loop is repeated until the norm of the residuals

$$\mathbf{r}^{(i)} = \left((\nu\mathbf{I} + \mathbf{A})\mathbf{x}^{(i)}\right) - \left(\nu\mathbf{x}^{(i)} + \mathbf{b}\right) \tag{19}$$

drops below a threshold of 10^{-7} or a maximum of 100 outer iterations is reached. The initial guess $\mathbf{x}^{(0)}$ is given by the mean of the variational distribution $q_\phi(\mathbf{x})$.

C Additional results

In this section, we present additional results regarding the scalability of our approach (Section C.1), and provide more detailed results for the experiments in Section 4 of the main paper (Section C.2 and C.3). Finally, in Section C.4 we provide estimates of the total computation time required for our experiments.

C.1 Scalability

To empirically demonstrate the scalability of our method, we generate additional advection-diffusion datasets with varying lattice size and compare the runtime of our ST-DGMRF approach to a naive Kalman smoother (KS) [5] approach. To this end, we consider a model with advection-diffusion transition matrix using $L_{\text{temporal}} = 2$ temporal and $L_{\text{spatial}} = 2$ spatial layers. To avoid additional matrix inversions in the KS approach, we set the spatial and temporal bias terms $\mathbf{b}_s, \mathbf{b}_f$ to zero, resulting in $\boldsymbol{\mu}_0 = \mathbf{0}$ and $\mathbf{c}_k = \mathbf{0}$. We train the model for 1 000 iterations and measure the average wall clock time per iteration. In addition, we measure the wall clock time needed to perform inference with the trained model.

ST-DGMRF For the ST-DGMRF approach, we proceed as before using a variational distribution with one temporal diffusion layer during training. For better comparability, we employed the standard (non-regularized) CG method (with a tolerance of 10^{-7}) to compute the posterior mean and marginal variances (based on 100 CG samples) and provide measurements of the average time per CG iteration instead of the total time needed to perform inference. Multiplying this with the average number of CG iterations needed until convergence results in an estimate of the average total time for inference.

Kalman smoother For the KS approach, instead of approximating the true posterior with a variational distribution and estimating the ELBO based on Monte-Carlo samples, we use the KS to obtain exact marginal posterior estimates, which are used to compute expectations in closed form. The marginal covariance and transition matrices required for the KS equations are extracted from the ST-DGMRF model in every iteration. The associated parameters are then optimized via a form of Generalized EM-algorithm [2], where in each iteration a single gradient ascent step is taken.

Results Figure 3 shows how ST-DGMRF and KS training and inference scale as the number of nodes N in the system increases. Clearly, the time per training iteration increases super-linearly when the KS is used to obtain exact marginal posterior distributions, while the variational ST-DGMRF training time increases only marginally and remains below the fastest KS iteration for all tested N. In addition, KS memory requirements (due to storing K dense $N \times N$ covariance matrices) exceeded the available GPU memory for $N > 1024$, making this approach infeasible for larger systems. In contrast, the ST-DGMRF exploits the sparsity of spatial and temporal graph-structured layers and thereby avoids storing dense matrices, remaining feasible for $N \gg 1024$.

For the tested systems, ST-DGMRF posterior inference with the CG method is slower than exact KS inference. However, the memory requirements of the KS approach again limit its feasibility to $N \leq 1024$, while the CG method only requires storing vectors of size $\mathcal{O}(N)$ making it feasible for $N \gg 1024$. Moreover, Figure 3 confirms that computations per CG iteration scale linearly in N. And since the number of CG iterations required for convergence remains approximately constant, the

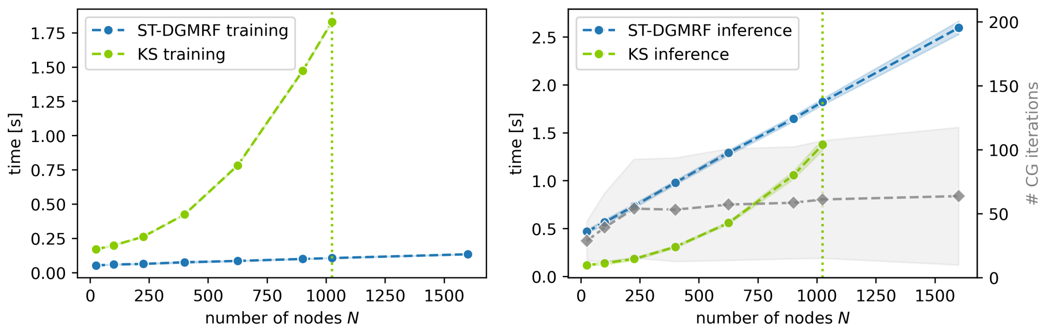

Figure 3: Comparison of ST-DGMRF and KS computation time in seconds for training (per iteration) and inference. For ST-DGMRF inference, the time per CG iteration is plotted together with the average number of CG iterations needed to converge (in gray). All quantities are plotted as mean $\pm$ std based on 5 runs with different random seeds. The vertical dotted lines indicate the maximum N for which the KS approach was applicable.

total computation time for CG inference also scales linearly in N. In contrast, KS inference again scales super-linearly. This means that even if the KS approach would remain feasible in terms of memory requirements, its computation time will quickly approach, and eventually exceed, the time needed for CG inference.

C.2 Advection-diffusion experiments

Table 1 summarizes all results for the advection-diffusion dataset with mask size $w = 9$, including standard deviations for all metrics based on 5 runs with different random seeds. As discussed in the main paper, the ST-DGMRF variants provide more accurate posterior estimates than the baselines relying on simplified spatiotemporal dependency structures.

Ablation results Table 1 contains additional results for the ST-DGMRF variants using different settings for the variational distribution (see Section B.1.2) For this dataset, we do not find a significant effect of introducing temporal dependencies in the variational distribution. Further, Figure 4 shows additional results for the ST-DGMRF variants when varying the number of temporal layers L_{temporal}. For all metrics, the performance improves significantly as we start adding temporal layers and stabilizes around $L_{\text{temporal}} = 3$. Note that around the same point, both ST-DGMRF variants converge towards the EnKS using the true data-generating dynamics, in terms of the RMSE$_\mu$, and even drop below it in terms of the CRPS. Only in terms of RMSE$_\sigma$, the ST-DGMRF models remain inferior to both EnKS variants. We hypothesize that increasing the expressivity (i.e. L_{spatial}) of the noise terms can further reduce this gap.

Table 1: Model performance for the advection-diffusion dataset with $w = 9$, reported as mean $\pm$ std over 5 runs with different random seeds. All ST-DGMRF variants use $L_{\text{spatial}} = 2$ and $L_{\text{temporal}} = 4$.

	VI dynamics	RMSE$_\mu$ ↓	RMSE$_\sigma$ ↓	CRPS ↓
ARMA	–	2.3054 –	0.6812 –	1.7064 –
ST-AR	–	1.4595$\pm$0.0098	1.9216$\pm$1.0392	0.9707$\pm$0.0163
DGMRF	–	0.5901$\pm$0.0037	0.3808$\pm$0.0010	0.3495$\pm$0.0022
EnKS				
true dynamics	–	0.0661$\pm$0.0030	0.0046$\pm$0.0000	0.1027$\pm$0.0035
estimated dynamics	–	0.1654$\pm$0.2031	0.0039$\pm$0.0005	0.1434$\pm$0.0902
ST-DGMRF (ours)				
advection-diffusion	*none*	0.0526$\pm$0.0001	0.1148$\pm$0.0003	0.0726$\pm$0.0001
advection-diffusion	*diffusion*	0.0526$\pm$0.0001	0.1146$\pm$0.0005	0.0726$\pm$0.0000
neural network	*none*	0.0839$\pm$0.0022	0.1334$\pm$0.0089	0.0833$\pm$0.0008
neural network	*diffusion*	0.0854$\pm$0.0027	0.1402$\pm$0.0061	0.0839$\pm$0.0008

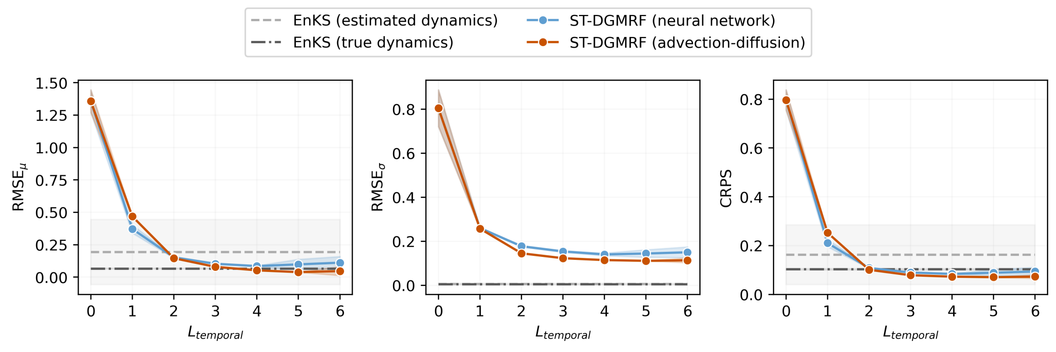

Figure 4: RMSE$_\mu$, RMSE$_\sigma$ and CRPS as a function of the number of temporal layers L_{temporal} for the advection-diffusion dataset with $w = 9$, plotted as mean $\pm$ std over 5 runs with different random seeds. Both ST-DGMRF variants are trained with a variational distribution using one temporal *diffusion* layer. Note that $L_{\text{temporal}} = 0$ corresponds to the spatial-only DGMRF baseline.

C.3 Air quality experiments

Table 2 summarizes all results for the air quality dataset. It contains additional results for the ST-DGMRF variants using different settings for the variational distribution (see Section B.2.2), and provides standard deviations for all metrics based on 5 runs with different random seeds.

Ablation results We find that, in contrast to our experiments on the advection-diffusion data, accounting for temporal dependencies in the variational distribution is clearly beneficial in the real world setting. Especially for the ST-DGMRF with neural network based transitions, adding the temporal *diffusion* layer results in significantly improved posterior estimates, and at the same time reduces the variability across different runs. Further, we find that at least two temporal layers are needed to achieve good posterior estimates that improve on the baselines (see Figure 5).

Example model outputs Figure 6 shows state estimates and associated uncertainties together with sensor measurements for two example sensors within the masked out area of the network, for ST-DGMRF, DGMRF and ARMA respectively. For all three models, state estimates are obtained by conditioning on the input data points (used for training), resulting in low errors and uncertainties for observed time points and higher errors and uncertainties for masked out time points. Moreover, for both ST-DGMRF and DGMRF, higher uncertainties coincide with larger errors and larger fluctuations in the measurements (top), while more accurate state estimates come with smaller uncertainties (bottom). Finally, Figure 7 visualizes how spatial and temporal ST-DGMRF layers transform samples from the estimated posterior over system states into (approximately) independent Gaussian noise, as derived in Section 3.1.2 and visualized in Figure 1. Clearly, temporal layers remove daily patterns and overall trends, while spatial layers remove dependencies between close-by sensors and increase temporal fluctuations.

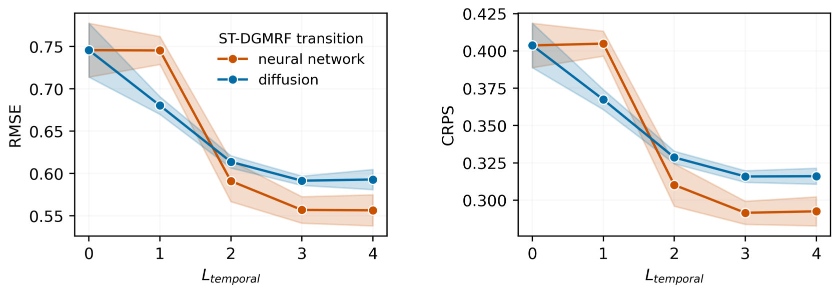

Figure 5: RMSE and CRPS for increasing L_{temporal} (mean $\pm$ std over 5 runs). Both models use $p = 2$. As before, $L_{\text{temporal}} = 0$ corresponds to the spatial-only DGMRF baseline.

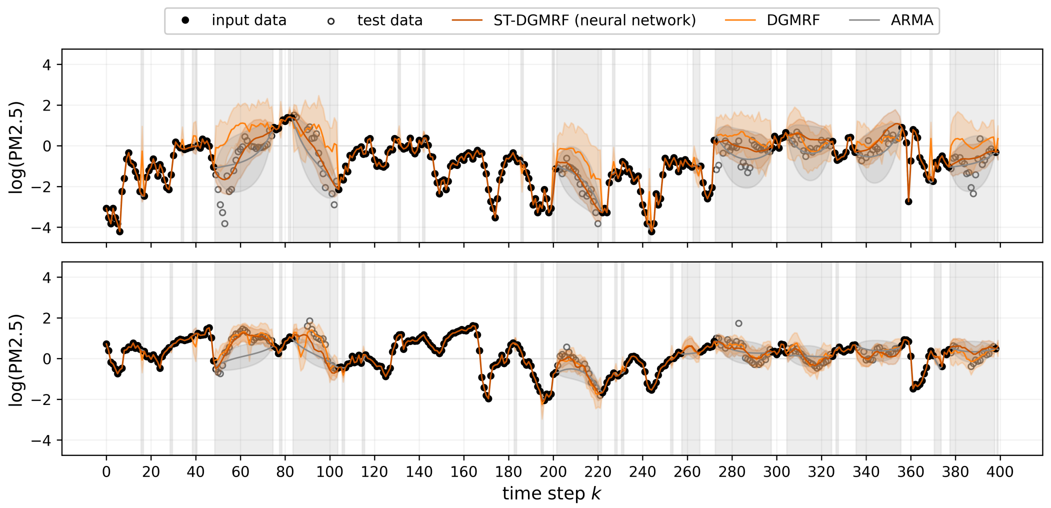

Figure 6: Model outputs for two air quality sensors falling within the masked area. Solid lines represent posterior mean estimates, while shaded areas represent posterior std estimates. Time points that have either missing data or fall within the masked time window are shaded in gray.

Table 2: Model performance for the air quality dataset, reported as mean ± std over 5 runs with different random seeds. All ST-DGMRF variants use $L_{\text{spatial}} = 2$ and $L_{\text{temporal}} = 4$.

	p	VI dynamics	RMSE ↓	CRPS ↓
ARMA	–	–	0.6820 –	0.3625 –
ST-AR	–	–	0.7350±0.0006	0.4261±0.0003
DGMRF	–	–	0.7368±0.0135	0.3966±0.0032
MLP	–	–	0.8038±0.0245	–
ST-DGMRF (ours)				
diffusion	1	*none*	0.6147±0.0082	0.3239±0.0058
diffusion	1	*diffusion*	0.6190±0.0073	0.3258±0.0043
diffusion	2	*none*	0.6020±0.0112	0.3214±0.0051
diffusion	2	*diffusion*	0.5928±0.0119	0.3161±0.0054
neural network	1	*none*	0.5995±0.0887	0.3147±0.0494
neural network	1	*diffusion*	0.5853±0.0457	0.3092±0.0257
neural network	2	*none*	0.5825±0.0626	0.3062±0.0353
neural network	2	*diffusion*	0.5565±0.0184	0.2925±0.0097

C.4 Total compute

Most computations were performed on a Nvidia Titan X GPU. On top of the final experiments, we performed hyperparameter sweeps and additional test runs. Here, we provide estimates of the total compute time grouped by experiment:

Advection-diffusion dataset:

- **Performance comparison & ablations:** ca. 50 GPU hours
- **Varying mask size:** ca. 30 GPU hours per w, resulting in ca. 210 GPU hours in total
- **Scalability:** ca. 15 GPU hours

Air quality dataset:

- **Performance comparison & ablations:** ca. 100 GPU hours

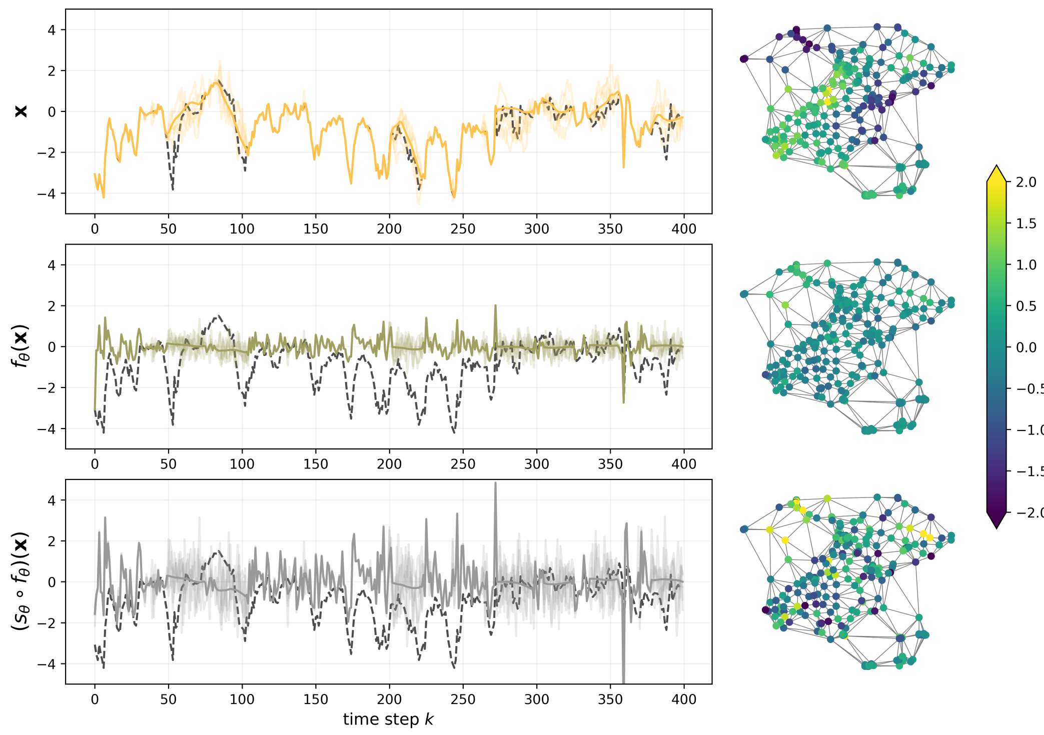

Figure 7: Effects of applying temporal ($\mathbf{f}_\theta$) and spatial ($\mathbf{s}_\theta$) ST-DGMRF layers to the predictive posterior. Left column: The dark yellow line (top row) shows the estimated posterior mean for an air quality sensor falling within the masked area. Light yellow lines represent corresponding posterior samples. Similarly, green lines (center row) represent states after applying the temporal transformation ($\mathbf{f}_\theta$), and gray lines (bottom row) represent states after applying both temporal and spatial layers ($\mathbf{s}_\theta \circ \mathbf{f}_\theta$). As a reference, we also plot ground truth log-transformed and normalized PM2.5 measurements (dashed black lines). Right column: corresponding (transformed) states for all sensors at time $k = 100$.

References

[1] Z. Z. Bai and S. L. Zhang. A regularized conjugate gradient method for symmetric positive definite system of linear equations. *Journal of Computational Mathematics*, pages 437–448, 2002.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.

[3] H. Hersbach, B. Bell, P. Berrisford, S. Hirahara, A. Horányi, J. Muñoz-Sabater, J. Nicolas, C. Peubey, R. Radu, D. Schepers, et al. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

[4] J. Oskarsson, P. Sidén, and F. Lindsten. Scalable deep Gaussian Markov random fields for general graphs. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

[5] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

[6] P. Sidén and F. Lindsten. Deep Gaussian Markov random fields. In *International Conference on Machine Learning*. PMLR, 2020.

[7] Y. Zheng, X. Yi, M. Li, R. Li, Z. Shan, E. Chang, and T. Li. Forecasting fine-grained air quality based on big data. In *Proceedings of the 21th SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD 2015, August 2015.