Adaptive Filter Attention:
Precision-Weighted State Estimation with Learned Dynamics

Peter Racioppo

June 2025

Note: This is an expanded version of an MS thesis I submitted to UCLA in June 2025, which can be found at the UCLA archive, at: eScholarship. This expanded version includes new sections on a simplified scalable algorithm, a natural gradient interpretation, and an expanded 'Related Work' section.

ABSTRACT:

This work introduces Adaptive Filter Attention (AFA), a novel attention mechanism that explicitly incorporates a learnable linear time-invariant dynamics model into the attention computation to improve the estimation of noisy trajectories in stochastic dynamical systems. Unlike standard self-attention, which relies on nonlocal averaging based solely on feature similarity, AFA computes the attention weights using a state-space model that models how latent states evolve over time. The similarity of each query-key pair is modeled by how well the key aligns with the query under the learned dynamics, weighting interactions by their consistency with the system's temporal evolution. This structure enables AFA to distinguish signal from noise based on the plausibility of dynamic transitions. By incorporating a learnable dynamics model, AFA functions as a model-based adaptive filter, introducing a principled inductive bias that should improve generalization in time series modeling. This view bridges adaptive filtering with modern attention architectures and enables principled incorporation of structured uncertainty.

CHAPTER 1

Introduction

1.1 Motivation: Attention for filtering stochastic dynamical systems

Self-attention is the dominant paradigm for sequence modeling. By computing pairwise interactions across all time steps in parallel, it achieves major improvements in training speed and scalability over recurrent neural networks (RNNs), which process sequences step by step. However, this parallelism comes at the cost of losing the recursive structure that RNNs naturally impose. In RNNs, hidden states evolve sequentially, providing an implicit temporal regularization that encourages consistency over time. In contrast, self-attention computes pairwise similarities across all time steps independently, without explicitly modeling how states evolve over time. While this allows for richer dependencies and long-range interactions, it can make learning dynamics more difficult—especially in noisy or underdetermined settings.

State space models (SSMs) offer a promising compromise. They retain RNN-like temporal structure by applying shared linear dynamics across time steps, but are implemented via convolutional operators that enable parallel training. This provides both temporal regularization and scalability. However, standard SSMs assume deterministic dynamics, limiting their ability to model uncertainty—a key feature of many real-world systems, which are

better captured as stochastic dynamical systems.

Self-attention is well-suited for filtering time series: it enables selective averaging over inputs, reducing noise by aggregating information from similar states. However, while attention implicitly encodes dynamics through learned query, key, and value matrices, it does not explicitly enforce consistency with the system's evolution, unlike state space models. Thus, the model may represent dynamics inconsistently over time, which may worsen their ability to generalize. This motivates incorporating structure, such as known or learned dynamics, into attention to restore some of the inductive bias lost with the removal of recurrence.

We introduce **Adaptive Filter Attention**, a novel mechanism that integrates the explicit structure of dynamical systems and principles of adaptive filtering into the attention process by applying a linear transformation that models state evolution. Instead of computing attention weights by directly comparing the inputs, the model compares them after propagating one through the learned dynamics model to the time frame of the other. This brings the inductive bias of state space models into the attention computation, enabling more consistent filtering in stochastic dynamical systems.

CHAPTER 2

Background

2.1 Linear Systems

A continuous-time linear system is defined, in state-space form, by:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t)$$

$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t)$$

The signals:

$$\mathbf{x}(t) : [0, \infty) \to \mathbb{R}^N, \quad \mathbf{u}(t) : [0, \infty) \to \mathbb{R}^M, \quad \mathbf{y}(t) : [0, \infty) \to \mathbb{R}^P$$

are known as the *state*, *input*, and *output*, respectively. In control applications, $\mathbf{u}(t)$ is usually a feedback control term. $\mathbf{A}(t), \mathbf{B}(t), \mathbf{C}(t), \mathbf{D}(t)$ are known as the state matrix, input matrix, output matrix, and direct transition matrix, respectively.

Letting $\mathbf{x}(t_0) = \mathbf{x}_0$, the unique solution to the above system is given by:

$$\mathbf{x}(t) = \mathbf{\Phi}(t, t_0)\mathbf{x}_0 + \int_{t_0}^{t} \mathbf{\Phi}(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)\, d\tau,$$

$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{\Phi}(t, t_0)\mathbf{x}_0 + \int_{t_0}^{t} \mathbf{C}(t)\mathbf{\Phi}(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)\, d\tau + \mathbf{D}(t)\mathbf{u}(t),$$

where $\mathbf{\Phi}(t, t_0)$, the *state transition matrix*, is the unique solution to

$$\frac{d}{dt}\mathbf{\Phi}(t, t_0) = \mathbf{A}(t)\mathbf{\Phi}(t, t_0),$$

with the initial condition $\mathbf{\Phi}(t_0, t_0) = \mathbf{I}$.

A linear time-invariant (LTI) system is one in which the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are fixed, real-valued matrices, (i.e. not functions of time): $\mathbf{A} \in \mathbb{R}^{N\times N}$, $\mathbf{B} \in \mathbb{R}^{N\times M}$, $\mathbf{C} \in \mathbb{R}^{P\times N}$, $\mathbf{D} \in \mathbb{R}^{P\times M}$.

An LTI system $\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t)$ is *asymptotically stable* (i.e. $\mathbf{x}(t)$ approaches a stable equilibrium point as $t \to \infty$) if $\mathbf{A}$ is Hurwitz, that is if every eigenvalue of $\mathbf{A}$ has strictly negative real part. Note that linear systems can have no more than one equilibrium point.

In an LTI system, the state transition matrix is given by:

$$\mathbf{\Phi}(t, t_0) = e^{\mathbf{A}(t-t_0)},$$

where the matrix exponential $e^{\mathbf{M}}$ is defined as:

$$e^{\mathbf{M}} = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{M}^k.$$

It follows that the unique solution to an LTI system is:

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t_0)}\mathbf{x}_0 + \int_{t_0}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)\, d\tau.$$

$$\mathbf{y}(t) = \mathbf{C}e^{\mathbf{A}(t-t_0)}\mathbf{x}_0 + \int_{t_0}^{t} \mathbf{C}e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)\, d\tau + \mathbf{D}\mathbf{u}(t),$$

See also: (Ant97), (SP05)

2.2 State Space Models

We can write the above linear system in discrete time as:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k$$

The discrete-time system $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k$ is asymptotically stable if the spectral radius of $\mathbf{A}$ is less than 1:

$$\rho(\mathbf{A}) = \max\{|\lambda_i|, \lambda_i \in \text{eig}(\mathbf{A})\} < 1.$$

We can improve the accuracy of the discrete system with the bilinear (Tustin) transformation:

$$\bar{\mathbf{A}} = (\mathbf{I} - \frac{\Delta}{2}\mathbf{A})^{-1}(\mathbf{I} + \frac{\Delta}{2}\mathbf{A})$$

$$\bar{\mathbf{B}} = (\mathbf{I} - \frac{\Delta}{2}\mathbf{A})^{-1}\Delta\mathbf{B}$$

$$\bar{\mathbf{C}} = \mathbf{C}$$

where Δ is the discretization step. This is essentially a linear RNN, where $\mathbf{x}_k$ is the hidden state and $\mathbf{y}_k$ is the model output, and $\bar{\mathbf{D}}\mathbf{u}_k$ is a residual connection. For now, we omit $\bar{\mathbf{D}}$ for simplicity.

The advantage of modeling the dynamics as a linear system is that we can unroll the recursion relation explicitly:

$$\mathbf{y}_k = \bar{\mathbf{C}}\bar{\mathbf{A}}^k\mathbf{x}_0 + \sum_{j=1}^{k}\bar{\mathbf{C}}\bar{\mathbf{A}}^{k-j}\bar{\mathbf{B}}\mathbf{u}_j = \sum_{j=0}^{k}\bar{\mathbf{h}}_j\mathbf{u}_{k-j}$$

with $\bar{\mathbf{h}}_j = \bar{\mathbf{C}}\bar{\mathbf{A}}^j\bar{\mathbf{B}}$. Hence, the output sequence can be expressed as a convolution:

$$\mathbf{y} = \bar{\mathbf{h}} * \mathbf{u}$$

with convolutional kernel: $\bar{\mathbf{h}} = [\bar{\mathbf{C}}\bar{\mathbf{B}}, \bar{\mathbf{C}}\bar{\mathbf{B}}\bar{\mathbf{A}}, \ldots, \bar{\mathbf{C}}\bar{\mathbf{B}}\bar{\mathbf{A}}^{L-1}]$, where $\bar{\mathbf{h}}_j$ is known as the impulse response at time j. Convolution can be performed more efficiently in the frequency domain, with the use of the Fourier transform.

Efficient Computation Using the HiPPO Framework:

SSMs still suffer from the vanishing/exploding gradients problem. The High-Order Polynomial Projection Operator (HiPPO) framework deals with this problem with a careful initialization of the state transition matrix. (GDE20)

The specific choice of $\mathbf{A}$ significantly impacts performance in sequence modeling tasks. HiPPO defines $\mathbf{A}$ using an orthogonal polynomial basis, with the goal of optimally maintaining a summary of the past in a continuous-time setting. The Legendre basis is optimal for this because it minimally distorts the information while compressing it into a fixed-dimensional state. The HiPPO matrix encodes the coefficients of a Legendre polynomial:

$$\mathbf{A}_{ij} = \begin{cases} (2i+1)^{1/2}(2j+1)^{1/2}, & i > j \\ i+1, & i = j \\ 0, & i < j \end{cases}$$

The authors show that "gradient norms of the model decay polynomially in time, instead of exponentially."

Structured State Space for Sequence Modeling (S4):

As the authors note, "The fundamental bottleneck in computing the discrete-time SSM is that it involves repeated matrix multiplication by $\bar{\mathbf{A}}$." (GGR22)

Instead of computing the recurrence explicitly, we can use the Z-transform to compute it as a convolution in the frequency domain. The Z-transform is the discrete-time equivalent of the Laplace transform. It converts a time-domain sequence $\mathbf{x}_k$ into a function of a complex variable z.

$$\mathbf{Y}(z) = \sum_{k=0}^{\infty} \mathbf{y}_k z^{-k}$$

Applying this to our convolution:

$$\mathbf{Y}(z) = \sum_{k=0}^{\infty} \sum_{i=0}^{k} \mathbf{h}_i \mathbf{u}_{k-i} z^{-k}$$

Changing the order of the summation:

$$\mathbf{Y}(z) = \sum_{i=0}^{\infty} \mathbf{h}_i z^{-i} \sum_{k=i}^{\infty} \mathbf{u}_{k-i} z^{-(k-i)} := \hat{\mathcal{K}}(z)\mathbf{U}(z)$$

where the convolution kernel (transfer function) is:

$$\hat{\mathcal{K}}(z) = \sum_{i=0}^{\infty} \bar{\mathbf{C}} \bar{\mathbf{A}}^i \bar{\mathbf{B}} z^{-i}$$

The authors show that the kernel can be simplified to:

$$\hat{\mathcal{K}}_L = \tilde{\mathbf{C}}(\mathbf{I} - \bar{\mathbf{A}}z)^{-1}\bar{\mathbf{B}}$$

If $\mathbf{A}$ is equal to a diagonal matrix $\mathbf{\Lambda}$, $\hat{\mathcal{K}}_L$ becomes:

$$\hat{\mathcal{K}}_{\mathbf{\Lambda}}(z) = c(z) \sum_i \frac{\mathbf{C}_i \mathbf{B}_i}{g(z) - \mathbf{\Lambda}_i} = c(z) \cdot k_{z,\mathbf{\Lambda}}(\mathbf{C}, \mathbf{B})$$

We can relax the assumption of diagonal $\mathbf{A}$ to a *Diagonal Plus Low-Rank* (DPLR) structure:

$$\mathbf{A} \approx \mathbf{\Lambda} - \mathbf{P}\mathbf{Q}^*$$

where $\mathbf{\Lambda}$ is a diagonal matrix, and $\mathbf{P}$ and $\mathbf{Q}$ are low-rank matrices.

We can use the matrix inversion lemma (also known as the Sherman–Morrison–Woodbury

formula) to comptue $\hat{\mathcal{K}}$:

$$(\mathbf{A}+\mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1}+\mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}$$

i.e. the inverse of the sum is the inverse of $\mathbf{A}$ minus a correction term, which depends on the inverse of a possibly lower-rank matrix. In the case of a rank-1 update, this reduces to the Sherman-Morrison formula:

$$(\mathbf{A}+\mathbf{uv}^T)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{uv}^T\mathbf{A}^{-1}}{1+\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}$$

In our case,

$$(\mathbf{\Lambda}+\mathbf{PQ}^*)^{-1} = \mathbf{\Lambda}^{-1} - \mathbf{\Lambda}^{-1}\mathbf{P}(\mathbf{I}+\mathbf{Q}^*\mathbf{\Lambda}^{-1}\mathbf{P})^{-1}\mathbf{Q}^*\mathbf{\Lambda}^{-1}$$

The authors show that, for rank-1 $\mathbf{P}$ and $\mathbf{Q}$,

$$\hat{K}_{\text{DPLR}}(z) = c(z)\left[k_{z,\mathbf{\Lambda}}(\mathbf{C},\mathbf{B}) - k_{z,\mathbf{\Lambda}}(\mathbf{C},\mathbf{P})(\mathbf{I}+k_{z,\mathbf{\Lambda}}(\mathbf{q}^*,\mathbf{P}))^{-1}k_{z,\mathbf{\Lambda}}(\mathbf{q}^*,\mathbf{B})\right]$$

Recall that a matrix $\mathbf{A}$ is called normal if it commutes with its conjugate transpose: $\mathbf{AA}^T = \mathbf{A}^T\mathbf{A}$. A matrix is normal if and only if it is unitarily diagonalizable, i.e $\mathbf{A} = \mathbf{U\Lambda U}^*$, for some diagonal $\mathbf{\Lambda}$ and unitary $\mathbf{U}$. A *Normal Plus Low-Rank* (NPLR) matrix can be written in the form:

$$\begin{aligned}\mathbf{A} &= \mathbf{U\Lambda U}^* - \mathbf{PQ}^* \\ &= \mathbf{U}(\mathbf{\Lambda} - \mathbf{U}^*\mathbf{PQ}^*\mathbf{U})\mathbf{U}^* \\ &= \mathbf{U}(\mathbf{\Lambda} - \mathbf{U}^*\mathbf{P}(\mathbf{U}^*\mathbf{Q})^*)\mathbf{U}^* \\ \mathbf{A} &= \mathbf{U}\left(\mathbf{\Lambda} - \tilde{\mathbf{P}}\tilde{\mathbf{Q}}^*\right)\mathbf{U}^*\end{aligned}$$

Hence, any NPLR matrix is unitarily similar to a DPLR matrix. This allows the above to be extended to NPLR matrices. One such NPLR matrix is the HiPPO matrix.

Note that we can ensure a matrix is unitary by constructing a skew-symmetric matrix

$\mathbf{S}$ from a lower triangular matrix $\mathbf{L}$: $\mathbf{S} = \mathbf{L} - \mathbf{L}^T$, and then constructing a unitary matrix by computing: $\mathbf{U} = e^{\mathbf{S}}$.

A follow-up work (GGD22) found that the algorithm is more stable if $\mathbf{A}$ is instead defined as:

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^* - \mathbf{P}\mathbf{P}^*$$

Note that a matrix $\mathbf{A} = \mathbf{\Lambda} - \mathbf{P}\mathbf{P}^*$ is Hurwitz if all entries of $\mathbf{\Lambda}$ have negative real part, so this parameterization makes it easier to control the spectrum of $\mathbf{A}$ (and hence the stability of thee system).

The Mamba models make additional improvements to the S4 model, resulting in performance that is competitive with transformers. (GD24)

Much of the above explanation was adapted from this useful blog: The Annotated S-4

2.3 Stochastic Linear Systems

Many phenomena must be modeled as stochastic rather than deterministic. Consider an LTI dynamical system with zero-mean Gaussian white noise:

$$\mathbf{x}_i = \mathbf{A}\mathbf{x}_{i-1} + \mathbf{B}\mathbf{u}_{i-1} + \boldsymbol{\eta}_i,$$

$$\mathbf{z}_i = \mathbf{C}\mathbf{x}_i + \boldsymbol{\gamma}_i,$$

$$\text{with } \mathbf{x}_0 = \bar{\mathbf{x}}_0 + \boldsymbol{\eta}_0,$$

where:

$$\boldsymbol{\eta}_i \sim \mathcal{N}(0, \mathbf{\Omega}_i), \quad \boldsymbol{\gamma}_i \sim \mathcal{N}(0, \mathbf{\Gamma}_i).$$

We refer to $\boldsymbol{\eta}_i$ as *process noise* and $\boldsymbol{\gamma}_i$ as *measurement noise*. In what follows, we will take $\boldsymbol{\Omega}_i = \boldsymbol{\Omega}$ and $\boldsymbol{\Gamma}_i = \boldsymbol{\Gamma}$, $\forall i$. (Spe08)

2.4 The Kalman Filter

Filtering or *state estimation* is the task of estimating the state variables $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ from a sequence of noisy measurements $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$.

The Kalman filter is the optimal recursive estimator for a *known* linear system with Gaussian white noise. It operates sequentially, updating the state estimate as new data arrives, rather than performing a global optimization. It operates in two main steps:

Prediction: The estimated state evolves according to the known dynamics:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k$$

$$\mathbf{V}_{k|k-1} = \mathbf{A}\mathbf{V}_{k-1}\mathbf{A}^T + \boldsymbol{\Omega}$$

Update: Update the estimates based on measurements.

Compute the *Kalman gain*:

$$\mathbf{K}_k = \mathbf{V}_{k|k-1}\mathbf{C}^T(\mathbf{C}\mathbf{V}_{k|k-1}\mathbf{C}^T + \boldsymbol{\Gamma})^{-1}$$

Then update the state estimate:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{C}\hat{\mathbf{x}}_{k|k-1})$$

And update the covariance:

$$\mathbf{V}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{C})\mathbf{V}_{k|k-1}$$

We refer to $\hat{\mathbf{x}}_{k|k-1}$ and $\mathbf{V}_{k|k-1}$ as the *a priori* estimates of the state and covariance, and $\hat{\mathbf{x}}_k$ and $\mathbf{V}_k$ as the *a posteriori* estimates.

In continuous time, the covariance is updated via the differential Lyapunov equation:

$$\dot{\mathbf{V}}(t) = \mathbf{A}(t)\mathbf{V}(t) + \mathbf{V}(t)\mathbf{A}(t)^T - \mathbf{V}(t)\mathbf{C}(t)^T\mathbf{\Gamma}^{-1}\mathbf{C}(t)\mathbf{V}(t) + \mathbf{\Omega}(t)$$

Intuitively, the Kalman filter balances *prediction from a model* and *correction from measurements*, weighting them based on their respective uncertainties. (Spe08)

2.5 Adaptive Filtering

In the Kalman filter, we compute an estimate covariance that depends entirely on the known dynamics. In contrast, in *adaptive filtering*, we optimize a function of the (noisy) input signal $\mathbf{x}_n$ to minimize the mean squared error between the signal and a desired output $\mathbf{d}_n$ (usually, with the purpose of denoising). These methods are called adaptive because the models adjust dynamically in response to variations in the system's behavior or the environment, whereas the Kalman filter assumes unchanging dynamics. Adaptive filtering is generally model-free. As it can be done online, it allows for rapid adaptation to data. Learning dynamics models from data is called *system identification*. It is generally done in batches, usually offline.

The two main classes of linear adaptive filters are Least Mean Squares (LMS) and Recursive Least Squares (RLS) filters. LMS iteratively updates filter coefficients using a simple gradient descent approach. Given an input vector $\mathbf{x}_k$ and filter weights $\mathbf{w}_k$ at the kth time step, the output is:

$$y_k = \mathbf{w}_k^T\mathbf{x}_k$$

The error signal is:

$$e_k = d_k - y_k$$

where d_n is the desired signal. Define a loss function:

$$\xi^2 = \mathbb{E}\left[e_k^2\right] = \mathbb{E}\left[d_k^2 + y_k^2 - 2d_k y_k\right]$$

$$= \mathbb{E}\left[d_k^2\right] + \mathbf{w}_k^T \mathbb{E}\left[\mathbf{x}_k \mathbf{x}_k^T\right] \mathbf{w}_k - 2\mathbb{E}\left[d_k \mathbf{x}_k^T\right] \mathbf{w}_k$$

Let $\mathbf{R} = \mathbb{E}\left[\mathbf{x}_k \mathbf{x}_k^T\right]$ and $\mathbf{P} = \mathbb{E}\left[d_k \mathbf{x}_k\right]$ so that:

$$\xi^2 = \mathbb{E}\left[d_k^2\right] + \mathbf{w}_k^T \mathbf{R} \mathbf{w}_k - 2\mathbf{P}^T \mathbf{w}_k$$

The gradient is:

$$\frac{d\xi^2}{d\mathbf{w}_k} = 2\mathbf{R}\mathbf{w}_k - 2\mathbf{P}$$

We can update $\mathbf{w}$ with gradient descent:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \frac{d\xi^2}{d\mathbf{w}_k} = \mathbf{w}_k - 2\mu(\mathbf{R}\mathbf{w}_k - \mathbf{P})$$

$$= \mathbf{w}_k - 2\mu(\mathbb{E}\left[\mathbf{x}_k \mathbf{x}_k^T \mathbf{w}_k - d_k \mathbf{x}_k\right]) = \mathbf{w}_k - 2\mu(\mathbb{E}\left[(\mathbf{w}_k^T \mathbf{x}_k - d_k)\mathbf{x}_k\right]) = \mathbf{w}_k + \mu\mathbb{E}\left[e_k \mathbf{x}_k\right]$$

where μ is the gradient descent step size. Replacing the expectation with the instantaneous value gives the LMS update rule:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu e_k \mathbf{x}_k$$

In RLS, unlike in LMS, there is no expectation operator over random variables; instead, it sums over past deterministic data points. RLS maintains an inverse correlation matrix $\mathbf{P}_n$ that captures second-order statistics, leading to faster convergence than LMS, at the cost of more computation. RLS operates in the following steps:

Compute the gain vector:

$$\mathbf{k}_n = \frac{\mathbf{P}_{n-1}\mathbf{x}_n}{\lambda + \mathbf{x}_n^T \mathbf{P}_{n-1} \mathbf{x}_n}$$

Compute the error signal:

$$e_k = d_k - \mathbf{w}_k^T \mathbf{x}_k$$

Update the filter weights:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{k}_k \mathbf{e}_k$$

Update the inverse correlation matrix:

$$\mathbf{P}_k = \frac{1}{\lambda} \left(\mathbf{P}_{k-1} - \mathbf{k}_k \mathbf{x}_k^T \mathbf{P}_{k-1} \right)$$

where λ is the forgetting factor, typically close to 1, which controls how much past data influences the update.

See also: (Ste87)

CHAPTER 3

Method

3.1 Summary of the Method

In standard state-space models (SSMs), the inputs are treated as exogenous forcing terms $\mathbf{u}_j$, driving the system without being inferred. We instead treat the inputs as endogenous state vectors $\mathbf{x}_j$, evolving jointly with state estimates and uncertainty propagation, i.e. we embed the input sequence into a learned autonomous dynamical system rather than assuming an independent driving process.

In dynamical filtering, the goal is not merely to compare states but to relate them through their dynamics. Consider a noisy sequence generated by a dynamical system: $\mathbf{x}_{i+1} = f(\mathbf{x}_i)$, with noisy measurements $\mathbf{z}_i = \mathbf{x}_i + \boldsymbol{\gamma}_i$, where $\boldsymbol{\gamma}_i$ is noise. To assess similarity between noisy measurements $\mathbf{z}_i$ and $\mathbf{z}_j$, we should first evolve $\mathbf{z}_j$ through the dynamics before comparing it to $\mathbf{z}_i$. That is, we seek similarity scores of the form $d\left(\mathbf{z}_i, f^{i-j}(\mathbf{z}_j)\right)$, where $d(\cdot)$ is a distance metric comparing $\mathbf{z}_i$ to the forward- or backward-evolved $\mathbf{z}_j$.

For nonlinear f, this transformation generally cannot be written in closed form. In the linear time-invariant (LTI) case, however, $f(\mathbf{x}) = \mathbf{A}\mathbf{x}$, and the n-step evolution can be written in closed form as $f^n(\mathbf{x}) = \mathbf{A}^n\mathbf{x}$. Computing $\mathbf{A}^n$ directly is expensive for large n, but in the

continuous-time case, where $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t)$, the solution is: $\mathbf{x}(t) = e^{\mathbf{A}(t-t_0)}\mathbf{x}_0$. Assuming $\mathbf{A}$ is diagonalizable as $\mathbf{A} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1}$, the matrix exponential simplifies to $\mathbf{S}e^{\mathbf{\Lambda}(t-t_0)}\mathbf{S}^{-1}$, enabling efficient computation of dynamics transformations. This makes it feasible to incorporate linear dynamic transformations directly into the attention mechanism, aligning similarity computations with system evolution.

Estimating the state of a stochastic system at all measurement times can be framed as a series of parallel maximum-likelihood estimation (MLE) problems, one at each time step. We form point estimates $\hat{\mathbf{z}}_i(t_j)$ for the true state $\mathbf{x}_i$ by mapping the measurements through the dynamics. The solution of the MLE problem is a precision-weighted estimator: a weighted sum of the individual estimates, with each estimate weighted by the inverse covariance (precision) matrix $\mathbf{P}^{-1}(t) = \mathbf{V}(t)$ at the time t at which the corresponding measurement was taken. In LTI systems, $\mathbf{V}(t)$ can be computed by propagating the uncertainty through the dynamics. Under the assumption of a simultaneously diagonalizable state matrix $\mathbf{A}$ and process and measurement noise matrices $\mathbf{\Omega}, \mathbf{\Gamma}$, $\mathbf{V}(t)$ and $\mathbf{P}(t)$ admit closed-form expressions via analytic integration of the differential Lyapunov equation. To account for model mis-specification, we can adaptively re-weight the values of the precision matrix predicted by the LTI model for each time step, using the residuals between pairs of states.

These precision-weighted updates are unnormalized attention scores, using a Mahalanobis similarity function rather than a softmax. The attention mechanism can then be viewed as an adaptive precision-weighted estimator. By using the same matrix exponential to both estimate the current latent states and predict the next states, the model is encouraged to incorporate the underlying system dynamics directly into the computation of the attention weights. The result is a generalization of the attention mechanism, with learned relative time-decay weightings on the attention weights. By imposing a recurrent structure on

the attention weights, we expect this state-space form of attention to exhibit improved generalization across time.

3.2 Model Setup and Definitions

A linear time-invariant (LTI) dynamical system with zero-mean Gaussian white noise can be written, in continuous time, as:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \boldsymbol{\eta}(t),$$

$$\dot{\mathbf{z}}(t) = \mathbf{C}\mathbf{x}(t) + \boldsymbol{\gamma}(t),$$

with initial condition $\mathbf{x}(t_0) := \mathbf{x}_0$, where $\mathbf{x}(t) \in \mathbb{R}^{d_x}$ is referred to as the state, $\mathbf{z}(t) \in \mathbb{R}^{d_z}$ as the measurement, $\mathbf{A} \in \mathbb{R}^{d_x \times d_x}$ as the state transition matrix, and $\mathbf{C} \in \mathbb{R}^{d_z \times d_x}$ as the output matrix. The process and measurement noise are given, respectively, by:

$$\boldsymbol{\eta}(t) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}), \quad \boldsymbol{\gamma}(t) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma}).$$

where $\boldsymbol{\eta}(t) \in \mathbb{R}^{d_x}$ and $\boldsymbol{\gamma}(t) \in \mathbb{R}^{d_z}$, and where the process noise covariance matrix $\boldsymbol{\Omega}$ and the measurement noise covariance matrix $\boldsymbol{\Gamma}$ are symmetric, positive definite matrices.

We can write the state $\mathbf{x}$ at time t_j in terms of the state at an earlier time t_i as:

$$\mathbf{x}(t_j) = e^{\mathbf{A}(t_j - t_i)}\mathbf{x}(t_i) + \int_{t_i}^{t_j} e^{\mathbf{A}(t_j - \tau)}\boldsymbol{\eta}(\tau)d\tau.$$

Since, by assumption, $\boldsymbol{\eta}(t)$ is stationary, we can write this, using a change of variables, as:

$$\mathbf{x}(t_j) = e^{\mathbf{A}\Delta t_{ij}}\left(\mathbf{x}(t_i) + \int_0^{\Delta t_{ij}} e^{-\mathbf{A}\tau}\boldsymbol{\eta}(\tau)d\tau\right), \quad \text{where} \quad \Delta t_{ij} := t_j - t_i.$$

The measurement at time t_j is:

$$\mathbf{z}(t_j) = \mathbf{C}e^{\mathbf{A}\Delta t_{ij}}\left(\mathbf{x}(t_i) + \int_0^{\Delta t_{ij}} e^{-\mathbf{A}\tau}\boldsymbol{\eta}(\tau)d\tau\right) + \boldsymbol{\gamma}(t_j).$$

If we know $\mathbf{x}(t_j)$, we can form an estimate of $\mathbf{x}(t_i)$ by multiplying by the negative matrix exponential:

$$\hat{\mathbf{x}}_i(t_j) := e^{-\mathbf{A}\Delta t_{ij}}\mathbf{x}(t_j)$$

Likewise, if we measure $\mathbf{z}(t_j)$, and the measurement matrix $\mathbf{C}$ is invertible (so $d_z = d_x$), we can estimate $\mathbf{x}(t_i)$, or equivalently $\mathbf{z}^C(t_i) := \mathbf{C}\mathbf{x}(t_i)$, by computing:

$$\hat{\mathbf{z}}_i(t_j) := \mathbf{C}e^{-\mathbf{A}\Delta t_{ij}}\mathbf{C}^{-1}\mathbf{z}(t_j)$$

3.3 Maximum Likelihood Trajectory Estimation

Our goal is to estimate the true trajectory, that is $\{\mathbf{x}(t_1), ..., \mathbf{x}(t_m)\}$ from m measurements $\{\mathbf{z}(t_1), ..., \mathbf{z}(t_m)\}$ taken at times $\{t_1, ..., t_m\}$. We can do so by solving a separate maximum likelihood estimation (MLE) problem for each $\mathbf{x}_i$.

Let $\mathbf{x}_i = \mathbf{x}(t_i)$, $\hat{\mathbf{x}}_{ij} = \hat{\mathbf{x}}_i(t_j)$, $\mathbf{z}_i = \mathbf{z}(t_i)$, $\hat{\mathbf{z}}_{ij} = \hat{\mathbf{z}}_i(t_j)$, and $\mathbf{V}_{ij} = \mathbf{V}(\Delta t_{ij})$.

The estimated state is distributed as: $\hat{\mathbf{x}}_{ij} \sim \mathcal{N}(\mathbf{x}_i, \mathbf{V}_{ij})$, where:

$$\mathbf{V}_{ij} = e^{\mathbf{A}\Delta t_{ij}}\left(\int_0^{|\Delta t_{ij}|} e^{-\mathbf{A}\tau}\mathbf{\Omega}e^{-\mathbf{A}^T\tau}d\tau\right)e^{\mathbf{A}^T\Delta t_{ij}}$$

is the solution of the differential Lyapunov Equation.

The estimated measurement is distributed as: $\hat{\mathbf{z}}_{ij} \sim \mathcal{N}\left(\mathbf{z}_i^C, \mathbf{V}_{ij}^C\right)$, where the propagated measurement covariance is:

$$\mathbf{V}_{ij}^C = \mathbf{C}\left(\mathbf{V}_{ij} + e^{-\mathbf{A}\Delta t_{ij}}\mathbf{C}^{-1}\mathbf{\Gamma}\mathbf{C}^{-T}e^{-\mathbf{A}^T\Delta t_{ij}}\right)\mathbf{C}^T$$

The MLE solution for $\mathbf{z}_i^C$ is then:

$$\bar{\mathbf{z}}_i := \left(\sum_{j\leq i}\mathbf{P}_{ij}^C\right)^{-1}\sum_{j\leq i}\mathbf{P}_{ij}^C\hat{\mathbf{z}}_{ij}, \quad \text{where} \quad \mathbf{P}_{ij}^C = \mathbf{V}_{ij}^{C-1}.$$

where the summation is over $j \leq i$ if the estimation is causal, and over all j otherwise. Repeating this computation for all t_i produces an estimate of the full trajectory. While this requires significantly more computation than recursive filters such as the Kalman filter, this formulation has the advantage that the values of $\bar{\mathbf{z}}_i$ can be computed in parallel for each i. The full derivation of the MLE can be found in the Appendix.

Having computed $\bar{\mathbf{z}}_i$ for the entire trajectory, we predict the next state by propagating it forward using the matrix exponential: $\mathbf{z}_i^{\text{pred}} = e^{\mathbf{A}(t_{i+1}-t_i)}\bar{\mathbf{z}}_i$. We then train a model with learnable $\mathbf{A}$ by comparing these predictions to the ground truth next states using an MLE loss: $\mathcal{L} = \sum_i ||\mathbf{z}_i^{\text{pred}} - \mathbf{z}_{i+1}||_2^2$.

3.4 Closed-form Precision Matrix Calculation for Diagonalizable Matrices

The estimation procedure above requires computing $\mathbf{P}^C(\Delta t_{ij})$ at every ij pair, which is generally expensive, as it involves integration and matrix inversion. However, if we assume that $\mathbf{A}, \mathbf{C}, \mathbf{\Omega}, \mathbf{\Gamma}$ are simultaneously diagonalizable, that is, we can write them in the form $\mathbf{A} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1}$, $\mathbf{\Omega} = \mathbf{S}\mathbf{\Lambda}_\Omega\mathbf{S}^{-1}$, $\mathbf{\Gamma} = \mathbf{S}\mathbf{\Lambda}_\Gamma\mathbf{S}^{-1}$ where $\mathbf{\Lambda}, \mathbf{\Lambda}_\Omega, \mathbf{\Lambda}_\Gamma$ are diagonal matrices and $\mathbf{S}$ an invertible matrix, the covariance and precision matrices admit closed-form solutions:

$$\mathbf{V}_F(\Delta t_{ij}) = \mathbf{S}\mathbf{\Lambda}_\mathbf{V}(|\Delta t_{ij}|)\mathbf{S}^*, \quad \text{where} \quad \mathbf{\Lambda}_\mathbf{V}(|\Delta t_{ij}|) = \mathbf{\Lambda}_\Omega \frac{\mathbf{I} - e^{2\text{Re}(\mathbf{\Lambda})|\Delta t_{ij}|}}{-2\text{Re}(\mathbf{\Lambda})}$$

And:

$$\mathbf{V}^C(\Delta t_{ij}) = \mathbf{S}\mathbf{\Lambda}_\mathbf{V}^C(\Delta t_{ij})\mathbf{S}^*, \quad \text{where} \quad \mathbf{\Lambda}_\mathbf{V}^C(\Delta t_{ij}) = \mathbf{\Lambda}_C^2\mathbf{\Lambda}_\mathbf{V}(\Delta t_{ij}) + e^{-2\text{Re}(\mathbf{\Lambda})\Delta t_{ij}}\mathbf{\Lambda}_\Gamma$$

Finally, $\mathbf{P}_C(\Delta t_{ij}) = \mathbf{S}^{-*}\mathbf{P}_C^\mathbf{S}(\Delta t_{ij})\mathbf{S}^{-1}$, where $\mathbf{\Lambda}_\mathbf{P}^C(\Delta t_{ij}) = \mathbf{\Lambda}_\mathbf{V}^C(\Delta t_{ij})^{-1}$.

The MLE is now tractable:

$$\bar{\mathbf{z}}_i = \mathbf{S}\left(\sum_{j\leq i} \mathbf{\Lambda}^C_{\mathbf{P}ji}\right)^{-1} \sum_{j\leq i} \mathbf{\Lambda}^C_{\mathbf{P}ji}\mathbf{S}^{-1}\hat{\mathbf{z}}_{ij}.$$

We can also extend this result to account for the possibility of continuous-time measurements (see Appendix).

3.5 Adaptive Maximum Likelihood Estimation

Our Maximum Likelihood Estimation (MLE) of a system's trajectory relies on accurate precision matrices, which represent the uncertainty of our estimates. However, if our dynamic model defined by $\{\mathbf{S}, \mathbf{S}^{-1}, \mathbf{\Lambda}, \mathbf{\Lambda}_C, \mathbf{\Lambda}_\Omega, \mathbf{\Lambda}_\Gamma\}$ doesn't perfectly match reality, these theoretical precision matrices can be misleading, leading to suboptimal state estimates.

One way to compensate for model mis-specification is to empirically adjust the precision matrix $\mathbf{P}_C(\Delta t_{ij})$ based on observed residuals, i.e. the difference between the observed and predicted states:

$$\mathbf{r}_{\mathbf{z}i}(t_j) := \hat{\mathbf{z}}_i(t_j) - \mathbf{z}(t_i)$$

If the residuals are approximately zero-mean and uncorrelated with the model error, then their magnitude encodes information about under- or over-confidence in $\mathbf{V}_C(t)$. Adjusting the precision matrix using the measured residuals is a form of adaptive filtering.

By Bayes' rule, the posterior distribution of $\mathbf{V}$ given $\mathbf{r}$ is:

$$p(\mathbf{V}|\mathbf{r}) \propto p(\mathbf{r}|\mathbf{V})p(\mathbf{V}).$$

Letting $\mathbf{V}^{C'}$ and $\mathbf{P}^{C'}$ be our priors for $\mathbf{V}^C$ and $\mathbf{P}^C$, respectively, the prior for the residual distribution under our model is:

$$p\left(\mathbf{r_z}|\mathbf{V}^{C'}\right) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{V}^{C'} + \mathbf{\Gamma}\right)$$

Using a maximum-entropy prior for $p(\mathbf{V})$, and assuming $\mathbb{E}[\mathbf{\Gamma}|\mathbf{r_z}] = \mathbf{\Gamma}$, that is, none of our model mis-specification is due to $\mathbf{\Gamma}$ (which is a conservative assumption in that it avoids underestimating total uncertainty), the posterior mean for our covariance matrix is:

$$\tilde{\mathbf{V}}^C := \mathbb{E}[\mathbf{V}^C|\mathbf{r_z}] = \mathbf{V}^{C'} + \mathbf{r}_z\mathbf{r}_z^T$$

We can then form a point estimate of the precision matrix, using the Sherman-Morrison formula for a rank-1 update of the inverse:

$$\tilde{\mathbf{P}}^C := \mathbb{E}[\mathbf{P}^C|\mathbf{r_z}] \approx \left(\mathbb{E}[\mathbf{V}^C|\mathbf{r_z}]\right)^{-1} = \mathbf{P}^{C'} - \frac{\mathbf{P}^{C'}\mathbf{r_z}\mathbf{r_z}^T\mathbf{P}^{C'}}{1 + \mathbf{r_z}^T\mathbf{P}^{C'}\mathbf{r_z}}.$$

Although $\mathbb{E}[\mathbf{P}^C|\mathbf{r_z}] \neq \left(\mathbb{E}[\mathbf{V}^C|\mathbf{r_z}]\right)^{-1}$ in general, we adopt this approximation as a practical point estimate of the posterior precision, which is common in adaptive filtering.

We can make this slightly more general using an RLS-like update with an exponentially-decaying prior:

$$\tilde{\mathbf{P}}^C \approx \frac{1}{\lambda}\left[\mathbf{P}^{C'} - \frac{\mathbf{P}^{C'}\mathbf{r_z}\mathbf{r_z}^T\mathbf{P}^{C'}}{\lambda + \mathbf{r_z}^T\mathbf{P}^{C'}\mathbf{r_z}}\right],$$

where $\lambda \in (0, 1]$ is a forgetting factor.

Plugging in the rank-1 Bayesian update of the precision matrices, our MLE becomes:

$$\bar{\mathbf{z}}_i = \left(\sum_j \tilde{\mathbf{P}}_{ij}^C\right)^{-1} \sum_j \tilde{\mathbf{P}}_{ij}^C\hat{\mathbf{z}}_{ij} = \mathbf{z}_i + \left(\sum_j \tilde{\mathbf{P}}_{ij}^C\right)^{-1} \sum_j \tilde{\mathbf{P}}_{ij}^C\mathbf{r}_{\mathbf{z}ij}$$

Letting $\mathbf{z}_{\mathbf{s}i} := \mathbf{S}^{-1}\mathbf{z}_i$ and $\hat{\mathbf{z}}_{\mathbf{s}ij} = \mathbf{S}^{-1}\hat{\mathbf{z}}_{ij}$, and: $\mathbf{r}_{\mathbf{zs}ij} := \hat{\mathbf{z}}_{\mathbf{s}ij} - \mathbf{z}_{\mathbf{s}i}$, the MLE becomes:

$$\bar{\mathbf{z}}_{\mathbf{s}i} = \mathbf{z}_{\mathbf{s}i} + \left(\sum_j \left(\mathbf{I} + \mathbf{\Lambda}_{\mathbf{R}ij}\right) \frac{\mathbf{\Lambda}_{\mathbf{P}ij}^C}{1 + \alpha\mathbf{r}_{\mathbf{zs}ij}^*\mathbf{\Lambda}_{\mathbf{P}ij}^C\mathbf{r}_{\mathbf{zs}ij}}\right)^{-1} \sum_j \frac{\mathbf{\Lambda}_{\mathbf{P}ij}^C}{1 + \alpha\mathbf{r}_{\mathbf{zs}ij}^*\mathbf{\Lambda}_{\mathbf{P}ij}^C\mathbf{r}_{\mathbf{zs}ij}}\mathbf{r}_{\mathbf{zs}ij}.$$

where $\mathbf{\Lambda}_{\mathbf{R}ij} = \mathbf{r}^*_{\mathbf{zs}ij}\mathbf{\Lambda}^C_{\mathbf{P}}\mathbf{r}_{\mathbf{zs}ij}\mathbf{I} - \mathbf{\Lambda}^C_{\mathbf{P}}\mathbf{r}_{\mathbf{zs}ij}\mathbf{r}^*_{\mathbf{zs}ij}$, and $\alpha := 1/\lambda \in [1, \infty)$.

Unfortunately, since $\mathbf{\Lambda}_{\mathbf{R}ij}$ is not diagonal, the inverse is now intractable. However, since the residuals are zero-mean, the off-diagonal terms in the sum of the precision matrices approximately average out in the system's eigenbasis, that is $\mathbb{E}\left[\mathbf{\Lambda}_{\mathbf{R}ij}\right] \approx \alpha_j \mathbf{I}$.

If we could approximate $\mathbf{\Lambda}_{\mathbf{R}ij}$ with a fixed diagonal matrix $\bar{\mathbf{\Lambda}}_{\mathbf{R}}$, then letting $\mathbf{\Delta}_{\mathbf{R}} = \left(\mathbf{I} + \bar{\mathbf{\Lambda}}_{\mathbf{R}}\right)^{-1}$, we have:

$$\bar{\mathbf{z}}_{\mathbf{s}i} \approx (\mathbf{I} - \mathbf{\Delta}_{\mathbf{R}})\mathbf{z}_{\mathbf{s}i} + \mathbf{\Delta}_{\mathbf{R}}\hat{\mathbf{z}}^{\text{avg}}_{\mathbf{s}i}.$$

where

$$\begin{aligned}\hat{\mathbf{z}}^{\text{avg}}_{\mathbf{s}i} &:= \left(\sum_j \frac{\mathbf{\Lambda}^C_{Pij}}{1 + \alpha \mathbf{r}^*_{\mathbf{s}ij}\mathbf{\Lambda}^C_{Pij}\mathbf{r}_{\mathbf{s}ij}}\right)^{-1} \sum_j \frac{\mathbf{\Lambda}^C_{Pij}}{1 + \alpha \mathbf{r}^*_{\mathbf{s}ij}\mathbf{\Lambda}^C_{Pij}\mathbf{r}_{\mathbf{s}ij}}\hat{\mathbf{z}}_{\mathbf{s}ij} \\ &= \left(\sum_j \frac{\boldsymbol{\lambda}^C_{Pij}}{1 + \alpha(\boldsymbol{\lambda}^C_{Pij})^T|\mathbf{r}_{\mathbf{s}ij}|^2}\right)^{-1} \odot \left[\sum_j \frac{\boldsymbol{\lambda}^C_{Pij}}{1 + \alpha(\boldsymbol{\lambda}^C_{Pij})^T|\mathbf{r}_{\mathbf{s}ij}|^2} \odot \hat{\mathbf{z}}_{\mathbf{s}ij}\right]\end{aligned}$$

where $\boldsymbol{\lambda}^C_{Pij}$ is the diagonal of $\mathbf{\Lambda}^C_{Pij}$.

Since each element of $\mathbf{\Delta}_{\mathbf{R}}$ is between 0 and 1, this is an affine combination of the current estimate and the weighted average, where the step size is determined by the uncertainty captured in $\mathbf{\Delta}_{\mathbf{R}}$.

Here, $\hat{\mathbf{z}}^{\text{avg}}_{\mathbf{s}i}$ can be interpreted as an M-estimator with influence function $\psi_{\mathbf{\Lambda}}(\mathbf{r}) = \frac{1}{1+\mathbf{r}^T\mathbf{\Lambda}\mathbf{r}}$, where $d = \mathbf{r}^T\mathbf{\Lambda}^C_{\mathbf{P}ij}\mathbf{r}$ is the Mahalanobis distance. The weighting function applies a data-dependent robust weighting to each individual estimate, down-weighting contributions from observations that yield large Mahalanobis distances (i.e. large residuals relative to their estimated uncertainty). This is a feature of M-estimators, which generalize maximum likelihood estimators to provide robustness against outliers or model mis-specification by

reducing the influence of observations that deviate significantly from the model's predictions.

The above motivates an iterative update rule for the state estimate in the eigenbasis:

$$\hat{\mathbf{z}}_{\mathbf{s}ii}^{(k+1)} = (\mathbf{I} - \boldsymbol{\Delta}_{\mathbf{R}})\hat{\mathbf{z}}_{\mathbf{s}ii}^{(k)} + \boldsymbol{\Delta}_{\mathbf{R}}\hat{\mathbf{z}}_{\mathbf{s}i}^{\text{avg}(k)},$$

$$\hat{\mathbf{z}}_{\mathbf{s}i}^{\text{avg}(k)} = \left(\sum_j w_{ij}^{(k)} \boldsymbol{\lambda}_{Pij}^{C}\right)^{-1} \odot \sum_j w_{ij}^{(k)} \boldsymbol{\lambda}_{Pij}^{C} \odot \hat{\mathbf{z}}_{\mathbf{s}ij}^{(k)},$$

$$w_{ij}^{(k)} = \frac{1}{1 + \alpha(\boldsymbol{\lambda}_{Pij}^{C})^T |\mathbf{r}_{\mathbf{s}ij}^{(k)}|^2}, \quad \text{where} \quad \mathbf{r}_{\mathbf{s}ij}^{(k)} := \hat{\mathbf{z}}_{\mathbf{s}ij}^{(k)} - \hat{\mathbf{z}}_{\mathbf{s}ii}^{(k)}.$$

and initial condition $\hat{\mathbf{z}}_{\mathbf{s}ii}^{(1)} = \mathbf{z}_{\mathbf{s}i}$.

This update resembles an Iteratively Reweighted Least Squares (IRLS) step, where each observation is reweighted by a robust factor: $w_j^{(k)}$ suppressing the influence of outliers via the Mahalanobis distance.

For each iteration k, the update for all states $i \in \{1, \ldots, m\}$ is computed simultaneously based on the estimates from iteration k. We can allow the step size $\boldsymbol{\Delta}_{\mathbf{R}}$ to be learned or fix it using the expressions above. This iteration can be implemented with a multi-layer application of the filter with shared parameters $\mathbf{S}, \boldsymbol{\Lambda}, \boldsymbol{\Lambda}_\Omega, \boldsymbol{\Lambda}_\Gamma$ across layers. If $\hat{\mathbf{z}}_{ii}^{(k)} \to \mathbf{z}^C$ in expectation with increasing k, we can take $\bar{\mathbf{z}}_{\mathbf{s}} \approx \hat{\mathbf{z}}_{\mathbf{s}ii}^{(k)}$ at the last layer, and then predict the measurement at the next time step in the eigenbasis using the (diagonalized) matrix exponential:

$$\mathbf{z}_{\mathbf{s}i}^{\text{pred}} = \boldsymbol{\Lambda}_C e^{\boldsymbol{\Lambda}(t_{i+1} - t_i)} \boldsymbol{\Lambda}_C^{-1} \bar{\mathbf{z}}_{\mathbf{s}i} = e^{\boldsymbol{\Lambda}(t_{i+1} - t_i)} \bar{\mathbf{z}}_{\mathbf{s}i}$$

And finally, compute the output prediction: $\mathbf{z}_i^{\text{pred}} = \mathbf{S}\mathbf{z}_{\mathbf{s}i}^{\text{pred}}$,
which we then compare to the actual measurement at the next time step $(t_i + 1)$: $\mathbf{z}_{i+1}$. We can then define a loss $\mathcal{L} = \sum_i ||\mathbf{z}_i^{\text{pred}} - \mathbf{z}_{i+1}||_2^2$ and train our model parameters $\mathbf{S}, \boldsymbol{\Lambda}, \boldsymbol{\Lambda}_\Omega, \boldsymbol{\Lambda}_\Gamma$

using backpropagation.

A sequence of layers with shared weights and a final prediction step can therefore act as an adaptive filter module. Unlike most classical adaptive filters, which update parameters sequentially over time, attention mechanisms compute fast weights — data-dependent weights that adapt dynamically to the input on each forward pass. Although the underlying model parameters are frozen at inference, these data-dependent computations allow the network to implicitly act as an adaptive filter.

In designing a multilayer architecture for dynamic state estimation and prediction, there are two primary choices: either perform a single estimation step followed immediately by the prediction step, or use multiple estimation layers with shared dynamic parameters that iteratively refine the state estimate before applying a single prediction at the end. More layers per module improves consistency between the learned dynamic models used for estimation and prediction by ensuring the final latent state is close to the true MLE, which should improve the accuracy of the prediction, but at the expense of more computation. Alternatively, if distinct parameters are learned per layer, each layer must compute an estimate as close to the MLE as possible, which means $\mathbf{\Delta}_R$ will have to be chosen more aggressively. The number of layers to put in a module depends on how fast the latent state converges toward the MLE and should be determined empirically. Multiple adaptive filter modules with different learned parameters can then be stacked on top of each other.

3.6 Adaptive Filter Attention

We now generalize the structured estimator above using learned query, key, value, and output matrices: $\mathbf{W}_Q \in \mathbb{C}^{d_k \times d_e}$, $\mathbf{W}_K \in \mathbb{C}^{d_k \times d_e}$, $\mathbf{W}_V \in \mathbb{C}^{d_v \times d_e}$, $\mathbf{W}_P \in \mathbb{C}^{d_e \times d_v}$.

In this formulation, $\mathbf{W}_V$ replaces $\mathbf{S}^{-1}$ and $\mathbf{W}_P$ replaces $\mathbf{S}$. Rather than enforcing inverse constraints between them, we allow the network to learn approximate inverses as needed. The final structure enables joint estimation and prediction under shared dynamics.

Define key and value model parameters: $\boldsymbol{\lambda}_{\mathbf{k}}, \boldsymbol{\lambda}_{\Omega\mathbf{k}}, \boldsymbol{\lambda}_{\Gamma\mathbf{k}} \in \mathbb{R}^{d_k}$, $\boldsymbol{\lambda}_{\mathbf{v}}, \boldsymbol{\lambda}_{\Omega\mathbf{v}}, \boldsymbol{\lambda}_{\Gamma\mathbf{v}} \in \mathbb{R}^{d_v}$
If we set $d_k = d_v$, then we can use a shared set of parameters for the key and value: $\boldsymbol{\lambda}, \boldsymbol{\lambda}_\Omega, \boldsymbol{\lambda}_\Gamma \in \mathbb{R}^{d_v}$.

The inputs to the network are: $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_m] \in \mathbb{C}^{d_e \times m}$

Compute query, key, and value vectors:

$$\mathbf{Z}_\mathbf{q} = \mathbf{W}_Q \mathbf{Z} \in \mathbb{C}^{d_k \times m}, \quad \mathbf{Z}_\mathbf{k} = \mathbf{W}_K \mathbf{Z} \in \mathbb{C}^{d_k \times m}, \quad \mathbf{Z}_\mathbf{v} = \mathbf{W}_V \mathbf{Z} \in \mathbb{C}^{d_v \times m}$$

Compute exponential tensors: $\mathcal{E}_\mathbf{k} \in \mathbb{C}^{d_k \times m \times m}$, $\mathcal{E}_\mathbf{v} \in \mathbb{C}^{d_v \times m \times m}$:

$$\mathcal{E}_\mathbf{k}[k, i, j] = e^{\lambda_{\mathbf{k}k}(t_i - t_j)}, \quad \mathcal{E}_\mathbf{v}[k, i, j] = e^{\lambda_{\mathbf{v}k}(t_i - t_j)}$$

Compute the point estimates $\hat{\mathcal{Z}}_\mathbf{k} \in \mathbb{C}^{d_k \times m \times m}$, $\hat{\mathcal{Z}}_\mathbf{v} \in \mathbb{C}^{d_v \times m \times m}$:

$$\hat{\mathcal{Z}}_\mathbf{k}[k, i, j] = \mathcal{E}_\mathbf{k}[k, i, j] \cdot \mathbf{Z}_\mathbf{k}[k, j], \quad \hat{\mathcal{Z}}_\mathbf{v}[k, i, j] = \mathcal{E}_\mathbf{v}[k, i, j] \cdot \mathbf{Z}_\mathbf{v}[k, j],$$

Compute the residuals $\mathcal{R} \in \mathbb{C}^{d_k \times m \times m}$:

$$\mathcal{R}_{\mathbf{qk}}[k, i, j] = \hat{\mathcal{Z}}_\mathbf{k}[k, i, j] - \mathbf{Z}_\mathbf{q}[k, i]$$

Compute the norm of the residuals:

$$|\mathcal{R}_{\mathbf{qk}}[k,i,j]|^2 = Re^2\left(\mathcal{R}_{\mathbf{qk}}[k,i,j]\right) + Im^2\left(\mathcal{R}_{\mathbf{qk}}[k,i,j]\right)$$

Compute the key and value precision tensors $\mathcal{P}^C_{\mathbf{k}} \in \mathbb{R}^{d_k \times m \times m}$, $\mathcal{P}^C_{\mathbf{v}} \in \mathbb{R}^{d_v \times m \times m}$:

$$\mathcal{P}^C_{\mathbf{k}}[:,i,j] = \left(\boldsymbol{\lambda}^2_{C\mathbf{k}} \odot \boldsymbol{\lambda}_{\Omega\mathbf{k}} \odot \frac{\mathbf{1} - e^{2\mathrm{Re}(\boldsymbol{\lambda}_{\mathbf{k}})|t_j - t_i|}}{-2\mathrm{Re}(\boldsymbol{\lambda}_{\mathbf{k}})} + \boldsymbol{\lambda}_{\Gamma\mathbf{k}} \odot e^{2\mathrm{Re}(\boldsymbol{\lambda}_{\mathbf{k}})|t_j - t_i|}\right)^{-1}$$

$$\mathcal{P}^C_{\mathbf{v}}[:,i,j] = \left(\boldsymbol{\lambda}^2_{C\mathbf{v}} \odot \boldsymbol{\lambda}_{\Omega\mathbf{v}} \odot \frac{\mathbf{1} - e^{2\mathrm{Re}(\boldsymbol{\lambda}_{\mathbf{v}})|t_j - t_i|}}{-2\mathrm{Re}(\boldsymbol{\lambda}_{\mathbf{v}})} + \boldsymbol{\lambda}_{\Gamma\mathbf{v}} \odot e^{2\mathrm{Re}(\boldsymbol{\lambda}_{\mathbf{v}})|t_j - t_i|}\right)^{-1}$$

Apply a causal attention mask to $\mathcal{P}^C_{\mathbf{k}}, \mathcal{P}^C_{\mathbf{v}}$, masking out all $j > i$.

Compute the weights $W[i,j] \in \mathbb{R}^{m \times m}$:

$$W[i,j] = \left(1 + \sum_k \mathcal{P}^C_{\mathbf{k}}[k,i,j] \cdot |\mathcal{R}_{\mathbf{qk}}[k,i,j]|^2\right)^{-1}$$

(So the value parameters $\mathbf{W}_V, \boldsymbol{\lambda}_{\mathbf{v}}, \boldsymbol{\lambda}_{\Omega\mathbf{v}}, \boldsymbol{\lambda}_{\Gamma\mathbf{v}}$ are responsible for the prior, while the query and key parameters $\mathbf{W}_Q, \mathbf{W}_K, \boldsymbol{\lambda}_{\mathbf{k}}, \boldsymbol{\lambda}_{\Omega\mathbf{k}}, \boldsymbol{\lambda}_{\Gamma\mathbf{k}}$ are responsible for the adaptive reweighting. The multiplier α in front of the sum can be absorbed into $\mathbf{W}_Q$ and $\mathbf{W}_K$.)

Compute the unnormalized attention scores $\tilde{\mathcal{Q}} \in \mathbb{R}^{d_v \times m \times m}$:

$$\tilde{\mathcal{Q}}[k,i,j] = W[i,j] \cdot \mathcal{P}^C_{\mathbf{v}}[k,i,j]$$

Normalize to obtain the attention tensor $\mathcal{Q} \in \mathbb{R}^{d_v \times m \times m}$:

$$\mathcal{Q}[k,i,j] = \tilde{\mathcal{Q}}[k,i,j] \Big/ \sum_{j' \le i} \tilde{\mathcal{Q}}[k,i,j']$$

Compute the estimate $\bar{\mathbf{Z}} \in \mathbb{C}^{d_v \times m}$:

$$\bar{\mathbf{Z}}[k,i] = \sum_{j \le i} \mathcal{Q}[k,i,j] \cdot \hat{\mathcal{Z}}_{\mathbf{v}}[k,i,j]$$

Optionally, add a residual connection:

$$\bar{\mathbf{Z}}[k,i] \leftarrow (\mathbf{1} - \boldsymbol{\Delta}_R) \odot Z_{\mathbf{v}}[k,i] + \boldsymbol{\Delta}_R \odot \bar{\mathbf{Z}}[k,i], \quad \boldsymbol{\Delta}_R \in (0,1]^{d_v}$$

After n layers with tied weights, compute the prediction $\mathbf{Z}^{\text{pred}} \in \mathbb{C}^{d_e \times m}$ using a one-step forward transition:

$$\mathbf{Z}_{\mathbf{v}}^{\text{pred}}[k,i] = \mathcal{E}_{\mathbf{v}}^{\text{pred}}[k,i+1,i] \cdot \bar{\mathbf{Z}}[k,i]$$

$$\mathbf{Z}^{\text{pred}}[k,i] = \mathbf{W}_P \mathbf{Z}_{\mathbf{v}}^{\text{pred}}[k,i]$$

This structure generalizes attention by introducing a more structured similarity measure that integrates uncertainty and time decay. If $\boldsymbol{\lambda}_{\mathbf{v}}, \boldsymbol{\lambda}_{\mathbf{k}} \to 0$, the relative temporal decay on the attention weights disappears and we are left with ordinary attention.

3.7 Speeding up the Computation

Vanilla multi-head attention with input length m and embedding dimension d has computational complexity $\mathcal{O}(m^2 d)$, with storage of the query, key, and value matrices requiring $\mathcal{O}(md)$ memory, and the attention matrix requiring $\mathcal{O}(m^2)$. In contrast, Adaptive Filter Attention (AFA) involves element-wise multiplication of $d_v \times m \times m$ tensors, resulting in time complexity $\mathcal{O}(m^2 d)$, but slower runtime in practice due to less-optimized element-wise operations. The main bottleneck, however, is not run-time, but memory, as we need to store tensors of size $\mathcal{O}(m^2 d)$.

Convolutional Representation:
If the time steps are equal, i.e. $\Delta t = t_{i+1} - t_i$ is fixed for all i, we can store all ij exponentials in a kernel of size $(2m-1)d$, in the causal case md, rather than $m^2 d$. In particular, we

compute: $\mathcal{K}_{\mathbf{v}}[\tau] = e^{\boldsymbol{\lambda}_{\mathbf{v}}\Delta t\tau}$ and $\mathcal{K}_{\mathbf{k}}[\tau] = e^{\boldsymbol{\lambda}_{\mathbf{k}}\Delta t\tau}$, for $\tau \in [m-1, -(m-1)]$:

$$\mathcal{K}_{\mathbf{v}} = \left[e^{\boldsymbol{\lambda}_{\mathbf{v}}(m-1)\Delta t}, e^{\boldsymbol{\lambda}_{\mathbf{v}}(m-2)\Delta t}, \ldots, e^{\boldsymbol{\lambda}_{\mathbf{v}}\Delta t}, e^{\mathbf{0}}, e^{-\boldsymbol{\lambda}_{\mathbf{v}}\Delta t}, \ldots, e^{-\boldsymbol{\lambda}_{\mathbf{v}}(m-1)\Delta t}, e^{-\boldsymbol{\lambda}_{\mathbf{v}}m\Delta t}\right],$$

or in the causal case, $\tau \in [m-1, 0]$:

$$\mathcal{K}_{\mathbf{v}} = \left[e^{\boldsymbol{\lambda}_{\mathbf{v}}(m-1)\Delta t}, e^{\boldsymbol{\lambda}_{\mathbf{v}}(m-2)\Delta t}, \ldots, e^{\boldsymbol{\lambda}_{\mathbf{v}}\Delta t}, 1, \ldots, 1\right],$$

(where we pad the kernel with m 1s for easier indexing.) We define $\mathcal{K}_{\mathbf{k}}$ similarly. We can then efficiently compute the exponentials using a Toeplitz matrix to index from $\mathcal{K}_{\mathbf{v}}$ and $\mathcal{K}_{\mathbf{k}}$. In particular, $e^{\boldsymbol{\lambda}_{\mathbf{v}k}(t_i - t_j)} = \mathcal{K}_{\mathbf{v}}[m-(i-j)]$. The point estimate tensors are then: $\hat{\mathcal{Z}}_{\mathbf{v}}[k,i,j] = \mathcal{K}_{\mathbf{v}}[m-(i-j)] \cdot \mathbf{Z}_{\mathbf{v}}[k,j]$ and $\hat{\mathcal{Z}}_{\mathbf{k}}[k,i,j] = \mathcal{K}_{\mathbf{k}}[m-(i-j)] \cdot \mathbf{Z}_{\mathbf{k}}[k,j]$.

Similarly, the precision tensor is Toeplitz, so we can again represent it with a kernel of size md:

$$\mathcal{K}^P_{\mathbf{k}}[\tau] = \left(\boldsymbol{\lambda}^2_{C\mathbf{k}} \odot \boldsymbol{\lambda}_{\Omega\mathbf{k}} \odot \frac{\mathbf{1} - e^{2\mathrm{Re}(\boldsymbol{\lambda}_{\mathbf{k}})\Delta t\tau}}{-2\mathrm{Re}(\boldsymbol{\lambda}_{\mathbf{k}})} + \boldsymbol{\lambda}_{\Gamma\mathbf{k}} \odot e^{2\mathrm{Re}(\boldsymbol{\lambda}_{\mathbf{k}})\Delta t\tau}\right)^{-1}$$

which we can express in terms of $\mathcal{K}_{\mathbf{k}}[\tau]$:

$$\mathcal{K}^P_{\mathbf{k}}[\tau] = \left(\boldsymbol{\lambda}^2_{C\mathbf{k}} \odot \boldsymbol{\lambda}_{\Omega\mathbf{k}} \odot \frac{\mathbf{1} - \mathcal{K}^*_{\mathbf{k}}[\tau]\mathcal{K}_{\mathbf{k}}[\tau]}{-2\mathrm{Re}(\boldsymbol{\lambda}_{\mathbf{k}})} + \boldsymbol{\lambda}_{\Gamma\mathbf{k}} \odot \left(\mathcal{K}^*_{\mathbf{k}}[\tau]\mathcal{K}_{\mathbf{k}}[\tau]\right)\right)^{-1}$$

The full precision tensor is then:

$$\mathcal{P}^C_{\mathbf{k}}[k,i,j] = \mathcal{K}^P_{\mathbf{k}}[|i-j|],$$

and similarly for $\mathcal{K}^P_{\mathbf{v}}[\tau]$ and $\mathcal{P}^C_{\mathbf{v}}$.

However, we must still store the full tensors of point estimates $\hat{\mathcal{Z}}_{\mathbf{k}}[k,i,j]$ and $\hat{\mathcal{Z}}_{\mathbf{v}}[k,i,j]$, which are of size $d_k \times m \times m$ and $d_v \times m \times m$, respectively.

Broadcasting for Point Estimates:

Alternatively, to avoid explicitly constructing $\hat{\mathcal{Z}}_{\mathbf{k}}[k,i,j]$ and $\hat{\mathcal{Z}}_{\mathbf{v}}[k,i,j]$, we can define separate arrays for the i and j components:

$$\mathbf{V}_{\mathbf{v}}[k,i] := e^{\boldsymbol{\lambda}_{\mathbf{v}k}t_i}, \quad \mathbf{U}_{\mathbf{v}}[k,j] := e^{-\boldsymbol{\lambda}_{\mathbf{v}k}t_j}\mathbf{Z}_{\mathbf{v}}[k,j].$$

We can then factorize $\hat{\mathcal{Z}}_{\mathbf{v}}[k,i,j]$:

$$\hat{\mathcal{Z}}_{\mathbf{v}}[k,i,j] = \mathbf{V}_{\mathbf{v}}[k,i] \cdot \mathbf{U}_{\mathbf{v}}[k,j],$$

And likewise for $\hat{\mathcal{Z}}_{\mathbf{k}}[k,i,j]$. This requires $2md$ computations rather than md, but reduces the memory requirement from m^2d to $2md$.

Since $\mathbf{V}_{\mathbf{v}}[k,i]$ does not depend on j, we can pull it outside the sum in the weighted sum for the estimation:

$$\bar{\mathbf{Z}}_v[k,i] := \sum_{j \le i} \mathcal{Q}[k,i,j] \cdot \hat{\mathbf{Z}}_{\mathbf{v}}[k,i,j] = \mathbf{V}_{\mathbf{v}}[k,i] \cdot \sum_{j \le i} \mathcal{Q}[k,i,j] \cdot \mathbf{U}_{\mathbf{v}}[k,j]$$

So we can avoid forming the full $d_v \times m \times m$ tensor $\hat{\mathcal{Z}}_{\mathbf{v}}[k,i,j]$.

Dealing with the Residuals:

Recall that:

$$\mathcal{R}_{\mathbf{qk}}[k,i,j] = \hat{\mathcal{Z}}_{\mathbf{k}}[k,i,j] - \mathbf{Z}_{\mathbf{q}}[k,i]$$

Letting:

$$\mathbf{V}_{\mathbf{k}}[k,i] = e^{\boldsymbol{\lambda}_{\mathbf{k}}t_i}, \quad \mathbf{U}_{\mathbf{k}}[k,j] = e^{-\boldsymbol{\lambda}_{\mathbf{k}}t_j}\mathbf{Z}_{\mathbf{k}}[k,j], \quad \mathbf{U}_{\mathbf{q}}[k,i] = e^{-\boldsymbol{\lambda}_{\mathbf{k}}t_i}\mathbf{Z}_{\mathbf{q}}[k,i],$$

$$\mathcal{R}_{\mathbf{qk}}[k,i,j] = \mathbf{V}_{\mathbf{k}}[k,i] \cdot (\mathbf{U}_{\mathbf{k}}[k,j] - \mathbf{U}_{\mathbf{q}}[k,i]) = \mathbf{V}_{\mathbf{k}}[k,i]\mathcal{R}_{\mathbf{u}}[k,i,j],$$

where:

$$\mathcal{R}_{\mathbf{u}}[k,i,j] := \mathbf{U}_{\mathbf{k}}[k,j] - \mathbf{U}_{\mathbf{q}}[k,i].$$

We need to compute:

$$d_{ij} := \sum_k \mathcal{R}^*_{\mathbf{qk}}[k,i,j] \cdot \mathcal{P}^C_{\mathbf{k}}[k,i,j] \cdot \mathcal{R}_{\mathbf{qk}}[k,i,j]$$

$$= \sum_k |\mathbf{V_k}[k,i]|^2 \mathcal{R}^*_{\mathbf{u}}[k,i,j] \cdot \mathcal{P}^C_{\mathbf{k}}[k,i,j] \cdot \mathcal{R}_{\mathbf{u}}[k,i,j]$$

$$= \sum_k |\mathbf{V_k}[k,i]|^2 \left(\mathbf{U_k}[k,j] - \mathbf{U_q}[k,i]\right)^* \cdot \mathcal{P}^C_{\mathbf{k}}[k,i,j] \cdot \left(\mathbf{U_k}[k,j] - \mathbf{U_q}[k,i]\right)$$

We can avoid computing $\mathcal{R}_{\mathbf{u}}$ by expanding the product:

$$d_{ij} = \sum_k \left|\mathbf{V_k}[k,i]\right|^2 \Big[\mathcal{P}^C_{\mathbf{k}}[k,i,j]\left|\mathbf{U_k}[k,j]\right|^2 + \mathcal{P}^C_{\mathbf{k}}[k,i,j]\left|\mathbf{U_q}[k,i]\right|^2 - 2\mathrm{Re}\left(\mathbf{U}^*_{\mathbf{q}}[k,i]\mathcal{P}^C_{\mathbf{k}}[k,i,j]\mathbf{U_k}[k,j]\right)\Big]$$

Unfortunately, this still requires constructing the full $d_k \times m \times m$ precision tensor $\mathcal{P}^C_{\mathbf{k}}$. Likewise, the attention tensor $\mathcal{Q}$ is still $d_k \times m \times m$. The only way to avoid this is to collapse the k dimension of $\mathcal{P}^C_{\mathbf{k}}$ and $\mathcal{P}^C_{\mathbf{v}}$, i.e. replace them with their sums: $\mathbf{p}^C_{\mathbf{k}} := \mathbf{M}_{\text{causal}} \odot \sum_k \mathcal{P}^C_{\mathbf{k}}$ and $\mathbf{p}^C_{\mathbf{v}} := \mathbf{M}_{\text{causal}} \odot \sum_k \mathcal{P}^C_{\mathbf{v}}$, where $\mathbf{M}_{\text{causal}} \in \mathbb{R}^{m \times m}$ is a causal mask. We can then pull $p^C_{\mathbf{k}} \in \mathbb{R}^{m \times m}$ outside the sum:

$$d_{ij} = \mathbf{p}^C_{\mathbf{k}}[i,j] \sum_k \left|\mathbf{V_k}[k,i]\right|^2 \Big[\left|\mathbf{U_k}[k,j]\right|^2 + \left|\mathbf{U_q}[k,i]\right|^2 - 2\mathrm{Re}\left(\mathbf{U}^*_{\mathbf{q}}[k,i]\mathbf{U_k}[k,j]\right)\Big]$$

We can now express each of these three sums as matrix products:

$$\sum_k \left|\mathbf{V_k}[k,i]\right|^2 \left|\mathbf{U_k}[k,j]\right|^2 = \left(\left|\mathbf{V_k}\right|^2\right)^T \left|\mathbf{U_k}\right|^2 \in \mathbb{R}^{m \times m},$$

$$\sum_k \left|\mathbf{V_k}[k,i]\right|^2 \left|\mathbf{U_q}[k,i]\right|^2 = \sum_k \left|\mathbf{V_k}\right|^2 \odot \left|\mathbf{U_q}\right|^2 \in \mathbb{R}^m,$$

$$\sum_k \left|\mathbf{V_k}[k,i]\right|^2 \mathbf{U}^*_{\mathbf{q}}[k,i]\mathbf{U_k}[k,j] = \left(\left|\mathbf{V_k}\right|^2 \odot \mathbf{U_q}\right)^* \mathbf{U_k} \in \mathbb{C}^{m \times m}$$

We then sum terms, where the middle term is broadcast from an $m \times 1$ vector to an $m \times m$ matrix:

$$\mathbf{D} := \mathbf{p}^C_{\mathbf{k}} \odot \Big[\left(\left|\mathbf{V_k}\right|^2\right)^T \left|\mathbf{U_k}\right|^2 + \sum_k \left|\mathbf{V_k}\right|^2 \odot \left|\mathbf{U_q}\right|^2 - 2\mathrm{Re}\Big(\left(\left|\mathbf{V_k}\right|^2 \odot \mathbf{U_q}\right)^* \mathbf{U_k}\Big)\Big]$$

We can then compute the unnormalized attention matrix:

$$\tilde{\mathbf{Q}} = \mathbf{p}_{\mathbf{v}}^{C} \odot \left(1 + \mathbf{D}\right)^{-1}$$

and normalize the rows:

$$\mathbf{Q}[i,j] = \tilde{\mathbf{Q}}[i,j] \Big/ \sum_{j} \tilde{\mathbf{Q}}[i,j]$$

The estimate becomes:

$$\bar{\mathbf{Z}}_{v}[k,i] = \mathbf{V}_{\mathbf{v}}[k,i] \cdot \sum_{j} \mathbf{Q}[i,j] \cdot \mathbf{U}_{\mathbf{v}}[k,j]$$

which, in matrix form, is:

$$\bar{\mathbf{Z}}_{\mathbf{v}} = \mathbf{V}_{\mathbf{v}} \odot \left(\mathbf{U}_{\mathbf{v}} \mathbf{Q}^{T}\right)$$

Or, in a form more typical for attention:

$$\bar{\mathbf{Z}}_{\mathbf{v}}^{T} = \mathbf{V}_{\mathbf{v}}^{T} \odot \left(\mathbf{Q} \mathbf{U}_{\mathbf{v}}^{T}\right)$$

Finally, the prediction is:

$$\mathbf{Z}_{\mathbf{v}}^{\text{pred}} = e^{\boldsymbol{\lambda}_{\mathbf{v}} \Delta t} \cdot \bar{\mathbf{Z}}_{\mathbf{v}}$$

This simplified version of AFA reduces the memory requirement from $\mathcal{O}(m^2 d)$ to $\mathcal{O}(m^2 + md)$, as in vanilla attention. This comes at the price of losing the second-order information provided by the full tensors $\mathcal{P}_{\mathbf{k}}^{C}$ and $\mathcal{P}_{\mathbf{v}}^{C}$. Whereas $\mathcal{P}_{\mathbf{k}}^{C}$ and $\mathcal{P}_{\mathbf{v}}^{C}$ represent the confidence of the k-th feature's projection from time t_j to time t_i, $\mathbf{p}_{\mathbf{k}}^{C}$ and $\mathbf{p}_{\mathbf{v}}^{C}$ represent a measurement of overall confidence between query and key. Alternatively, rather than a simple sum, we could use a weighted sum, with learnable positive weights: $\mathbf{p}_{\mathbf{k}}^{C}[i,j] = \sum_k \alpha_{\mathbf{k}k}^2 \mathcal{P}_{\mathbf{k}}^{C}[k,i,j] + \beta_{\mathbf{k}}^2$, $\mathbf{p}_{\mathbf{v}}^{C}[i,j] = \sum_k \alpha_{\mathbf{v}k}^2 \mathcal{P}_{\mathbf{v}}^{C}[k,i,j] + \beta_{\mathbf{v}}^2$.

CHAPTER 4

Implementation and Experiments

4.1 Implementation

An implementation of single-head Adaptive Filter Attention can be found at the following Github link.

The above formulation relies on complex-valued weight matrices. Prior work on complex-valued neural networks suggests that incorporating specialized techniques to handle complex parameters natively may improve training stability. These techniques include complex-valued activation functions, parameter initialization strategies, and the use of Wirtinger calculus, which treats complex parameters as pairs of real variables to enable effective gradient computation. (TBZ18), (BRV23), (Ham24). However, for simplicity, we use real-valued matrices to represent the real and imaginary parts separately, along with the appropriate definitions for complex multiplication and exponentiation. This approach allows us to use standard automatic differentiation functionality without requiring specialized complex-valued gradient methods.

Input: A sequence of input vectors: $\mathbf{z}_1, \ldots, \mathbf{z}_m \in \mathbb{R}_e^d$ at times $t_1, \ldots, t_m$, stacked vertically in $\mathbf{Z}_r \in \mathbb{R}^{d_e \times m}$.

Target: A shifted sequence of vectors: $\mathbf{z}_2, \ldots, \mathbf{z}_{m+1} \in \mathbb{R}_e^d$ at times $t_2, \ldots, t_{m+1}$, stacked vertically in $\mathbf{Z}'_r \in \mathbb{R}^{d_e \times m}$.

Initialization:

Initialize weight matrices: $\mathbf{W}_Q \in \mathbb{R}^{2\times d_k \times d_e}$, $\mathbf{W}_K \in \mathbb{R}^{2\times d_k \times d_e}$, $\mathbf{W}_V \in \mathbb{R}^{2\times d_v \times d_e}$, $\mathbf{W}_P \in \mathbb{R}^{2\times d_e \times d_v}$. Each of $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_P$ is represented as a real-valued tensor, where the first dimension corresponds to the real (index 0) and imaginary (index 1) parts of the complex matrix. Given matrices $\mathbf{A} = \mathbf{A}_r + i\mathbf{A}_i$ and $\mathbf{B} = \mathbf{B}_r + i\mathbf{B}_i$, their produce is: $\mathbf{AB} = (\mathbf{A}_r\mathbf{B}_r - \mathbf{A}_i\mathbf{B}_i) + i\,(\mathbf{A}_r\mathbf{B}_i + \mathbf{A}_i\mathbf{B}_r)$, so we can handle complex-valued matrices using two real-valued matrices representing the real and imaginary parts.

In order to ensure an isotropic distribution, we initialize these as:

$$\mathbf{W}_{ij} = \alpha_{ij} \begin{bmatrix} \cos(\phi_{ij}) \\ \sin(\phi_{ij}) \end{bmatrix}$$

where $\alpha_{ij} \sim \sqrt{\frac{2}{d_{in}+d_{out}}}\mathcal{N}(0,1)$ and $\phi_{ij} \sim \mathcal{U}(0, 2\pi)$.

Initialize parameter arrays for the real and imaginary parts of the eigenvalues of the state transition matrix: $\boldsymbol{\lambda}_r, \boldsymbol{\lambda}_i \in \mathbb{R}^{\frac{d_v}{2}}$. In order to ensure stability, we require that the real component $\boldsymbol{\lambda}_r$ be non-positive, for example using a negative square or negative ReLU. This also guarantees that the propagated covariance is bounded, since $t_j \leq t_i$ for causal systems. We then construct the other $d_v/2$ values as the complex conjugates of the first $d_v/2$.

Alternatively, we can relax the assumption of strictly non-positive eigenvalues by bounding the positive magnitude of $\boldsymbol{\lambda}_r$ scaled by the time interval $T = t_f - t_0$, for example, using:

$\boldsymbol{\lambda}_r = \frac{\lambda_{max}}{2T}\left(1 - ReLU(-\tilde{\boldsymbol{\lambda}}_r)\right)$.

Here, the relevant scale for the real parts is not the embedding dimension, but rather the time-decay constant τ of the dynamics. A reasonable starting value so that the dynamics don't die off too fast is $\tau_i = T$. Since $\tau_i = -1/\mathrm{Re}(\lambda_i)$, the real part of our initial eigenvalues should be near $\mathrm{Re}(\lambda_i) \approx -1/T$. This ensures that the time scale of the dynamics roughly spans the full input window, enabling both slow and fast modes to contribute meaningfully at initialization. However, in practice, we find that it is better to start with larger magnitudes, so we just use a normal distribution: $\tilde{\boldsymbol{\lambda}} \sim \mathcal{N}(0, 1)$.

Initialize parameter arrays for the eigenvalues of the process and measurement noise covariances matrices, and the measurement matrix: $\boldsymbol{\lambda}_\Omega$, $\boldsymbol{\lambda}_\Gamma$, $\boldsymbol{\lambda}_C \in \mathbb{R}^{d_v}$. These must be positive for $\boldsymbol{\Omega}$, $\boldsymbol{\Gamma}$ to be positive definite, and initialized on a similar scale as $\boldsymbol{\lambda}$, so we can draw their square roots from a standard normal distribution: $\tilde{\boldsymbol{\lambda}}_\Omega, \tilde{\boldsymbol{\lambda}}_\Gamma \sim \mathcal{N}(0, 1) \in \mathbb{R}^{d_v}$, and then square these values. Note that $\boldsymbol{\lambda}_C$ can be absorbed into $\boldsymbol{\lambda}_\Omega$ but we leave it here for clarity, initializing it to an array of 1s, i.e. full pass-through.

Prepare the parameter arrays and inputs:

Compute the eigenvalues $\boldsymbol{\lambda}$ of the state transition matrix, using a sigmoid function $\sigma(\cdot)$ to ensure stability:
$\boldsymbol{\lambda}_r = -\frac{\lambda_{max}}{2T}\sigma\left(\tilde{\boldsymbol{\lambda}}_r\right)$, $\boldsymbol{\lambda}_i = \frac{2\pi}{T}\tilde{\boldsymbol{\lambda}}_i \in \mathbb{R}^{\frac{d_v}{2}}$.
Then concatenate the complex-conjugate pairs to form the eigenvalue array:

$$\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda_r}, \boldsymbol{\lambda_i} \\ \boldsymbol{\lambda_r}, -\boldsymbol{\lambda_i} \end{bmatrix} \in \mathbb{R}^{2\times d_v}$$

Square the process and measurement noise covariance eigenvalues: $\boldsymbol{\lambda}_\Omega = \tilde{\boldsymbol{\lambda}}_\Omega^2$,$\boldsymbol{\lambda}_\Gamma = \tilde{\boldsymbol{\lambda}}_\Gamma^2 \in \mathbb{R}^{d_v}$ to ensure non-negativity.

Concatenate an array of zeros to $\mathbf{Z}_r$ and $\mathbf{Z}'_r$ so that they are complex-valued: $\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_r \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{Z}' = \begin{bmatrix} \mathbf{Z}'_r \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2\times d_e \times m}$.

Compute query, key, and value vectors: $\mathbf{Z}_Q = \mathbf{W}_Q\mathbf{Z}$, $\mathbf{Z}_K = \mathbf{W}_K\mathbf{Z}$, $\mathbf{Z}_V = \mathbf{W}_V\mathbf{Z}$. This gives: $\mathbf{Z}_Q \in \mathbb{R}^{d_k\times m}$, $\mathbf{Z}_K \in \mathbb{R}^{d_k\times m}$, $\mathbf{Z}_V \in \mathbb{R}^{d_v\times m}$.

Forward Pass:

Execute a forward pass of AFA, or the simplified version, as described in the previous two sections. In the case of the former, this requires the following:

- Compute query, key, and value vectors $\mathbf{Z}_\mathbf{q}, \mathbf{Z}_\mathbf{k}, \mathbf{Z}_\mathbf{v} \in \mathbb{R}^{2\times d_v\times m}$
- Compute exponential tensors $\mathcal{E}_\mathbf{k} \in \mathbb{R}^{2\times d_k\times m\times m}, \quad \mathcal{E}_\mathbf{v} \in \mathbb{R}^{2\times d_v\times m\times m}$
- Compute the estimates $\hat{\mathcal{Z}}_\mathbf{k} \in \mathbb{R}^{2\times d_k\times m\times m}, \quad \hat{\mathcal{Z}}_\mathbf{v} \in \mathbb{R}^{2\times d_v\times m\times m}$
- Compute the residuals $\mathcal{R} \in \mathbb{R}^{2\times d_k\times m\times m}$
- Compute the element-wise norm of the residuals: $|\mathcal{R}_\mathbf{qk}[k,i,j]|^2 \in \mathbb{R}^{d_k\times m\times m}$
- Compute the precision tensors $\mathcal{P}^C_\mathbf{k} \in \mathbb{R}^{d_k\times m\times m}, \quad \mathcal{P}^C_\mathbf{v} \in \mathbb{R}^{d_v\times m\times m}$.
- Apply a causal attention mask to $\mathcal{P}^C_\mathbf{k}, \mathcal{P}^C_\mathbf{v}$, masking out all $j > i$.
- Compute the attention tensor $\mathcal{Q} \in \mathbb{R}^{d_v\times m\times m}$.

- Compute the estimate $\bar{\mathbf{Z}} \in \mathbb{R}^{2\times d_v \times m}$.
- Compute the prediction $\mathbf{Z}^{\text{pred}} \in \mathbb{C}^{2\times d_e \times m}$.

This defines a single AFA module. Note that all of these operations can be computed in parallel. We predict the next time step across all inputs in parallel: given network inputs $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_m]$, we aim to predict $\mathbf{Z}' = [\mathbf{z}_2, \ldots, \mathbf{z}_{m+1}]$.

Since $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_P$ are unconstrained, $\mathbf{A}$ can still be complex-valued. The inputs to the network are real, but the outputs can hence be complex valued. Therefore, we can use a Mean Squared Error loss on both the real and imaginary parts:

$$\mathcal{L}_{MSE}(\bar{\mathbf{z}}_m, \mathbf{z}_{m+1}) = \frac{1}{m}\sum_m \left(\mathcal{R}(\bar{\mathbf{z}}_{m+1} - \mathbf{z}_{m+1})^2 + \mathcal{I}(\bar{\mathbf{z}}_{m+1} - \mathbf{z}_{m+1})^2\right)$$

(In fact, this is just a regular MSE loss on the full $2 \times d_e \times m$ output). Rather than learning to set $\mathbf{W}_P = \mathbf{S}$, $\mathbf{W}_V = \mathbf{S}^{-1}$, and setting $\boldsymbol{\lambda}$ to the correct values, it is possible for the model to learn a degenerate solution by zeroing out the dynamics $\boldsymbol{\lambda} = \mathbf{0}$ and the noise terms $\boldsymbol{\lambda}_\Omega, \boldsymbol{\lambda}_{\Omega_0}, \boldsymbol{\lambda}_\Gamma = \mathbf{0}$, so that the attention matrix becomes identity, and $\mathbf{W}_P\mathbf{W}_V = \mathbf{A}$. That is, the model bypasses the diagonalization of the dynamics and the maximum likelihood estimation, and uses the rotation matrices $\mathbf{W}_P, \mathbf{W}_V$ to represent the dynamics. If the scale of the noise in the data is sufficient, this behavior should be discouraged by the need to use multiple time steps for estimation. Optionally, we can introduce a regularization term to discourage this behavior: $\mathcal{L}_{inverse} = \|\mathbf{W}_P\mathbf{W}_V - \mathbf{I}_{d_e}\|_F^2$, where $\mathbf{I}_{d_e}$ is the $d_e \times d_e$ identity matrix and F denotes the Frobenius norm. However, if $d_v < d_e$, then $\mathbf{W}_P, \mathbf{W}_V$ are not full rank, so we can instead apply a lower-dimensional penalty: $\mathcal{L}_{inverse} = \|\mathbf{W}_V\mathbf{W}_P - \mathbf{I}_{d_v}\|_F^2$.

4.2 Experiments

We tested our model on a simple 2D dynamical system with state transition $\mathbf{A} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1}$, where:

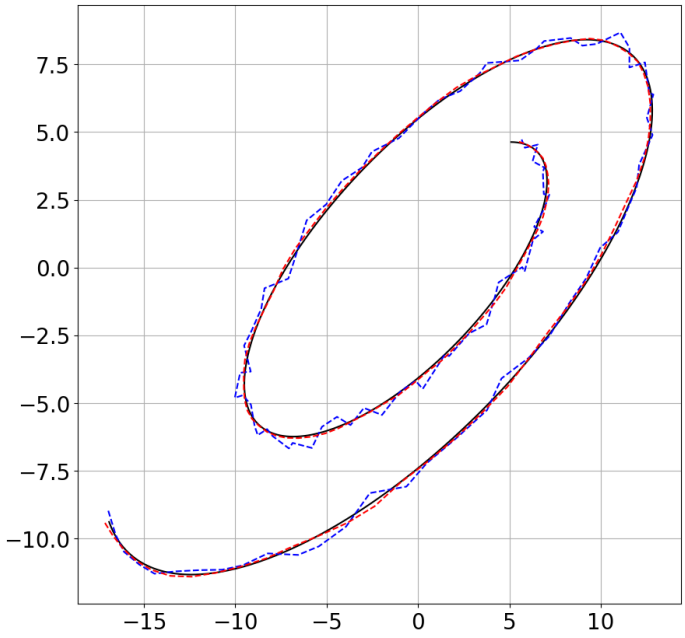
$$\mathbf{\Lambda} = \begin{bmatrix} -0.1 - i & 0 \\ 0 & -0.1 + i \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} 1 - i & 1 + i \\ 1 & 1 \end{bmatrix}, \quad \mathbf{S}^{-1} = \frac{1}{2} \begin{bmatrix} i & 1 - i \\ -i & 1 + i \end{bmatrix}$$

with zero-mean Gaussian process and measurement noise, projected into an embedding dimension of $d_e = 128$. After learning, we project the model's output back into 2D to assess its performance. This toy example allows us to debug the model and verify its ability to recover underlying temporal structure and perform accurate state estimation under uncertainty.

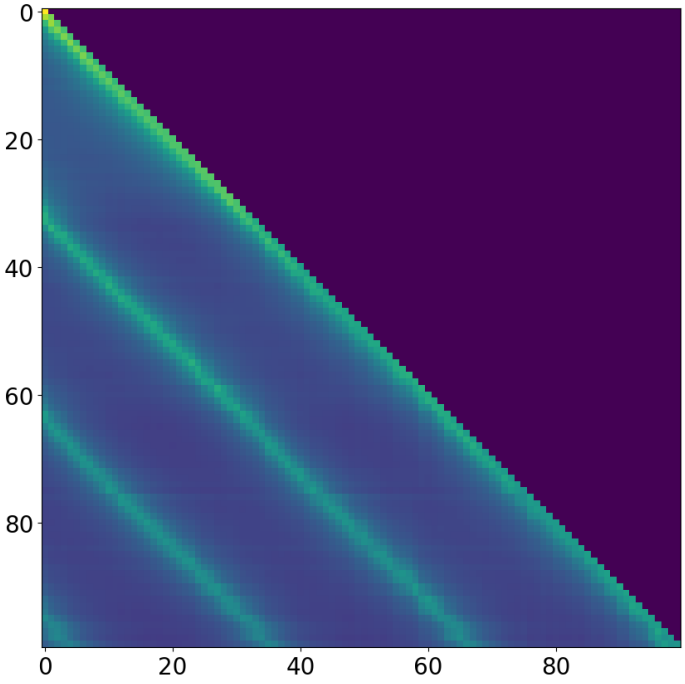
The results show that the model successfully learns a periodic attention pattern reflecting the underlying time dependence. As shown in Figure 4.1, the predicted trajectory closely tracks the true trajectory despite measurement noise.

To check that the model has learned the right dynamics, we map each measurement forward through the learned dynamics, to compute their estimates at four time points. Figure 4.2 shows the resulting estimates at three points during training. As training progresses, these state estimates converge to point clouds near the ground truth.

Finally, we test the consistency of the model by recursively applying the filter with the correct dynamic model. As shown in Figure 4.3, repeated application of the adaptive filter converges to a smooth trajectory near the ground truth. Figure 4.3 confirms that recursive application of the filter converges smoothly to accurate trajectories, supporting the model's practical utility in online inference. While these results validate our approach on a simple

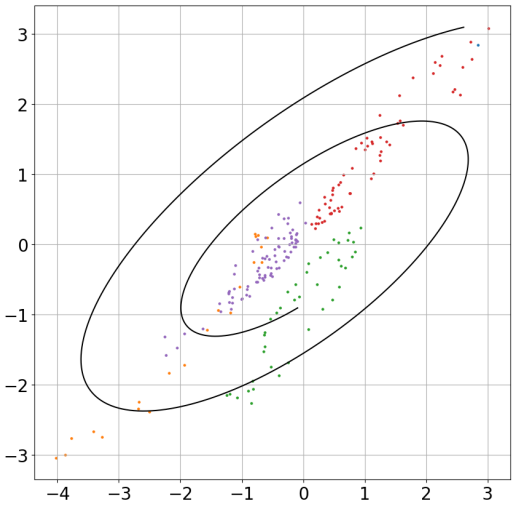


(a) Noisy trajectory (blue) and predicted trajectory (red).

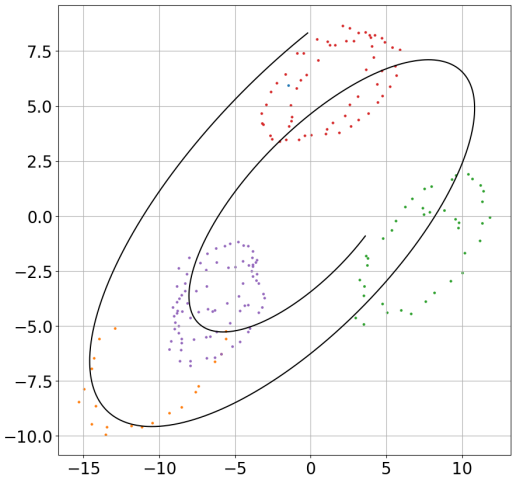


(b) Attention matrix, showing learned periodic structure.

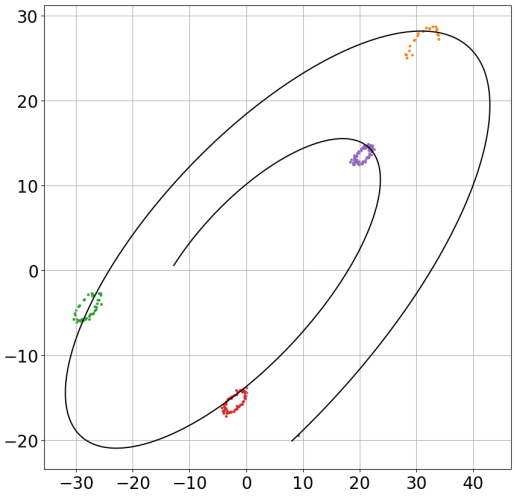
Figure 4.1: Performance of the filter on a simple 2D system.



(a) State estimates early in learning.



(b) State estimates midway through learning.



(c) State estimates near the end of learning.

Figure 4.2: Comparison of state estimates at different learning stages.

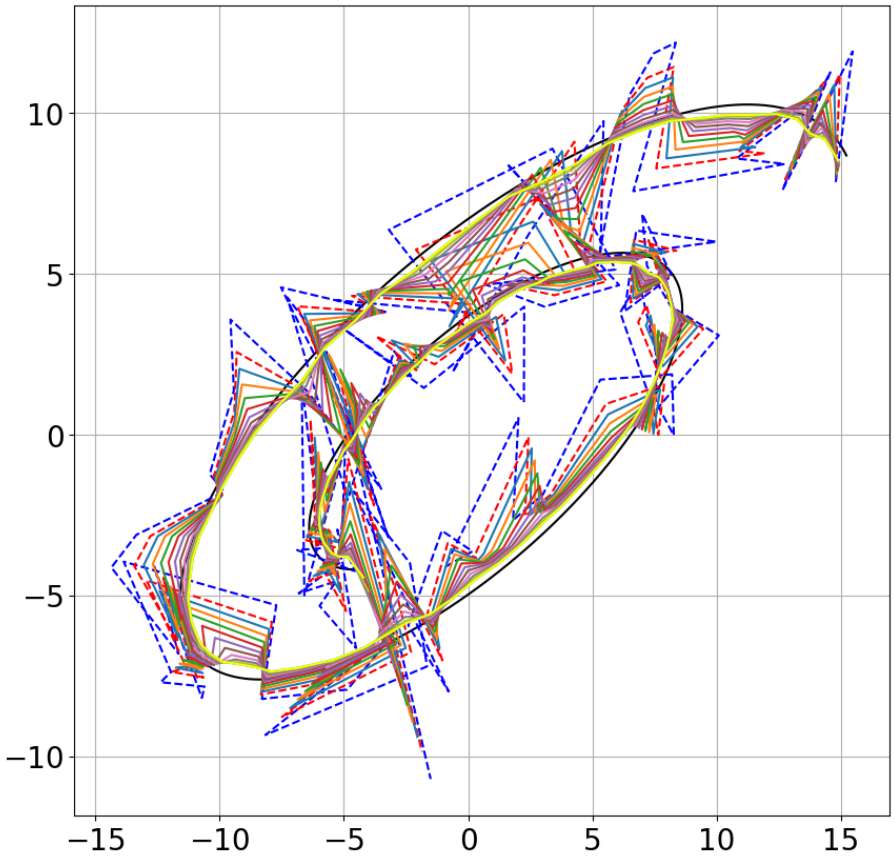


Figure 4.3: Iterated application of the filter converges from noisy measurements (blue dotted line) to a much smoother trajectory (yellow).

test case, extending evaluation to higher-dimensional and nonlinear systems is an important direction for future work.

CHAPTER 5

Related Work

Classical Methods for Distributed Filtering

The Kalman filter provides an optimal recursive solution for linear state estimation in the presence of Gaussian noise (Kal60). For foundational material on linear systems and optimal control, we refer the reader to standard texts such as (Ant97; SP05; DFT91; Kir04). For introductory material on stochastic processes and adaptive filtering, standard texts include (Jaz70; Spe08; Ste87; Hay02).

The Diffusion Kalman Filter (DKF) is a decentralized algorithm for distributed state estimation over sensor networks, in which each node independently performs a Kalman filter update based on its local measurements, and then diffuses its estimate and error covariance to neighboring nodes through a weighted sum (CLS09).

Building on this foundation, the Consensus+Innovations Kalman Filter (CIKF) separates the update into two stages: first, each node performs a consensus step, averaging its estimate with those of its neighbors; then, it applies an innovation step, correcting this consensus estimate using its own measurement and a local Kalman gain (DM17).

The Diffusion Least Mean Squares (LMS) algorithm, in contrast, is a model-free, adaptive

filtering method where each node updates its estimate by locally minimizing the prediction error using stochastic gradient descent, and then sharing and combining this information with neighbors (CS10).

While DKF and CIKF rely on known linear dynamical models to perform model-based Bayesian filtering, diffusion LMS methods emphasize data-driven, iterative adaptation, trading off model precision for flexibility and robustness. Our method is a model-based precision-weighted attention mechanism that adaptively estimates uncertainty from residual errors, combining the structured inference of diffusion Kalman filters with the adaptive character of diffusion LMS. For an overview of diffusion algorithms for adaptation and learning on networks, see (Say13).

Neural State Space Models and Dynamical Sequence Modeling

State space models (SSMs) provide a powerful and principled framework for modeling sequences and time series data by representing system dynamics through latent states that evolve over time. A key challenge in sequence modeling is maintaining memory over long time horizons, and avoiding the vanishing and exploding gradient problem faced by recurrent neural networks. The HiPPO framework improves memory retention in SSMs by introducing recurrent memory with optimal polynomial projections, which allows the model to compress and retain important information from the entire history in a structured latent state space (GDE20).

Building on this foundation, the Structured State Space Sequence model (S4) designs efficient state space layers that scale linearly with sequence length, using structured representations of the system matrices (GGR22) . Subsequent works such as Mamba further

enhance efficiency by introducing selective state space components that dynamically focus computation on relevant parts of the sequence (GD24).

The S4ND model applies state space models to multidimensional signals such as images and videos, interpreting them as higher-dimensional spatiotemporal signals (NGG22) . Goel et al. propose a multi-scale architecture based on S4 designed specifically for raw audio waveform generation, and address instability issues in SSM sequence modeling (GGD22). Mamba Mixer combines structured state space models with selective token and channel mixing via learned gating, enabling efficient and adaptive modeling of long sequences by focusing on relevant elements and features (BSZ24). Finally, simplified variants of state space layers have been proposed to reduce model complexity and improve training stability while retaining the key benefits of state space modeling in sequence tasks (SWL23).

Attention as a Structured State Space Computation

The attention mechanism was first introduced by Bahdanau et al. (2015) (BCB16) to improve neural machine translation by allowing models to dynamically focus on relevant input tokens, and was later generalized into the self-attention architecture of the Transformer by Vaswani et al. (2017), which became the foundation for modern large language models (VSP23).

Attention mechanisms can be interpreted through the lens of structured state space computations. Several recent works have explored these connections and proprosed structured alternatives to standard Transformer attention based on state space models.

In "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention,"

the authors propose a Linear Attention framework that approximates attention weights using kernel feature maps, enabling the Transformer to be expressed in a recurrent form. While this formulation offers computational benefits, it does not explicitly connect the recurrence to an underlying system dynamics model; rather, the recurrent structure serves primarily as an efficient implementation (KVP20).

"Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality" establishes that an SSM transformation is identical to multiplication by an N-sequentially semi-separable (SSS) matrix. Since Transformers attention matrices can be written as structured matrix multiplications of this kind, this result shows that self-attention can be implemented using SSMs (DG24).

"Transformers to SSMs: Distilling Quadratic Knowledge to Subquadratic Models" focuses on distilling quadratic self-attention into efficient SSMs. The goal is to match the performance of Transformers while reducing computation (BLX25).

Other recent works explore the integration of attention mechanisms with state space models. For example, "Understanding the differences in Foundation Models: Attention, State Space Models, and Recurrent Neural Networks" shows how attention kernels, including softmax, can be recast into causal convolutional forms, equivalent to this recurrence (SAD24).

Continuous-Time and Dynamical Systems Perspectives on Attention

A growing body of work explores attention mechanisms through the lens of continuous-time and dynamical systems, where neural networks learn or exploit latent state dynamics, often modeled as ordinary differential equations (ODEs) or stochastic processes. These

approaches provide principled ways to model temporal dependencies, uncertainty, and adaptivity in sequence modeling.

Stochastic Recurrent Networks (STORNs) combine recurrent neural networks with time-dependent latent variables and variational inference, allowing them to model complex, multimodal temporal dynamics by capturing both deterministic transitions and structured uncertainty at each time step (BO15).

Neural ODEs model the evolution of hidden states as continuous-time dynamics parameterized by a neural network defining the derivative, and use an ODE solver to compute outputs adaptively in time (CRB19).

Neural Processes (NPs) are a class of models that combine the function uncertainty modeling of Gaussian processes with the scalability and expressiveness of neural networks. NPs learn to map input-output examples to a latent representation of the underlying function, which is then used to generate predictions and uncertainty estimates for new inputs (GSR18).

Neural ODE Processes (NDPs) combine Neural ODEs with Neural Processes to model time series with both temporal dynamics and uncertainty. They define a distribution over ODE functions by inferring a latent initial state and a global control signal from context data, enabling stochastic predictions over continuous time. NDPs model global uncertainty over continuous latent dynamics via distributions over ODE trajectories, whereas we model uncertainty via learned process and noise covariance matrices (NBD21).

Attentive Neural Processes (ANPs) generalize NPs by introducing attention-based dynamic weighting over context sets, bridging the gap between amortized function learning and

localized, example-specific conditioning (KMS19).

In "Transformers for Modeling Physical Systems," the authors use transformers to model physical dynamical systems by first embedding the high-dimensional physical states into a lower-dimensional vector space using Koopman-based embeddings. The transformer is then trained auto-regressively on these embedded time series to predict future states (GZ22).

Attentive Differential Networks (ADNs) apply attention over continuous-time dynamics to handle irregularly sampled data, but they do not integrate the system's state transition matrix into the attention computation (CC23).

"ACE-NODE: Attentive Co-Evolving Neural Ordinary Differential Equations" introduces a pair of neural ODEs — one for modeling primary dynamics, and another for evolving attention weights in time. This method learns attention weights through an auxiliary ODE without explicitly incorporating known system dynamics (JJK21).

"Neural Controlled Differential Equations for Irregular Time Series" uses CDEs (controlled differential equations) for modeling sequence data. The inputs drive the system's hidden state dynamics. Whereas they model latent trajectories driven by inputs, our model learns internal dynamics and uses them in attention weighting, rather than only for state propagation (KMF20).

"Attentive Neural Controlled Differential Equations for Time-series Classification and Forecasting" integrates attention into the neural controlled differential equation (NCDE) framework. It uses two coupled NCDEs: a bottom NCDE to compute continuous-time attention from the input path, and a top NCDE to process an attention-weighted version of

the input. ANCDEs generate attention at each time step based on the local dynamics of the signal, not by comparing a query to keys as in attention mechanisms (JSH21).

"Hopfield Networks is All You Need" establishes a fundamental equivalence between modern continuous-state Hopfield networks and the Transformer attention mechanism, framing attention as a dynamical system converging to energy minima that represent memory states (RSL21).

Shim (2021) shows that attention corresponds mathematically to a gradient ascent step on the log normalizer (log-sum-exp) of an exponential family distribution, linking it to modern Hopfield networks (Shi22).

In "Understanding Self-attention Mechanism via Dynamical System Perspective," the authors identify that high-performance neural networks exhibit a phenomenon akin to stiffness in ODEs, where certain components of the system evolve at much different rates. To address this, they propose that the self-attention mechanism functions as a stiffness-aware step size adapter to better handle these varying rates of change (HLQ23).

"Neural Continuous-Discrete State Space Models for Irregularly-Sampled Time Series" combines continuous-time latent state dynamics with discrete observations using learned stochastic differential equations and stable filtering techniques. It employs amortized variational inference to jointly learn the generative and inference models (AHL23).

"Score-Based Generative Modeling through Stochastic Differential Equations" shows that diffusion models can be formulated as continuous-time stochastic differential equations (SDEs). By learning the time-dependent score function (the gradient of the log density)

of perturbed data, these models use SDE solvers to sample high-quality data, achieving state-of-the-art results in image generation (SSK21).

Krishnan et al. introduce Structured Inference Networks, which use recurrent neural networks to perform efficient variational inference in nonlinear state space models. This approach captures the temporal dependencies of latent states, enabling scalable approximate Bayesian filtering in complex dynamical systems (KSS16).

Adaptive Filter Attention differs from these works by deriving attention weights directly from uncertainty propagation through learned system dynamics, integrating filtering and attention into a unified inference framework.

Neural Networks for Filtering

While the above methods focus on modeling dynamics and uncertainty, other works focus on integrating neural networks with classical filtering algorithms.

Deep Kalman Filters (DKFs) extend traditional Kalman filters by incorporating deep neural networks into the generative and inference models. The method introduces a probabilistic framework in which latent states evolve over time based on previous states, actions, and time intervals, while observations are generated from these latent states via neural networks. DKFs learn both the generative process and the approximate posterior using variational inference (KSS15).

In "Can a Transformer Represent a Kalman Filter?" the authors show that Transformers, despite their nonlinear architecture and softmax-based self-attention, can approximate

the classical Kalman Filter for linear time-invariant systems to arbitrary precision. The authors construct an explicit Transformer architecture — called the Transformer Filter — that replicates the Kalman filtering update by representing a kernel smoothing estimator (Nadaraya–Watson with a Gaussian kernel) through self-attention (GB24).

"Focus Your Attention (with Adaptive IIR Filters)" adds a preprocessing step before attention that shapes the input sequence over time using a learned, signal-processing-style filter (essentially a neural version of an IIR filter), and then passes the result to standard attention layers (LZW23).

Dynamic Filter Networks generate convolutional filters dynamically based on the input or conditioning data, allowing the filtering operation to adapt spatially or temporally (BJT16).

Probabilistic Attention and Uncertainty-Aware Inference

Other approaches explicitly model uncertainty in neural networks and attention mechanisms.

Foundational works such as "A Practical Bayesian Framework for Backpropagation Networks," (Mac92) and "Bayesian Learning for Neural Networks" (Nea96) laid the groundwork for modern developments in approximate Bayesian inference in deep learning, such as variational inference, dropout as approximate Bayesian inference, and deep ensembles. "Weight Uncertainty in Neural Networks" introduced "Bayes by Backprop," a practical variational inference method for training BNNs efficiently with stochastic gradient descent (BCK15).

Various approaches that embed structured or probabilistic biases directly into attention

mechanisms have been explored. For example, Structured Attention Networks (Kim et al., 2017) incorporate graphical models such as trees or conditional random fields to compute attention distributions, introducing learned structural biases into attention (KDH17).

"Probabilistic Transformers" model attention as MAP inference in a Gaussian mixture model, where each key/value pair has its own precision parameter, with expectation-maximization (EM) parameter updates (MG20). "Probabilistic Attention for Interactive Segmentation" applies this probabilistic interpretation to enable online adaptation of keys and values during inference, particularly for interactive tasks where user corrections are available (e.g. segmentation) (GBM21).

"Uncertainty-Aware Attention for Reliable Interpretation and Prediction" introduces an attention mechanism that models input-specific uncertainty through variational inference, improving both prediction reliability and interpretability by quantifying how confident the model is in each attention weight (HLK18).

BayesFormer introduces a principled Bayesian framework for uncertainty estimation in Transformer models by reinterpreting dropout as approximate variational inference. The key idea is to treat each learnable parameter as a random variable with a prior distribution, typically a Gaussian distribution (SWF22).

Other recent probabilistic attention methods such as the Uncertainty-Guided Probabilistic Transformer (UGPT) (GWJ22), Attention Gaussian Process (AGP) (SMM23), Gaussian Adaptive Attention (GAAM) (ICE24a), and Probabilistic Attention Regularization for Language Guided Image Classification (PARIC) (NGS25), introduce uncertainty-aware mechanisms into Transformer-based architectures by modeling attention scores or embeddings

as random variables. Other approaches including Shifting Attention to Relevance (SAR), which shifts attention to more relevant tokens and sentences by re-weighting token-level and sentence-level entropy based on learned relevance scores (DCW24). The Correlated Gaussian Process Transformer (CGPT) (BHD25) models self-attention as cross-covariances between correlated Gaussian processes to enable asymmetric attention with uncertainty quantification, while Kernel-Eigen Pair Sparse Variational Gaussian Processes (KEP-SVGP) (CTT24) model asymmetric self-attention kernels in Transformers via kernel singular value decomposition.

Structuring Attention with Priors and Positional Biases

The original Transformer lacks built-in modeling of relative or absolute positions, relying instead on added absolute position encodings. Several methods have proposed extending self-attention to efficiently incorporate relative position information, which are analogous to the relative exponential decay in our method (VSP23).

For example, "Self-Attention with Relative Position Representations" introduces a mechanism to encode relative distances between tokens directly into the attention computation (SUV18).

"RoFormer: Enhanced Transformer with Rotary Position Embedding" applies a position-dependent rotation to each dimension of the query and key vectors before computing their dot products. This rotation encodes the relative distance between positions directly into the attention mechanism (SLP23).

The Retentive Network (RetNet) introduces a retention mechanism that blends each to-

ken's hidden state with an exponentially decaying summary of previous tokens. RetNet uses multiple attention heads, each having its own learned exponential decay rate, enabling multi-scale memory of different temporal lengths. Attention is applied with decay masks that enforce causality and decay over relative positions, while a gating mechanism modulates the retained summaries to improve expressivity and control the contribution of past information (SDH23).

The Dynamically Context-Sensitive Time-Decay Attention mechanism learns a time-decay function to adjust attention weights. However, this approach focuses on temporal proximity rather than integrating known system dynamics (SYC18).

The SAT-Transformer (2024) introduces learned temporal kernels that multiply attention weights element-wise before normalization (KL24).

"Transformer Dissection: A Unified Understanding of Transformer's Attention via the Lens of Kernel" interprets attention as a kernel smoother, where attention weights correspond to similarity scores from a kernel function (TBY19).

Implicit Kernel Attention (IKA) learns attention kernels by modeling the spectral density with neural networks, allowing the similarity measure to adapt beyond fixed RBF kernels. IKA focuses on learning adaptable pairwise similarity functions from data rather than an uncertainty-driven input weighting (SJK21).

Performer approximates softmax attention with positive-definite kernel methods to achieve scalable, linear-time self-attention for long sequences (CLD22).

Gaussian Adaptive Attention (GAAM) introduces a kernel-based attention mechanism that adaptively biases attention weights using Gaussian functions centered on each query position, effectively emphasizing nearby tokens and encouraging locality in attention (ICE24b).

Elliptical Attention replaces the traditional dot-product attention with a Mahalanobis distance-based mechanism, to emphasize contextually relevant directions in the feature space (NAT24).

Several papers have explored element-wise (in the embedding dimension) attention mechanisms. For example, "Element-wise Attention Is All You Need," uses an exponentiated Euclidean distance as a similitary metric (Fen25). "Element-Wise Attention Layers: an option for optimization" replaces matrix multiplications with learned spatial weight arrays applied directly via element-wise multiplication, focusing attention along rows and columns (BS23).

Our method differs from these methods in that the paramaterization of our smoothing term arises naturally from the propagation of the covariance through the dynamics, rather than heuristics or learned kernels.

CHAPTER 6

Discussion and Conclusion

This work explores a connection between classical control theory and attention mechanisms by framing attention through the lens of linear dynamical systems and adaptive filtering. We find that a form of attention emerges naturally as the solution to a maximum likelihood estimation under the assumption of a linear time invariant dynamical system with Gaussian noise. Attention functions as a single step in an amortized adaptive filter by interpreting each key-value pair as a noisy observation of a latent state and performing a precision-weighted Bayesian update that aggregates evidence to refine the state estimate in a computationally efficient, feedforward manner.

The rank-1 update, and its scalar reweighting approximation, provide a principled way to integrate prior information about the precision matrix into the attention mechanism. Unlike softmax-based attention, which lacks a statistical grounding in the dynamics or uncertainty structure, this update arises naturally from Bayesian filtering theory. It allows us to modulate the influence of each past state based on both the model's confidence (via the precision priors) and the similarity to the current estimate.

I plan to apply the method to higher-dimensional dynamical systems and real-world data, such as in natural language and computer vision, and compare its performance with that of

other architectures.

Several promising directions remain for future work. Currently, the AFA dynamics model is limited to diagonalizable linear systems. It is possible to derive a generalization for diagonalizable plus low-rank matrices (see Appendix), but the resulting expression is unwieldy and applies only for small perturbations from the diagonalizable case. It may be possible to improve this approximation, or to make use of the Jordan canonical form or Krylov subspace methods to capture a wider class of dynamical systems.

The learned precision matrices and exponential kernels in our model encode uncertainty propagation, effectively modeling how the probability density over the latent state evolves in time according to those dynamics. The layer-wise iterative update of the posterior mean and precision can be interpreted as a discretized approximation of the evolution of the filtering distribution's density, akin to numerically solving the Fokker–Planck equation forward in time. Each attention layer in AFA acts like a learned, uncertainty-aware denoising step in a diffusion model—removing noise from latent states by combining noisy measurements with learned priors. It may be useful to explore further connections to score-based techniques, including score-matching and diffusion models.

In standard attention, each layer computes a weighted average over all previous inputs, meaning the output at each step depends on the entire input history, not just the output of the previous layer. This makes the forward pass inherently non-Markovian across layers. To enable parallel training, the covariance is recomputed independently at each layer from scratch, rather than propagated forward. By contrast, classical filtering methods like the Kalman filter perform recursive updates of both the state estimate and its covariance, relying only on the previous estimate and the current observation. At inference time, since we've

already learned a transition model and noise structure during training, we could potentially replace each attention layer with a sequential Kalman-style update, recursively propagating the state and precision forward. This would avoid recomputing attention at every layer and significantly improve inference speed, though at the cost of replacing a flexible, expressive attention mechanism with a simpler, fixed recursive update.

Extending AFA to incorporate control inputs would allow the model to account for how actions influence system dynamics, enabling controlled state estimation rather than just passive inference. AFA integrates the Kolmogorov forward equation (Fokker–Planck), which evolves distributions over states forward in time, capturing uncertainty evolution. A complementary formulation could potentially integrate the Hamilton-Jacobi-Bellman equation for stochastic control, which governs how the value function evolves backward in time, incorporating the effect of noise (diffusion).

Finally, it would be valuable to further explore connections between neural network models and classical methods in control theory and adaptive filtering. For example, developing a serial formulation closer to that of the Kalman Filter could enable incremental updates with lower computational and memory costs. We could also investigate separating the consensus and innovation steps, as in the CIKF algorithm — doing consensus via attention first, then applying an innovation-like correction using local data.

CHAPTER 7

Appendix

7.1 Propagation of variance through an LTI

Recall that the state at time t_j is:

$$\mathbf{x}(t_j) = e^{\mathbf{A}\Delta t_{ij}} \left(\mathbf{x}(t_i) + \int_0^{\Delta t_{ij}} e^{-\mathbf{A}\tau} \boldsymbol{\eta}(\tau) d\tau \right).$$

The covariance of the integral term is:

$$\begin{aligned}
\mathbf{V}_B(\Delta t_{ij}) &:= \mathbb{E}\left[\left(\int_0^{\Delta t_{ij}} e^{-\mathbf{A}\tau_1} \boldsymbol{\eta}(\tau_1) d\tau_1 \right) \left(\int_0^{\Delta t_{ij}} e^{-\mathbf{A}\tau_2} \boldsymbol{\eta}(\tau_2) d\tau_2 \right)^T \right] \\
&= \mathbb{E}\left[\int_0^{\Delta t_{ij}} \int_0^{\Delta t_{ij}} e^{-\mathbf{A}\tau_1} \boldsymbol{\eta}(\tau_1) \boldsymbol{\eta}(\tau_2)^T e^{-\mathbf{A}^T\tau_2} d\tau_1 d\tau_2 \right] \\
&= \int_0^{\Delta t_{ij}} e^{-\mathbf{A}\tau} \mathbb{E}\left[\boldsymbol{\eta}(\tau) \boldsymbol{\eta}(\tau)^T \right] e^{-\mathbf{A}^T\tau} d\tau
\end{aligned}$$

(by independence of $\boldsymbol{\eta}(\tau_1)$ and $\boldsymbol{\eta}(\tau_2)$ and linearity of expectation)

$$= \int_0^{\Delta t_{ij}} e^{-\mathbf{A}\tau} \boldsymbol{\Omega} e^{-\mathbf{A}^T\tau} d\tau$$

As the system evolves, the state distribution evolves as:

$$\mathbf{x}(t_j) \sim \mathcal{N}\left(e^{\mathbf{A}\Delta t_{ij}} \mathbf{x}(t_i), \mathbf{V}_F(\Delta t_{ij}) \right)$$

where the forward-propagated covariance is:

$$\mathbf{V}_F(\Delta t_{ij}) = e^{\mathbf{A}\Delta t_{ij}} \mathbf{V}_B(\Delta t_{ij}) e^{\mathbf{A}^T \Delta t_{ij}}$$

Recall that the estimate of the state is:

$$\hat{\mathbf{x}}_i(t_j) := e^{-\mathbf{A}\Delta t_{ij}}\mathbf{x}(t_j) = \mathbf{x}(t_i) + \int_0^{\Delta t_{ij}} e^{-\mathbf{A}\tau}\boldsymbol{\eta}(\tau)d\tau.$$

Hence, $\hat{\mathbf{x}}_i(t_j) \sim \mathcal{N}\left(\mathbf{x}(t_i), \mathbf{V}_B(\Delta t_{ij})\right)$, where: $\mathbf{V}_B(\Delta t_{ij})$ quantifies the uncertainty of the estimate $\hat{\mathbf{x}}_i(t_j)$ of the state $\mathbf{x}(t_i)$ at an earlier time t_i, given the state $\mathbf{x}(t_j)$ at a later time t_j. Conversely, if we observe the state at an earlier time t_j and wish to quantify our uncertainty about an estimate of its state at a later time t_i (i.e. the causal case), we can switch indices i and j and solve for $\mathbf{x}(t_j)$:

$$\mathbf{x}(t_i) = e^{\mathbf{A}(t_i - t_j)}\mathbf{x}(t_j) + \int_{t_j}^{t_i} e^{\mathbf{A}(t_i-\tau)}\boldsymbol{\eta}(\tau)d\tau$$

$$= e^{-\mathbf{A}\Delta t_{ij}}\left(\mathbf{x}(t_j) + \int_0^{-\Delta t_{ij}} e^{-\mathbf{A}\tau}\boldsymbol{\eta}(\tau)d\tau\right).$$

$$\mathbf{x}(t_j) = e^{\mathbf{A}\Delta t_{ij}}\mathbf{x}(t_i) - \int_0^{-\Delta t_{ij}} e^{-\mathbf{A}\tau}\boldsymbol{\eta}(\tau)d\tau$$

We again form an estimate:

$$\hat{\mathbf{x}}_i(t_j) = e^{-\mathbf{A}\Delta t_{ij}}\mathbf{x}(t_j) = \mathbf{x}(t_i) - e^{-\mathbf{A}\Delta t_{ij}}\int_0^{-\Delta t_{ij}} e^{-\mathbf{A}\tau}\boldsymbol{\eta}(\tau)d\tau$$

which has distribution: $\hat{\mathbf{x}}_i(t_j) \sim \mathcal{N}\left(\mathbf{x}(t_i), \mathbf{V}_F(-\Delta t_{ij})\right)$,
where $\mathbf{V}_F(-\Delta t_{ij}) = e^{-\mathbf{A}\Delta t_{ij}}\mathbf{V}_B(-\Delta t_{ij})e^{-\mathbf{A}^T\Delta t_{ij}} = e^{\mathbf{A}|\Delta t_{ij}|}\mathbf{V}_B(|\Delta t_{ij}|)e^{\mathbf{A}^T|\Delta t_{ij}|}$.

So, in general, $\hat{\mathbf{x}}_i(t_j) \sim \mathcal{N}\left(\mathbf{x}(t_i), \mathbf{V}(\Delta t_{ij})\right)$, where

$$\mathbf{V}(\Delta t_{ij}) = \begin{cases} \mathbf{V}_B(\Delta t_{ij}) & \Delta t_{ij} > 0 \quad \text{(non-causal case)} \\ \mathbf{V}_F(|\Delta t_{ij}|) & \Delta t_{ij} \leq 0 \quad \text{(causal case)} \end{cases}$$

Note that this is not the same as simply reversing time in the system, i.e. flipping the sign of $\mathbf{A}$ in the expression for $\mathbf{V}_B$, because in stochastic systems, the noise process $\boldsymbol{\eta}(t)$ is not time-reversible: the uncertainty added between t_i and t_j depends on the direction of

propagation.

Recall that the measurement at time t_j is:

$$\mathbf{z}(t_j) = \mathbf{C}e^{\mathbf{A}\Delta t_{ij}} \left(\mathbf{x}(t_i) + \int_0^{\Delta t_{ij}} e^{-\mathbf{A}\tau} \boldsymbol{\eta}(\tau) d\tau \right) + \boldsymbol{\gamma}(t_j).$$

The measurement process is distributed as:

$$\mathbf{z}(t_j) \sim \mathcal{N}\left(\mathbf{C}e^{\mathbf{A}\Delta t_{ij}} \mathbf{x}(t_i), \mathbf{C}\mathbf{V}_F(\Delta t_{ij})\mathbf{C}^T + \boldsymbol{\Gamma} \right)$$

Recall that our estimated measurement is defined as:

$$\hat{\mathbf{z}}_i(t_j) := \mathbf{C}e^{-\mathbf{A}\Delta t_{ij}} \mathbf{C}^{-1} \mathbf{z}(t_j) := \mathbf{Q}(\Delta t_{ij}) \mathbf{z}(t_j)$$

where $\mathbf{Q}(\Delta t) = \mathbf{C}e^{-\mathbf{A}\Delta t}\mathbf{C}^{-1}$. In terms of the state estimate, this is:

$$\hat{\mathbf{z}}_i(t_j) = \mathbf{C}e^{-\mathbf{A}\Delta t_{ij}} \mathbf{x}(t_j) + \mathbf{C}e^{-\mathbf{A}\Delta t_{ij}} \mathbf{C}^{-1} \boldsymbol{\gamma}(t_j)$$

$$= \mathbf{C}\hat{\mathbf{x}}_i(t_j) + \mathbf{Q}(\Delta t_{ij}) \boldsymbol{\gamma}(t_j)$$

Hence,

$$\hat{\mathbf{z}}_i(t_j) \sim \mathcal{N}\left(\mathbf{z}^C(t_i), \mathbf{V}^C(\Delta t_{ij}) \right)$$

where the propagated measurement covariance is:

$$\mathbf{V}^C(\Delta t_{ij}) = \mathbf{C}\mathbf{V}(\Delta t_{ij})\mathbf{C}^T + \mathbf{Q}(\Delta t_{ij})\boldsymbol{\Gamma}\mathbf{Q}^T(\Delta t_{ij})$$

$$= \mathbf{C}e^{-\mathbf{A}\Delta t_{ij}} \left(\mathbf{V}_F(\Delta t_{ij}) + \mathbf{C}^{-1}\boldsymbol{\Gamma}\mathbf{C}^{-T} \right) e^{-\mathbf{A}^T \Delta t_{ij}} \mathbf{C}^T$$

Note that $\mathbf{V}(\Delta t)$ and $\mathbf{V}^C(\Delta t)$ and their inverses are positive definite (PD) and symmetric for all Δt. (This is because, for any symmetric PD matrix $\mathbf{A}$ and any matrix $\mathbf{B}$, $\mathbf{B}\mathbf{A}\mathbf{B}^T$ is also symmetric and PD, since, for any $\mathbf{x}$, $\mathbf{x}^T\mathbf{B}\mathbf{A}\mathbf{B}^T\mathbf{x} = \mathbf{y}^T\mathbf{A}\mathbf{y} \geq 0$. The sum (or integral) of two symmetric, PD matrices is also symmetric and PD. Finally, the inverse of a symmetric PD matrix is also symmetric and PD.)

7.2 Maximum Likelihood Trajectory Estimation

We wish to estimate the trajectory of the system from n measurements taken at times $\{t_1, ..., t_n\}$, which we denote $\{\mathbf{z}(t_1), ..., \mathbf{z}(t_n)\}$. We can do so by solving a separate maximum likelihood estimation (MLE) problem for each $\mathbf{x}_i$.

Let $\mathbf{x}_i = \mathbf{x}(t_i)$, $\hat{\mathbf{x}}_{ij} = \hat{\mathbf{x}}_i(t_j)$ and $\mathbf{V}_{ij} = \mathbf{V}(\Delta t_{ij})$. Following (Tab21), note that the PDF of $\hat{\mathbf{x}}_{ij}$ is:

$$f_X\left(\hat{\mathbf{x}}_{ij}\right) = \frac{(2\pi)^{-N/2}}{\sqrt{\left|\det\left(\mathbf{V}_{ij}\right)\right|}} \exp\left(-\frac{1}{2}\left(\hat{\mathbf{x}}_{ij} - \mathbf{x}_i\right)^T \mathbf{V}_{ij}^{-1}\left(\hat{\mathbf{x}}_{ij} - \mathbf{x}_i\right)\right)$$

The likelihood function is given by:

$$L(\mathbf{x}_i, \mathbf{V}_{1i}, \ldots, \mathbf{V}_{ni} \mid \mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{j=1}^{n} f_X\left(\hat{\mathbf{x}}_{ij} \mid \mathbf{x}_i, \mathbf{V}_{ij}\right)$$

$$= \prod_{j=1}^{n} (2\pi)^{-\frac{N}{2}} \left|\det(\mathbf{V}_{ij})\right|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\left(\hat{\mathbf{x}}_{ij} - \mathbf{x}_i\right)^T \mathbf{V}_{ij}^{-1}\left(\hat{\mathbf{x}}_{ij} - \mathbf{x}_i\right)\right)$$

$$= (2\pi)^{-\frac{Nn}{2}} \left(\prod_{j=1}^{n} \left|\det(\mathbf{V}_{ij})\right|\right)^{-\frac{1}{2}} \exp\left(\sum_{j=1}^{n} -\frac{1}{2}\left(\hat{\mathbf{x}}_{ij} - \mathbf{x}_i\right)^T \mathbf{V}_{ij}^{-1}\left(\hat{\mathbf{x}}_{ij} - \mathbf{x}_i\right)\right)$$

So the log-likelihood is:

$$\ell = -\frac{Nn}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{n} \log\left|\det(\mathbf{V}_{ij})\right| - \frac{1}{2}\sum_{j=1}^{n} \left(\hat{\mathbf{x}}_{ij} - \mathbf{x}_i\right)^T \mathbf{V}_{ij}^{-1}\left(\hat{\mathbf{x}}_{ij} - \mathbf{x}_i\right)$$

Factoring out the term involving $\mathbf{x}_i$:

$$\ell = -\frac{1}{2}\sum_{j} \left(\hat{\mathbf{x}}_{ij}\mathbf{V}_{ij}^{-1}\hat{\mathbf{x}}_{ij} + \mathbf{x}_i^T\mathbf{V}_{ij}^{-1}\mathbf{x}_i - 2\mathbf{x}_i^T\mathbf{V}_{ij}^{-1}\hat{\mathbf{x}}_{ij}\right) + \text{const.}$$

Taking the derivative with respect to $\mathbf{x}_i$,

$$\frac{\partial \ell}{\partial \mathbf{x}_i} = \sum_{j} \left(\mathbf{x}_i^T\left(\mathbf{V}_{ij}^{-1} + \mathbf{V}_{ij}^{-T}\right) - 2\left(\mathbf{V}_{ij}^{-1}\hat{\mathbf{x}}_{ij}\right)^T\right) = 0.$$

$$\left(\sum_j \left(\mathbf{V}_{ij}^{-1} + \mathbf{V}_{ij}^{-T}\right)\right)^T \mathbf{x}_i = 2\sum_j \mathbf{V}_{ij}^{-1}\hat{\mathbf{x}}_{ij}.$$

Defining a precision matrix $\mathbf{P}_{ij} = \mathbf{V}_{ij}^{-1}$, the MLE solution for $\mathbf{x}(t_i)$ is a precision-weighted estimator:

$$\bar{\mathbf{x}}_i := \left(\sum_j \mathbf{P}_{ij}\right)^{-1} \sum_j \mathbf{P}_{ij}\hat{\mathbf{x}}_{ij}$$

(since $\mathbf{V}_{ij}$ is symmetric).

Letting $\mathbf{z}_i = \mathbf{z}(t_i)$, $\hat{\mathbf{z}}_{ij} = \hat{\mathbf{z}}_i(t_j)$ and $\mathbf{V}_{ij}^C = \mathbf{V}^C(\Delta t_{ij})$, and repeating the above with the PDF for $\hat{\mathbf{z}}_{ij}$, the log-likelihood is:

$$\ell = -\frac{1}{2}\sum_{j=1}^{n} \left(\hat{\mathbf{z}}_{ij} - \mathbf{z}_i^C\right)^T \mathbf{V}_{ij}^{C-1} \left(\hat{\mathbf{z}}_{ij} - \mathbf{z}_i^C\right) + \text{const.}$$

The MLE solution for $\mathbf{z}^C(t_i)$ is hence:

$$\bar{\mathbf{z}}_i := \left(\sum_j \mathbf{P}_{ij}^C\right)^{-1} \sum_j \mathbf{P}_{ij}^C \hat{\mathbf{z}}_{ij}, \quad \text{where} \quad \mathbf{P}_{ij}^C = \mathbf{V}_{ij}^{C-1}.$$

Repeating this computation for all t_i produces an estimate of the full trajectory.

For the causal case, we can simply change the range of summation, since the MLE is defined relative to the specific set of available data; therefore, restricting the summation to $j \leq i$ yields the MLE based only on the data accumulated up to index i:

$$\bar{\mathbf{z}}_i := \left(\sum_{j\leq i} \mathbf{P}_{ij}^C\right)^{-1} \sum_{j\leq i} \mathbf{P}_{ij}^C \hat{\mathbf{z}}_{ij}$$

7.3 Closed-form Precision Matrix Calculation for Diagonalizable Matrices

The estimation procedure above requires computing $\mathbf{P}_C(\Delta t_{ij})$, which is generally expensive, as it involves integration and matrix inversion. However, if we assume that $\mathbf{A}$ is diagonalizable, that is we can write it in the form $\mathbf{A} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1}$, where $\mathbf{\Lambda}$ is a diagonal matrix and $\mathbf{S}$ an invertible matrix, the covariance becomes:

$$\mathbf{V}_B(\Delta t) = \int_0^{|\Delta t|} e^{-\mathbf{A}\tau}\mathbf{\Omega}(\tau)e^{-\mathbf{A}^*\tau}d\tau$$

$$= \mathbf{S}\left(\int_0^{|\Delta t|} e^{-\mathbf{\Lambda}\tau}(\mathbf{S}^{-1}\mathbf{\Omega}\mathbf{S}^{-*})e^{-\mathbf{\Lambda}^*\tau}d\tau\right)\mathbf{S}^*$$

Let $\mathbf{G} = \mathbf{S}^{-1}\mathbf{\Omega}\mathbf{S}^{-*}$ and $\mathbf{K}(\Delta t) = \int_0^{|\Delta t|} e^{-\mathbf{\Lambda}\tau}(\mathbf{S}^{-1}\mathbf{\Omega}\mathbf{S}^{-*})e^{-\mathbf{\Lambda}^*\tau}d\tau$.

Note that $(e^{-\mathbf{\Lambda}t}\mathbf{G}e^{-\mathbf{\Lambda}^*t})_{kl} = \mathbf{G}_{kl}e^{-\lambda_k t}e^{-\lambda_l^* t} = \mathbf{G}_{kl}e^{-(\lambda_k+\lambda_l^*)t}$. Hence, the klth component of $\mathbf{K}(\Delta t)$ is:

$$\mathbf{K}_{kl}(\Delta t) = \mathbf{G}_{kl}\int_0^{|\Delta t|} e^{-(\lambda_k+\lambda_l^*)\tau}d\tau = \mathbf{G}_{kl}\frac{1-e^{-(\lambda_k+\lambda_l^*)|\Delta t|}}{\lambda_k+\lambda_l^*}$$

If we assume that $\mathbf{\Omega} = \mathbf{S}\mathbf{\Lambda}_\Omega\mathbf{S}^*$, i.e. that the process noise acts independently along the decoupled modes of the system, then $\mathbf{G} = \mathbf{\Lambda}_\Omega$ and $\mathbf{K}(t)$ becomes diagonal, with kkth component:

$$\mathbf{K}_{kk}(\Delta t) = \lambda_{\Omega k}\frac{1-e^{-2\mathrm{Re}(\lambda_k)|\Delta t|}}{2\mathrm{Re}(\lambda_k)}$$

Hence,

$$\mathbf{V}_B(\Delta t) = \mathbf{S}\mathbf{\Lambda}_{\mathbf{V}B}(\Delta t)\mathbf{S}^*$$

where

$$\mathbf{\Lambda}_{\mathbf{V}B}(\Delta t) = \mathbf{\Lambda}_\Omega\frac{\mathbf{I}-e^{-2\mathrm{Re}(\mathbf{\Lambda})|\Delta t|}}{2\mathrm{Re}(\mathbf{\Lambda})} = \mathbf{\Lambda}_\Omega\frac{e^{-2\mathrm{Re}(\mathbf{\Lambda})|\Delta t|}-\mathbf{I}}{-2\mathrm{Re}(\mathbf{\Lambda})}$$

(Here, the division notation is short-hand for element-wise division.)

For the causal case,

$$\begin{aligned}\mathbf{V}_F(\Delta t_{ij}) &= e^{\mathbf{A}|\Delta t_{ij}|}\mathbf{V}_B(|\Delta t_{ij}|)e^{\mathbf{A}^T|\Delta t_{ij}|} \\ &= \mathbf{S}e^{\mathbf{\Lambda}|\Delta t_{ij}|}\mathbf{\Lambda}_{\mathbf{V}B}(|\Delta t_{ij}|)e^{\mathbf{\Lambda}^*|\Delta t_{ij}|}\mathbf{S}^* \\ &= \mathbf{S}e^{2\mathrm{Re}(\mathbf{\Lambda})|\Delta t_{ij}|}\left(\mathbf{\Lambda}_\Omega\frac{e^{-2\mathrm{Re}(\mathbf{\Lambda})|\Delta t_{ij}|}-\mathbf{I}}{-2\mathrm{Re}(\mathbf{\Lambda})}\right)\mathbf{S}^* \\ &= \mathbf{S}\mathbf{\Lambda}_{\mathbf{V}}(|\Delta t_{ij}|)\mathbf{S}^*\end{aligned}$$

where

$$\mathbf{\Lambda}_{\mathbf{V}}(|\Delta t_{ij}|) = \mathbf{\Lambda}_\Omega\frac{\mathbf{I}-e^{2\mathrm{Re}(\mathbf{\Lambda})|\Delta t_{ij}|}}{-2\mathrm{Re}(\mathbf{\Lambda})}$$

$\mathbf{\Lambda}_{\mathbf{V}}(|\Delta t_{ij}|)$ is positive semi-definite provided $\mathbf{\Lambda}_\Omega \geq \mathbf{0}$ (which is true by definition). (Here, the inequalities are applied element-wise.)

The propagated precision matrix $\mathbf{P}(t)$ can be easily obtained by inverting $\mathbf{\Lambda}_{\mathbf{V}}(\Delta t)$ element-wise: $\mathbf{P}(\Delta t) = \mathbf{S}^{-*}\mathbf{\Lambda}_{\mathbf{P}}(\Delta t)\mathbf{S}^{-1}$, where $\mathbf{\Lambda}_{\mathbf{P}}(\Delta t) = \mathbf{\Lambda}_{\mathbf{V}}(\Delta t)^{-1}$.

To include the measurement model, recall that the propagated measurement covariance is:

$$\mathbf{V}_C(\Delta t_{ij}) = \mathbf{C}\mathbf{V}(\Delta t_{ij})\mathbf{C}^* + \mathbf{Q}(\Delta t_{ij})\mathbf{\Gamma}\mathbf{Q}^*(\Delta t_{ij}), \quad \text{where} \quad \mathbf{Q}(\Delta t_{ij}) = \mathbf{C}e^{-\mathbf{A}\Delta t_{ij}}\mathbf{C}^{-1}.$$

If $\mathbf{C} = \mathbf{S}\mathbf{\Lambda}_C\mathbf{S}^{-1}$, this becomes: $\mathbf{Q}(\Delta t_{ij}) = \mathbf{S}\mathbf{\Lambda}_C e^{-\mathbf{\Lambda}\Delta t_{ij}}\mathbf{\Lambda}_C^{-1}\mathbf{S} = \mathbf{S}e^{-\mathbf{\Lambda}\Delta t_{ij}}\mathbf{S}^{-1}$. If, in addition, $\mathbf{\Gamma} = \mathbf{S}\mathbf{\Lambda}_\Gamma\mathbf{S}^*$, then:

$$\begin{aligned}\mathbf{V}^C(\Delta t_{ij}) &= \mathbf{S}\mathbf{\Lambda}_C\mathbf{\Lambda}_{\mathbf{V}}(\Delta t_{ij})\mathbf{\Lambda}_C^*\mathbf{S}^* + \mathbf{S}e^{-\mathbf{\Lambda}\Delta t_{ij}}\mathbf{\Lambda}_\Gamma e^{-\mathbf{\Lambda}^*\Delta t_{ij}}\mathbf{S}^* \\ &= \mathbf{S}\mathbf{\Lambda}_{\mathbf{V}}^C(\Delta t_{ij})\mathbf{S}^*\end{aligned}$$

where

$$\mathbf{\Lambda}_{\mathbf{V}}^C(\Delta t_{ij}) = \mathbf{\Lambda}_C^2\mathbf{\Lambda}_{\mathbf{V}}(\Delta t_{ij}) + e^{-2\mathrm{Re}(\mathbf{\Lambda})\Delta t_{ij}}\mathbf{\Lambda}_\Gamma$$

$$= \mathbf{\Lambda}_C^2 \mathbf{\Lambda}_\Omega \frac{\mathbf{I} - e^{2\mathrm{Re}(\mathbf{\Lambda})|\Delta t_{ij}|}}{-2\mathrm{Re}(\mathbf{\Lambda})} + e^{2\mathrm{Re}(\mathbf{\Lambda})|\Delta t_{ij}|}\mathbf{\Lambda}_\Gamma$$

(Stable dynamics attenuate the effect of measurement noise.) Finally, $\mathbf{P}_C(\Delta t_{ij}) = \mathbf{S}^{-*}\mathbf{P}_C^{\mathbf{S}}(\Delta t_{ij})\mathbf{S}^{-1}$, where $\mathbf{\Lambda}_{\mathbf{P}}^C(\Delta t_{ij}) = \mathbf{\Lambda}_{\mathbf{V}}^C(\Delta t_{ij})^{-1}$.

The MLE is now tractable:

$$\bar{\mathbf{z}}_i = \mathbf{S}\left(\sum_j \mathbf{\Lambda}_{\mathbf{P}}^C(\Delta t_{ij})\right)^{-1} \sum_j \mathbf{\Lambda}_{\mathbf{P}}^C(\Delta t_{ij})\mathbf{S}^{-1}\hat{\mathbf{z}}_i(t_j).$$

7.4 Accounting for Continuous-Time Measurements

In this section, we generalize the diagonalized expression for $\mathbf{V}(\Delta t)$ to account for the possibility of continuous-time measurements.

Recall that the total forward-propagated covariance $\mathbf{V}_F(\Delta t)$ is:

$$\mathbf{V}_F(\Delta t) = \int_0^{\Delta t} e^{\mathbf{A}(\Delta t - \tau)}\mathbf{\Omega}(\tau)e^{\mathbf{A}^T(\Delta t - \tau)}d\tau.$$

Using Leibniz's rule for differentiating an integral with respect to its upper limit:

$$\frac{d}{dt}\mathbf{V}_F(t) = \mathbf{\Omega}(t) + \int_0^t \frac{\partial}{\partial t}\left(e^{\mathbf{A}(t-\tau)}\mathbf{\Omega}(\tau)e^{\mathbf{A}^T(t-\tau)}\right)d\tau$$

$$= \int_0^t \left(\mathbf{A}e^{\mathbf{A}(t-\tau)}\mathbf{\Omega}(\tau)e^{\mathbf{A}^T(t-\tau)} + e^{\mathbf{A}(t-\tau)}\mathbf{\Omega}(\tau)\mathbf{A}^T e^{\mathbf{A}^T(t-\tau)}\right)d\tau + \mathbf{\Omega}(t)$$

Hence,

$$\dot{\mathbf{V}}_F(t) = \mathbf{A}\mathbf{V}_F(t) + \mathbf{V}_F(t)\mathbf{A}^T + \mathbf{\Omega}(t)$$

This is known as the differential Lyapunov equation.

In general, the derivative of a matrix inverse is $\dot{\mathbf{V}}_{\mathbf{F}}^{-1}(t) = -\mathbf{V}_F^{-1}(t)\dot{\mathbf{V}}_F(t)\mathbf{V}_F^{-1}(t)$, so:

$$\dot{\mathbf{P}}_F(t) = -\mathbf{P}_F(t)\dot{\mathbf{V}}_F(t)\mathbf{P}_F(t) = -\mathbf{P}_F(t)\left(\mathbf{A}\mathbf{V}_F(t) + \mathbf{V}_F(t)\mathbf{A}^T + \mathbf{\Omega}(t)\right)\mathbf{P}_F(t)$$

$$= -\mathbf{P}_F(t)\mathbf{A} - \mathbf{A}^T\mathbf{P}_F(t) - \mathbf{P}_F(t)\mathbf{\Omega}(t)\mathbf{P}_F(t)$$

If continuous-time measurements are incorporated, the covariance evolution is governed by the Riccati equation, which modifies the differential Lyapunov equation to include the measurement update:

$$\dot{\mathbf{V}}_F(t) = \mathbf{A}\mathbf{V}_F(t) + \mathbf{V}_F(t)\mathbf{A}^T + \mathbf{\Omega}(t) - \mathbf{V}_F(t)\mathbf{C}^T\mathbf{\Gamma}^{-1}(t)\mathbf{C}\mathbf{V}_F(t)$$

Generally, this does not have a closed-form solution. However, if we again assume that $\mathbf{A} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1}$, $\mathbf{C} = \mathbf{S}\mathbf{\Lambda}_C\mathbf{S}^{-1}$, $\mathbf{\Omega} = \mathbf{S}\mathbf{\Lambda}_\Omega\mathbf{S}^*$, and $\mathbf{\Gamma} = \mathbf{S}\mathbf{\Lambda}_\Gamma\mathbf{S}^*$, we can write down a closed-form solution. Left and right multiplying by $\mathbf{S}^{-1}$ and $\mathbf{S}^{-*}$, and defining $\tilde{\mathbf{V}}_F = \mathbf{S}^{-1}\mathbf{V}_F\mathbf{S}^{-*}$,

$$\dot{\tilde{\mathbf{V}}}_F(t) = \mathbf{\Lambda}\tilde{\mathbf{V}}_F(t) + \tilde{\mathbf{V}}_F(t)\mathbf{\Lambda}^* + \mathbf{\Lambda}_\Omega - \tilde{\mathbf{V}}_F(t)\mathbf{\Lambda}_C\mathbf{\Lambda}_\Gamma^{-1}(t)\mathbf{\Lambda}_C\tilde{\mathbf{V}}_F(t)$$

Note that if $\tilde{\mathbf{V}}_F(0)$ is diagonal, then $\tilde{\mathbf{V}}_F(\Delta t)$ remains diagonal for all Δt. That is, the matrix equations decouple into separate scalar differential equations for each diagonal element. The kth diagonal term is:

$$\dot{v}_k(t) = 2\mathrm{Re}(\lambda_i)v_k(t) + \lambda_{\Omega k} - \frac{\lambda_{Ck}^2}{\lambda_{\Gamma k}}v_k(t)^2$$

This is a scalar Ricatti equation with constant coefficients, which in general can be written as:

$$\dot{x}(t) = ax^2(t) + bx(t) + c$$

We can solve this using separation of variables:

$$\int \frac{dx}{ax^2 + bx + c} = \int dt$$

Hence,

$$\frac{2}{\sqrt{4ac-b^2}}\arctan\left(\frac{2ax+b}{\sqrt{4ac-b^2}}\right)+c_1=\Delta t$$

Solving for x:

$$\begin{aligned}x(\Delta t)&=\frac{1}{2a}\left(\left(\sqrt{4ac-b^2}\right)\tan\left(\frac{1}{2}\sqrt{4ac-b^2}(\Delta t-c_1)\right)-b\right)\\&=\frac{1}{2a}\left(\left(i\sqrt{b^2-4ac}\right)\tan\left(\frac{1}{2}i\sqrt{b^2-4ac}\,(\Delta t-c_1)\right)-b\right)\\&=-\frac{1}{2a}\left(\sqrt{b^2-4ac}\tanh\left(\frac{1}{2}\sqrt{b^2-4ac}\,(\Delta t-c_1)\right)+b\right)\end{aligned}$$

Plugging in $a=-\frac{\lambda_{Ck}^2}{\lambda_{\Gamma k}}$, $b=2\mathrm{Re}(\lambda_k)$, $c=\lambda_{\Omega k}$,

$$v_k(\Delta t_{ij})=\frac{\lambda_{\Gamma k}}{\lambda_{Ck}^2}\left(c_2\tanh\left(\frac{c_2}{2}\left(\Delta t_{ij}-c_1\right)\right)+\mathrm{Re}(\lambda_k)\right)$$

where $c_2=\sqrt{\mathrm{Re}(\lambda_k)^2+\frac{\lambda_{\Omega k}}{\lambda_{\Gamma k}}\lambda_{Ck}^2}$.

It remains to determine c_1 using the initial condition: $v_k(t_0)=0$:

$$\frac{\lambda_{\Gamma k}}{\lambda_{Ck}^2}\left(c_2\tanh\left(\frac{1}{2}c_2c_1\right)+\mathrm{Re}(\lambda_k)\right)=0$$

Letting $c_3=-\mathrm{Re}(\lambda_k)$, we obtain:

$$c_1=\frac{2}{c_2}\tanh^{-1}\left(\frac{c_3}{c_2}\right)=\frac{1}{c_2}\log\left(\frac{c_2+c_3}{c_2-c_3}\right)$$

As a check, setting the measurement term $\mathbf{\Lambda}_C\mathbf{\Lambda}_\Gamma^{-1}(t)\mathbf{\Lambda}_C=\mathbf{0}$, the above scalar Ricatti equation simplifies to:

$$\dot{x}(t)=bx(t)+c$$

which has solution:

$$x(t)=c_1e^{bt}-\frac{c}{b}$$

Plugging in $b=2\mathrm{Re}(\lambda_k)$, $c=\lambda_{\Omega k}$, this is:

$$v_k(\Delta t_{ij})=c_1e^{2\mathrm{Re}(\lambda_k)\Delta t_{ij}}-\frac{\lambda_{\Omega k}}{2\mathrm{Re}(\lambda_k)}$$

Using the initial condition ($v_k(0) = 0$):

$$v_k(\Delta t_{ij}) = \frac{\lambda_{\Omega k}}{2\mathrm{Re}(\lambda_k)} \left(e^{2\mathrm{Re}(\lambda_k)\Delta t_{ij}} - 1\right)$$

which is the same as our result above for $\mathbf{\Lambda_V}(\Delta t)$.

See also: (BBH18).

7.5 Extending the Expression for Propagated Covariance to Diagonalizable Plus Low Rank Matrices

Let us now relax the assumption of diagonalizable $\mathbf{A}$ to *Diagonalizable Plus Low-Rank* (DPLR), that is, suppose we can write $\mathbf{A}$ in the form:

$$\mathbf{A} = \mathbf{S}\left(\mathbf{\Lambda} - \mathbf{p}\mathbf{q}^*\right)\mathbf{S}^{-1}$$

where $\mathbf{p}$ and $\mathbf{q}$ are vectors. Substituting $\mathbf{A}$ into $\mathbf{V}_B(|\Delta t|)$:

$$\mathbf{V}_B(|\Delta t|) = \mathbf{S}\left(\int_0^{|\Delta t|} e^{-(\mathbf{\Lambda}-\mathbf{p}\mathbf{q}^*)\tau}\mathbf{\Lambda}_\Omega e^{-(\mathbf{\Lambda}-\mathbf{p}\mathbf{q}^*)^*\tau} d\tau\right)\mathbf{S}^*$$

According to the Baker-Campbell-Hausdorff (BCH) formula, $e^{\mathbf{A}}e^{\mathbf{B}} \approx e^{\mathbf{A}+\mathbf{B}+\frac{1}{2}[\mathbf{A},\mathbf{B}]+\cdots}$, where $[\mathbf{A},\mathbf{B}]$ is the commutator of $\mathbf{A}$ and $\mathbf{B}$. If $||\mathbf{\Lambda}|| \gg ||\mathbf{p}\mathbf{q}^*||$, then $\mathbf{A}$ is nearly diagonal, so $[\mathbf{A},\mathbf{A}^*]$ is small, and we can approximate the integrand as $e^{-2Re(\mathbf{\Lambda}-\mathbf{p}\mathbf{q}^*)\tau}$. Hence, for small $\mathbf{p}$ and $\mathbf{q}$,

$$\mathbf{V}_B(|\Delta t|) \approx \mathbf{S}\left(\mathbf{\Lambda}_\Omega \int_0^{|\Delta t|} e^{-2Re(\mathbf{\Lambda}-\mathbf{p}\mathbf{q}^*)\tau} d\tau\right)\mathbf{S}^*$$

$$= \mathbf{S}\left(-\frac{1}{2}\mathbf{\Lambda}_\Omega \left(Re(\mathbf{\Lambda} - \mathbf{p}\mathbf{q}^*)\right)^{-1}\left(e^{-2Re(\mathbf{\Lambda}-\mathbf{p}\mathbf{q}^*)|\Delta t|} - \mathbf{I}\right)\right)\mathbf{S}^*$$

Again assuming small $\mathbf{p}\mathbf{q}^*$,

$$\mathbf{V}_F(|\Delta t_{ij}|) = e^{\mathbf{A}|\Delta t_{ij}|}\mathbf{V}_B(|\Delta t_{ij}|)e^{\mathbf{A}^*|\Delta t_{ij}|} \approx \mathbf{S}e^{2Re(\mathbf{\Lambda}-\mathbf{p}\mathbf{q}^*)|\Delta t_{ij}|}\mathbf{S}^{-1}\mathbf{V}_B(|\Delta t_{ij})|$$

$$= \mathbf{S}\left(-\frac{1}{2}\mathbf{\Lambda}_\Omega \left(Re(\mathbf{\Lambda} - \mathbf{p}\mathbf{q}^*)\right)^{-1}\left(\mathbf{I} - e^{-2Re(\mathbf{\Lambda}-\mathbf{p}\mathbf{q}^*)|\Delta t|}\right)\right)\mathbf{S}^*$$

$$\approx \mathbf{S}\left(-\frac{1}{2}\mathbf{\Lambda}_\Omega \left(\mathrm{Re}(\mathbf{\Lambda} - \mathbf{p}\mathbf{q}^*)\right)^{-1}\left(\mathbf{I} - e^{-2Re(\mathbf{\Lambda})|\Delta t|}e^{2\mathrm{Re}(\mathbf{p}\mathbf{q}^*)|\Delta t|}\right)\right)\mathbf{S}^*$$

For small $\mathrm{Re}(\mathbf{p}\mathbf{q}^*)|\Delta t|$, we can approximate the matrix exponential as:

$$e^{2\mathrm{Re}(\mathbf{p}\mathbf{q}^*)|\Delta t|} \approx \mathbf{I} + 2\mathrm{Re}(\mathbf{p}\mathbf{q}^*)|\Delta t| + O(\|\mathbf{p}\mathbf{q}^*\|^2).$$

Also, if $\mathbf{p} = \mathbf{q}$ is real, then we can compute $\left(\mathrm{Re}(\mathbf{\Lambda}) - \mathbf{p}\mathbf{p}^T\right)^{-1}$ using the Sherman Morrison formula. Altogether, this gives an approximation for $\mathbf{V}_F$:

$$\mathbf{V}_F(|\Delta t|) \approx \mathbf{S}\left[\frac{\mathbf{\Lambda}_\Omega}{-2\mathrm{Re}(\mathbf{\Lambda})}\left(\mathbf{I} - \frac{\mathbf{p}(\boldsymbol{\lambda}\odot\mathbf{p})^T}{1+\boldsymbol{\lambda}^T\mathbf{p}^2}\right)\left(\mathbf{I} - e^{-2Re(\mathbf{\Lambda})|\Delta t|}(\mathbf{I} + 2\mathbf{p}\mathbf{p}^T|\Delta t|)\right)\right]\mathbf{S}^*$$

We do not use this expression here, as it is unwieldy and only valid when $\mathbf{p}$ and Δt are small.

7.6 Estimating the Covariance and Precision Matrices from Data

Recall that:

$$\mathbf{r}_i(t_j) := \hat{\mathbf{x}}_i(t_j) - \mathbf{x}(t_i) = \int_{t_i}^{t_j} e^{-\mathbf{A}\tau}\boldsymbol{\eta}(\tau)d\tau$$

Since the residuals are sums of normally-distributed random variables, they are also normally distributed. If the residuals are approximately zero-mean and uncorrelated with the model error, then their magnitude encodes information about under- or over-confidence in $\mathbf{V}_C(t)$. In particular,

$$\mathbb{E}[\mathbf{r}_i(t_j)] = \int_{t_i}^{t_j} e^{-\mathbf{A}\tau}\mathbb{E}[\boldsymbol{\eta}(\tau)]d\tau = \mathbf{0}.$$

And:

$$\mathbf{r}_i(t_j)\mathbf{r}_i(t_j)^T = \left(\int_{t_i}^{t_j} e^{-\mathbf{A}\tau}\boldsymbol{\eta}(\tau)d\tau\right)\left(\int_{t_i}^{t_j} e^{-\mathbf{A}\tau}\boldsymbol{\eta}(\tau)d\tau\right)^T$$

$$= \int_{t_i}^{t_j} \int_{t_i}^{t_j} e^{-\mathbf{A}\tau_1} \boldsymbol{\eta}(\tau_1) \boldsymbol{\eta}^T(\tau_2) e^{-\mathbf{A}^T\tau_2} d\tau_1 d\tau_2$$
$$= \int_{t_i}^{t_j} \int_{t_i}^{t_j} e^{-\mathbf{A}\tau_1} \boldsymbol{\eta}(\tau_1) \boldsymbol{\eta}^T(\tau_2) e^{-\mathbf{A}^T\tau_2} d\tau_1 d\tau_2$$

Taking the expectation,

$$\mathbb{E}\left[\mathbf{r}_i(t_j)\mathbf{r}_i(t_j)^T\right] = \int_{t_i}^{t_j} \int_{t_i}^{t_j} e^{-\mathbf{A}\tau_1} \mathbb{E}[\boldsymbol{\eta}(\tau_1)\boldsymbol{\eta}^T(\tau_2)] e^{-\mathbf{A}^T\tau_2} d\tau_1 d\tau_2$$

$$= \int_{t_i}^{t_j} \int_{t_i}^{t_j} e^{-\mathbf{A}\tau_1} \delta(\tau_1 - \tau_2) \boldsymbol{\Omega}(\tau_1) e^{-\mathbf{A}^T\tau_2} d\tau_1 d\tau_2$$

$$= \int_{t_i}^{t_j} e^{-\mathbf{A}\tau} \boldsymbol{\Omega}(\tau) e^{-\mathbf{A}^T\tau} d\tau$$

$$= \mathbf{V}(\Delta t_{ij})$$

Hence, $\mathbf{r}_i(t_j) \sim \mathcal{N}(\mathbf{0}, \mathbf{V}(\Delta t_{ij}))$.

We can also define a measurement residual:

$$\mathbf{r}_{\mathbf{z}i}(t_j) := \hat{\mathbf{z}}_i(t_j) - \mathbf{z}(t_i) = \mathbf{C}\hat{\mathbf{x}}(t_j) + \mathbf{Q}(\Delta t_{ij})\boldsymbol{\gamma}(t_j) - (\mathbf{C}\mathbf{x}(t_i) + \boldsymbol{\gamma}(t_i))$$

$$= \mathbf{C}(\hat{\mathbf{x}}(t_j) - \mathbf{x}(t_i)) + \mathbf{Q}(\Delta t_{ij})\boldsymbol{\gamma}(t_j) - \boldsymbol{\gamma}(t_i)$$

$$= \mathbf{C}\mathbf{r}_i(t_j) + \boldsymbol{\zeta}_i(t_j)$$

where $\boldsymbol{\zeta}_i(t_j) = \mathbf{Q}(\Delta t_{ij})\boldsymbol{\gamma}(t_j) - \boldsymbol{\gamma}(t_i)$ is normally distributed:

$$\boldsymbol{\zeta}_i(t_j) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}(\Delta t_{ij}))\boldsymbol{\Gamma}\mathbf{Q}^T(\Delta t_{ij}) + \boldsymbol{\Gamma}).$$

Hence,

$$\mathbb{E}[\mathbf{r}_{\mathbf{z}i}(t)] = \mathbf{C}\mathbb{E}[\mathbf{r}_i(t_j)] + \mathbb{E}[\boldsymbol{\zeta}_i(t_j)] = \mathbf{0}$$

The expectation of the outer product becomes:

$$\mathbb{E}\left[\mathbf{r}_{\mathbf{z}i}(t_j)\mathbf{r}_{\mathbf{z}i}^T(t_j)\right] = \mathbf{C}\mathbf{V}(\Delta t_{ij})\mathbf{C}^T + \mathbf{Q}(\Delta t_{ij})\boldsymbol{\Gamma}\mathbf{Q}(\Delta t_{ij}) + \boldsymbol{\Gamma}$$

$$= \mathbf{V}^C(\Delta t_{ij}) + \mathbf{\Gamma}$$

So,

$$\mathbf{r}_{\mathbf{z}i}(t_j) \sim \mathcal{N}(\mathbf{0}, \mathbf{V}^C(\Delta t_{ij}) + \mathbf{\Gamma})$$

7.7 Bayesian Residual-based Adaption of the Covariance and Precision Matrices

Since $\mathbb{E}[\mathbf{r}\mathbf{r}^T] = \mathbf{V}$, we can use the residuals to update our estimate of $\mathbf{V}$.

From the above, $\mathbf{r} \sim \mathcal{N}(0, \mathbf{V})$. Given a prior $\mathbf{V}$, which we will call $\mathbf{V}'$, our goal is to update our estimate of $\mathbf{V}$ from a single measurement of the residual $\mathbf{r}$.
The likelihood of a single observation of the residual $\mathbf{r}$ given $\mathbf{V}$ is:

$$p(\mathbf{r}|\mathbf{V}) = \frac{|\mathbf{V}|^{-1/2}}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\mathbf{r}^T\mathbf{V}^{-1}\mathbf{r}\right).$$

The quantity $\mathbf{r}^T\mathbf{V}^{-1}\mathbf{r}$ is the squared *Mahalanobis distance*, which is a measure of the distance between a vector $\mathbf{r}$ and a distribution $\mathcal{N}(0, \mathbf{V})$.

The maximum entropy prior for a random positive semidefinite matrix $\mathbf{V}$ is the *inverse Wishart* distribution, a multi-variate generalization of the inverse-Gamma distribution. (Zha21)

$$p(\mathbf{V}) = \mathcal{IW}(\mathbf{S}_r, m, d) = \frac{|\mathbf{S}|^{m/2}|\mathbf{V}|^{-(m+d+1)/2} \exp\left[-\text{tr}(\mathbf{S}\mathbf{V}^{-1})/2\right]}{2^{md/2}\Gamma_d(m/2)}$$

where m is the degrees of freedom parameter (related to prior confidence in $\mathbf{V}$), $\mathbf{S}$ is the scale matrix (which represents prior information about $\mathbf{V}$), and $\Gamma_d(\cdot)$ is the multivariate gamma function.

By Bayes' rule, the posterior distribution of $\mathbf{V}$ given $\mathbf{r}$ is:

$$p(\mathbf{V}|\mathbf{r}) \propto p(\mathbf{r}|\mathbf{V})p(\mathbf{V}).$$

Substituting the likelihood and prior distribution:

$$p(\mathbf{V}|\mathbf{r}) \propto |\mathbf{S}|^{m/2}|\mathbf{V}|^{-(m+d+1)/2}|\mathbf{V}|^{-1/2}\exp\left(-\frac{1}{2}\mathbf{r}^T\mathbf{V}^{-1}\mathbf{r}\right)\exp\left(-\frac{1}{2}\text{tr}(\mathbf{S}\mathbf{V}^{-1})\right)$$

$$\propto |\mathbf{V}|^{-(m+1+d+1)/2}\exp\left(-\frac{1}{2}\text{tr}\left((\mathbf{S}+\mathbf{r}\mathbf{r}^T)\mathbf{V}^{-1}\right)\right)$$

This is the kernel of another inverse-Wishart distribution with updated parameters:

$$p(\mathbf{V}|\mathbf{r}) = \mathcal{IW}(\mathbf{S}+\mathbf{r}\mathbf{r}^T, m+1)$$

Substituting the prior, the mean of the posterior Wishart distribution is given by:

$$\tilde{\mathbf{V}} := \mathbb{E}[\mathbf{V}|\mathbf{r}] = \nu\left(\mathbf{V}' + \mathbf{r}\mathbf{r}^T\right)$$

where $\nu := \frac{1}{m-d}$. We can approximate the posterior precision matrix as:

$$\tilde{\mathbf{P}} := \mathbb{E}[\mathbf{V}^{-1}|\mathbf{r}] \approx \mathbb{E}\left[\mathbf{V}|\mathbf{r}\right]^{-1} = \frac{1}{\nu}\left(\mathbf{V}' + \mathbf{r}\mathbf{r}^T\right)^{-1}$$

This is a rank-1 update of the inverse, which we can compute with the Sherman-Morrison formula:

$$\tilde{\mathbf{P}} = \frac{1}{\nu}\left(\mathbf{V}' + \mathbf{r}\mathbf{r}^T\right)^{-1} = \frac{1}{\nu}\left(\mathbf{P}' - \frac{\mathbf{P}'\mathbf{r}\mathbf{r}^T\mathbf{P}'}{1+\mathbf{r}^T\mathbf{P}'\mathbf{r}}\right)$$

Since matrix inversion is convex on the space of positive semidefinite matrices, $\mathbb{E}[\mathbf{V}^{-1}|\mathbf{r}] \succeq (\mathbb{E}[\mathbf{V}|\mathbf{r}])^{-1}$ by Jensen's inequality, so this approximation is a lower bound on the expected precision matrix. While not exact, it is a tractable, conservative approximation (in the sense that it overestimates uncertainty), easy to implement via Sherman-Morrison, and increasingly accurate in the low-variance limit.

We can generalize the above to include the measurement model. Recall that:

$$\mathbb{E}\left[\mathbf{r_z}{\mathbf{r_z}}^T\right] = \mathbf{V}^C + \mathbf{\Gamma} := \mathbf{V}^C_\Gamma$$

Then, following the above:

$$\mathbb{E}[\mathbf{V}^C_\Gamma|\mathbf{r_z}] = \nu\left(\mathbf{V}^C_\Gamma + \mathbf{r_z}{\mathbf{r_z}}^T\right)$$

Hence,

$$\mathbb{E}[\mathbf{V}^C + \mathbf{\Gamma}|\mathbf{r_z}] = \nu\left(\mathbf{V}^{C'} + \mathbf{r_z}{\mathbf{r_z}}^T\right) + \nu\mathbf{\Gamma}$$

So,

$$\mathbb{E}[\mathbf{V}^C|\mathbf{r_z}] = \nu\left(\mathbf{V}^{C'} + \mathbf{r_z}{\mathbf{r_z}}^T\right) + \nu\mathbf{\Gamma} - \mathbb{E}[\mathbf{\Gamma}|\mathbf{r_z}]$$

If we assume that none of our model mis-specification is due to $\mathbf{\Gamma}$, we can take $\mathbb{E}[\mathbf{\Gamma}|\mathbf{r_z}] = \mathbf{\Gamma}$. This is a conservative assumption in that it avoids underestimating total uncertainty or misattributing signal to noise.

Using the maximum-entropy value of $\nu = 1$:

$$\tilde{\mathbf{V}}^C := \mathbb{E}[\mathbf{V}^C|\mathbf{r_z}] = \mathbf{V}^{C'} + \mathbf{r}_z\mathbf{r}_z^T$$

We can then form a point estimate of the precision matrix:

$$\tilde{\mathbf{P}}^C := \mathbb{E}[\mathbf{P}^C|\mathbf{r_z}] \approx \mathbf{P}^{C'} - \frac{\mathbf{P}^{C'}\mathbf{r_z}{\mathbf{r_z}}^T\mathbf{P}^{C'}}{1 + {\mathbf{r_z}}^T\mathbf{P}^{C'}\mathbf{r_z}}.$$

7.8 Plugging in the Rank-1 Precision Matrix Update

In what follows, we will denote $\mathbf{x}_i := \mathbf{x}(t_i)$, $\hat{\mathbf{x}}_{ij} := \hat{\mathbf{x}}_i(t_j)$, $\mathbf{P}_{ij} := \mathbf{P}(\Delta t_{ij})$, etc.

Plugging in the rank-1 Bayesian update of the precision matrices, our MLE becomes:

$$\bar{\mathbf{z}}_i = \left(\sum_j \tilde{\mathbf{P}}_{ij}^C\right)^{-1} \sum_j \tilde{\mathbf{P}}_{ij}^C \hat{\mathbf{z}}_{ij} = \mathbf{z}_i + \left(\sum_j \tilde{\mathbf{P}}_{ij}^C\right)^{-1} \sum_j \tilde{\mathbf{P}}_{ij}^C \mathbf{r}_{\mathbf{z}ij}$$

where (suppressing the ij indices):

$$\begin{aligned}\tilde{\mathbf{P}}^C &= \mathbf{P}^C - \frac{\mathbf{P}^C \mathbf{r}_\mathbf{z} \mathbf{r}_\mathbf{z}^* \mathbf{P}^C}{1 + \mathbf{r}_\mathbf{z}^* \mathbf{P}^C \mathbf{r}_\mathbf{z}} \\ &= \left(\mathbf{I} - \frac{\mathbf{P}^C \mathbf{r}_\mathbf{z} {\mathbf{r}_\mathbf{z}}^*}{1 + {\mathbf{r}_\mathbf{z}}^* \mathbf{P}^C \mathbf{r}_\mathbf{z}}\right) \mathbf{P}^C \\ &= (\mathbf{I} + \mathbf{R}) \frac{\mathbf{P}^C}{1 + {\mathbf{r}_\mathbf{z}}^* \mathbf{P}^C \mathbf{r}_\mathbf{z}}\end{aligned}$$

where:

$$\mathbf{R} = {\mathbf{r}_\mathbf{z}}^* \mathbf{P}^C \mathbf{r}_\mathbf{z} \mathbf{I} - \mathbf{P}^C \mathbf{r}_\mathbf{z} {\mathbf{r}_\mathbf{z}}^*$$

But $\mathbf{r}_\mathbf{z}$ is in the null space of the $\mathbf{R}$ term:

$$\mathbf{R}\mathbf{P}^C \mathbf{r}_\mathbf{z} = \left({\mathbf{r}_\mathbf{z}}^* \mathbf{P}^C \mathbf{r}_\mathbf{z}\right) \mathbf{P}^C \mathbf{r}_\mathbf{z} - \mathbf{P}^C \mathbf{r}_\mathbf{z} \left({\mathbf{r}_\mathbf{z}}^* \mathbf{P}^C \mathbf{r}_\mathbf{z}\right) = \mathbf{0}$$

So the MLE simplifies to:

$$\bar{\mathbf{z}}_i = \mathbf{z}_i + \left(\sum_j (\mathbf{I} + \mathbf{R}_{ij}) \frac{\mathbf{P}_{ij}^C}{1 + \mathbf{r}_{\mathbf{z}ij}^* \mathbf{P}_{ij}^C \mathbf{r}_{\mathbf{z}ij}}\right)^{-1} \sum_j \frac{\mathbf{P}_{ij}^C}{1 + \mathbf{r}_{\mathbf{z}ij}^* \mathbf{P}_{ij}^C \mathbf{r}_{\mathbf{z}ij}} \mathbf{r}_{\mathbf{z}ij}.$$

Let $\mathbf{z}_{\mathbf{s}i} := \mathbf{S}^{-1}\mathbf{z}_i$ and $\hat{\mathbf{z}}_{\mathbf{s}ij} = \mathbf{S}^{-1}\hat{\mathbf{z}}_{ij}$.

And let: $\mathbf{r}_{\mathbf{zs}ij} := \mathbf{S}^{-1}\mathbf{r}_{\mathbf{z}ij} = \mathbf{S}^{-1}\left(\hat{\mathbf{z}}_{ij} - \mathbf{z}_i\right) = \hat{\mathbf{z}}_{\mathbf{s}ij} - \mathbf{z}_{\mathbf{s}i}$.

Then: ${\mathbf{r}_\mathbf{z}}^* \mathbf{P}^C \mathbf{r}_\mathbf{z} = \mathbf{r}_\mathbf{zs}^* \mathbf{\Lambda}_\mathbf{P}^C \mathbf{r}_\mathbf{zs}$.

And: $\mathbf{R} = \mathbf{S}^{-*} \mathbf{\Lambda}_\mathbf{R} \mathbf{S}^*$ where $\mathbf{\Lambda}_\mathbf{R} = \mathbf{r}_\mathbf{zs}^* \mathbf{\Lambda}_\mathbf{P}^C \mathbf{r}_\mathbf{zs} \mathbf{I} - \mathbf{\Lambda}_\mathbf{P}^C \mathbf{r}_\mathbf{zs} \mathbf{r}_\mathbf{zs}^*$.

So the MLE becomes:

$$\bar{\mathbf{z}}_{\mathbf{s}i} = \mathbf{z}_{\mathbf{s}i} + \left(\sum_j (\mathbf{I} + \mathbf{\Lambda}_{\mathbf{R}ij}) \frac{\mathbf{\Lambda}_{\mathbf{P}ij}^C}{1 + \mathbf{r}_{\mathbf{zs}ij}^* \mathbf{\Lambda}_{\mathbf{P}ij}^C \mathbf{r}_{\mathbf{zs}ij}}\right)^{-1} \sum_j \frac{\mathbf{\Lambda}_{\mathbf{P}ij}^C}{1 + \mathbf{r}_{\mathbf{zs}ij}^* \mathbf{\Lambda}_{\mathbf{P}ij}^C \mathbf{r}_{\mathbf{zs}ij}} \mathbf{r}_{\mathbf{zs}ij}.$$

Unfortunately, the inverse term is still intractable as $\mathbf{\Lambda}_{\mathbf{R}ij}$ is not diagonal.

However, observe that the expectation of $\mathbf{R}_{ij}$ is:

$$\begin{aligned}
\mathbb{E}[\mathbf{R}_{ij}] &= \mathbb{E}\left[\mathbf{r}_{\mathbf{z}ij}^{*}\mathbf{P}_{ij}^{C}\mathbf{r}_{\mathbf{z}ij}\mathbf{I} - \mathbf{P}^{C}\mathbf{r}_{\mathbf{z}ij}\mathbf{r}_{\mathbf{z}ij}^{*}\right] \\
&= \text{tr}\left(\mathbf{P}_{ij}^{C}\mathbb{E}\left[\mathbf{r}_{\mathbf{z}ij}\mathbf{r}_{\mathbf{z}ij}^{*}\right]\right) - \mathbf{P}_{ij}^{C}\mathbb{E}\left[\mathbf{r}_{\mathbf{z}ij}\mathbf{r}_{\mathbf{z}ij}^{*}\right] \\
&= \text{tr}\left(\mathbf{P}_{ij}^{C}\left(\mathbf{V}_{ij}^{C} + \mathbf{\Gamma}\right)\right)\mathbf{I} - \mathbf{P}_{ij}^{C}\left(\mathbf{V}_{ij}^{C} + \mathbf{\Gamma}\right) \\
&= \text{tr}\left(\mathbf{I} + \mathbf{P}_{ij}^{C}\mathbf{\Gamma}\right)\mathbf{I} - \left(\mathbf{I} + \mathbf{P}_{ij}^{C}\mathbf{\Gamma}\right) \\
&= \mathbf{S}^{-*}\left(\text{tr}\left(\mathbf{I} + \mathbf{\Lambda}_{\mathbf{P}ij}^{C}\mathbf{\Lambda}_{\Gamma}\right)\mathbf{I} - \left(\mathbf{I} + \mathbf{\Lambda}_{\mathbf{P}ij}^{C}\mathbf{\Lambda}_{\Gamma}\right)\right)\mathbf{S}^{*}
\end{aligned}$$

So,

$$\begin{aligned}
\bar{\mathbf{\Lambda}}_{\mathbf{R}ij} := \mathbb{E}\left[\mathbf{\Lambda}_{\mathbf{R}ij}\right] &= \text{tr}\left(\mathbf{I} + \mathbf{\Lambda}_{\mathbf{P}ij}^{C}\mathbf{\Lambda}_{\Gamma}\right)\mathbf{I} - \left(\mathbf{I} + \mathbf{\Lambda}_{\mathbf{P}ij}^{C}\mathbf{\Lambda}_{\Gamma}\right) \\
&= \text{tr}\left(\mathbf{I} + \mathbf{\Delta}_{j}\right)\mathbf{I} - \left(\mathbf{I} + \mathbf{\Delta}_{j}\right)
\end{aligned}$$

where $\mathbf{\Delta}_{j} = \left(\mathbf{\Lambda}_{C}^{2}\mathbf{\Lambda}_{\Omega}\frac{\mathbf{I} - e^{2\text{Re}(\mathbf{\Lambda})|\Delta t_{ij}|}}{-2\text{Re}(\mathbf{\Lambda})} + e^{2\text{Re}(\mathbf{\Lambda})|\Delta t_{ij}|}\mathbf{\Lambda}_{\Gamma}\right)^{-1}\mathbf{\Lambda}_{\Gamma}$ is a diagonal, PD matrix. In particular, $\mathbf{\Delta}_{1} = \mathbf{I}$, $\mathbf{\Delta}_{\infty} = \frac{2|\text{Re}(\mathbf{\Lambda})|\mathbf{\Lambda}_{\Gamma}}{\mathbf{\Lambda}_{C}^{2}\mathbf{\Lambda}_{\Omega}}$, and $\min(\mathbf{\Delta}_{1}, \mathbf{\Delta}_{\infty}) < \mathbf{\Delta}_{j} < \max(\mathbf{\Delta}_{1}, \mathbf{\Delta}_{\infty})\forall j$. (So, if the measurement and process noise are on the same scale, $\mathbf{\Delta}_{j} \approx \mathbf{I}, \forall j$), and $\bar{\mathbf{\Lambda}}_{\mathbf{R}ij} \approx \alpha_{j}\mathbf{I}$.

If we could approximate $\mathbf{\Lambda}_{\mathbf{R}ij}$ with its expectation, the MLE again becomes tractable:

$$\bar{\mathbf{z}}_{\mathbf{s}i} \approx \mathbf{z}_{\mathbf{s}i} + \left(\sum_{j}\left(\mathbf{I} + \bar{\mathbf{\Lambda}}_{\mathbf{R}ij}\right)\frac{\mathbf{\Lambda}_{\mathbf{P}ij}^{C}}{1 + \mathbf{r}_{\mathbf{zs}ij}^{*}\mathbf{\Lambda}_{\mathbf{P}ij}^{C}\mathbf{r}_{\mathbf{zs}ij}}\right)^{-1}\sum_{j}\frac{\mathbf{\Lambda}_{\mathbf{P}ij}^{C}}{1 + \mathbf{r}_{\mathbf{zs}ij}^{*}\mathbf{\Lambda}_{\mathbf{P}ij}^{C}\mathbf{r}_{\mathbf{zs}ij}}\mathbf{r}_{\mathbf{zs}ij}$$

7.9 Decomposing the Corrected MLE Loss

We now provide a justification for the above approximation of $\mathbf{\Lambda}_{\mathbf{R}ij}$ with its expectation.

Let $\mathbf{e}_{\mathbf{z}ij} := \hat{\mathbf{z}}_{ij} - \mathbf{z} = (\hat{\mathbf{z}}_{ij} - \mathbf{z}_i) + (\mathbf{z}_i - \mathbf{z}) = \mathbf{r}_{ij} + \boldsymbol{\delta}_{\mathbf{z}i}$, where $\boldsymbol{\delta}_{\mathbf{z}i} := \mathbf{z}_i - \mathbf{z}$.

The MLE is the solution to the following minimization problem:

$$\bar{\mathbf{z}}_i = \arg\min_{\mathbf{z}} \sum_j \|\hat{\mathbf{z}}_{ij} - \mathbf{z}\|^2_{\tilde{\mathbf{P}}^C_{ij}} = \arg\min_{\mathbf{z}} \sum_j \mathbf{e}^*_{\mathbf{z}ij} \tilde{\mathbf{P}}^C_{ij} \mathbf{e}_{\mathbf{z}ij},$$

where the full corrected precision matrix is given by

$$\tilde{\mathbf{P}}^C_{ij} = (\mathbf{I} + \mathbf{R}_{ij})\, \tilde{\mathbf{P}}^{approx}_{ij},$$

where $\tilde{\mathbf{P}}^{approx}_{ij} = \frac{\mathbf{P}^C_{ij}}{1+\mathbf{r}^*_{ij}\mathbf{P}^C_{ij}\mathbf{r}_{ij}}$ and with residuals $\mathbf{r}_{ij} = \hat{\mathbf{z}}_{ij} - \mathbf{z}_i$, and

$$\mathbf{R}_{ij} = \mathbf{r}^*_{ij}\mathbf{P}^C_{ij}\mathbf{r}_{ij} \cdot \mathbf{I} - \mathbf{P}^C_{ij}\mathbf{r}_{ij}\mathbf{r}^*_{ij}.$$

We are minimizing $\sum_j \mathcal{L}_j$ where

$$\mathcal{L}_j = \mathbf{e}^*_{\mathbf{z}ij}\tilde{\mathbf{P}}^C_{ij}\mathbf{e}_{\mathbf{z}ij} = (\mathbf{r}_{ij} + \boldsymbol{\delta}_{\mathbf{z}i})^* (\mathbf{I} + \mathbf{R}_{ij})\, \tilde{\mathbf{P}}^{approx}_{ij}(\mathbf{r}_{ij} + \boldsymbol{\delta}_{\mathbf{z}i})$$

$$= \mathbf{r}^*_{ij} (\mathbf{I} + \mathbf{R}_{ij})\, \tilde{\mathbf{P}}^{approx}_{ij}\mathbf{r}_{ij} + \mathbf{r}^*_{ij} (\mathbf{I} + \mathbf{R}_{ij})\, \tilde{\mathbf{P}}^{approx}_{ij}\boldsymbol{\delta}_{\mathbf{z}i}$$
$$+\boldsymbol{\delta}^*_{\mathbf{z}i} (\mathbf{I} + \mathbf{R}_{ij})\, \tilde{\mathbf{P}}^{approx}_{ij}\mathbf{r}_{ij} + \boldsymbol{\delta}^*_{\mathbf{z}i} (\mathbf{I} + \mathbf{R}_{ij})\, \tilde{\mathbf{P}}^{approx}_{ij}\boldsymbol{\delta}_{\mathbf{z}i}$$

But:

$$\mathbf{r}^*_{\mathbf{z}ij}\mathbf{R}_{ij} = (\mathbf{r}^*_{\mathbf{z}ij}\mathbf{P}^C_{ij}\mathbf{r}_{\mathbf{z}ij})\mathbf{r}^*_{\mathbf{z}ij} - \mathbf{r}^*_{\mathbf{z}ij}(\mathbf{P}^C_{ij}\mathbf{r}_{\mathbf{z}ij}\mathbf{r}^*_{\mathbf{z}ij}) = \mathbf{0}$$

So this is:

$$\mathcal{L}_j = \mathbf{r}^*_{ij}\tilde{\mathbf{P}}^{approx}_{ij}\mathbf{r}_{ij} + \mathbf{r}^*_{ij}\tilde{\mathbf{P}}^{approx}_{ij}\boldsymbol{\delta}_{\mathbf{z}i} + \boldsymbol{\delta}^*_{\mathbf{z}i}\tilde{\mathbf{P}}^{approx}_{ij}\mathbf{r}_{ij} + \boldsymbol{\delta}^*_{\mathbf{z}i} (\mathbf{I} + \mathbf{R}_{ij})\, \tilde{\mathbf{P}}^{approx}_{ij}\boldsymbol{\delta}_{\mathbf{z}i}$$

$$= \mathbf{e}^*_{\mathbf{z}ij}\tilde{\mathbf{P}}^{approx}_{ij}\mathbf{e}_{\mathbf{z}ij} + \boldsymbol{\delta}^*_{\mathbf{z}i}\mathbf{R}_{ij}\tilde{\mathbf{P}}^{approx}_{ij}\boldsymbol{\delta}_{\mathbf{z}i}$$

So we can rewrite the loss as:

$$\mathcal{L} = \sum_j \|\hat{\mathbf{z}}_{ij} - \mathbf{z}\|^2_{\tilde{\mathbf{P}}^{approx}_{ij}} + \|\mathbf{z}_i - \mathbf{z}\|^2_{\mathbf{R}_{ij}\tilde{\mathbf{P}}^{approx}_{ij}}$$

That is, the loss decomposes into a weighted squared error between the denoised predictions and the estimate, plus a regularization term based on how far the current estimate is from the "central" anchor point $\mathbf{z}_i$. Since $\mathbb{E}\left[\mathbf{\Lambda}_{\mathbf{R}ij}\right] \approx \alpha_j \mathbf{I}$, we can approximate this in the eigenbasis as:

$$\mathcal{L} = \sum_j \left\|\hat{\mathbf{z}}_{ij} - \mathbf{z}\right\|^2_{\tilde{\mathbf{\Lambda}}^{approx}_{Pij}} + \left\|\mathbf{z}_i - \mathbf{z}\right\|^2_{\alpha_j \tilde{\mathbf{\Lambda}}^{approx}_{Pij}}$$

Replacing this with a simpler regularizer:

$$\mathcal{L} = \sum_j \left\|\hat{\mathbf{z}}_{ij} - \mathbf{z}\right\|^2_{\tilde{\mathbf{\Lambda}}^{approx}_{Pij}} + \left\|\mathbf{z}_i - \mathbf{z}\right\|^2_{\alpha \tilde{\mathbf{\Lambda}}^{approx}_{Pij}}$$

we again obtain an affine combination of the current estimate and weighted average.

7.10 Convergence of the Iteration

The above formulation relies on two assumptions:

1. $\mathbf{z}^{(k)}$ converges to nearly $\mathbf{z}^C$ in expectation when $\mathbf{S}, \mathbf{\Lambda}, \mathbf{\Lambda}_\Omega, \mathbf{\Lambda}_\Gamma$ are fixed to approximately the correct values, i.e. $\mathbb{E}\left[\hat{\mathbf{z}}^{(k)}_{ii}\right] \to \mathbf{z}^C_i \quad \text{as } k \to \infty$.
2. $\mathbb{E}\left[\sum_i ||\mathbf{z}^{\text{pred}}_i - \mathbf{z}_{i+1}||^2_2\right]$ is minimized by nearly the correct $\mathbf{S}, \mathbf{\Lambda}, \mathbf{\Lambda}_\Omega, \mathbf{\Lambda}_\Gamma$, where $\mathbf{z}_{i+1} \sim \mathcal{N}(\mathbf{S}e^{\mathbf{\Lambda}(t_{i+1}-t_i)}\mathbf{z}^C_{\mathbf{s}i}, \mathbf{V}_{i+1,i})$.

We here provide informal arguments for why these conditions can be expected to hold.

Convergence of State Estimates:

The weights $w^{(k)}_{ij}$ are a non-linear function of the residuals. Due to this non-linearity and data dependence, the expected value of the weighted average, $\mathbb{E}[\hat{\mathbf{z}}^{\text{avg}(k)}_{\mathbf{s}i}]$, may not be

exactly $\mathbf{z}_i^C$. This introduces a small finite-sample bias, $\mathbf{b}_i^{(k)}$: $\mathbb{E}[\hat{\mathbf{z}}_{\mathbf{s}i}^{\text{avg}(k)}] = \mathbf{z}_i^C + \mathbf{b}_i^{(k)}$.

Let's define the expected error at iteration k as $\mathbf{e}^{(k)} := \mathbb{E}[\hat{\mathbf{z}}_{\mathbf{s}ii}^{(k)}] - \mathbf{z}_i^C$. Taking expectations of both sides of the update rule for $\hat{\mathbf{z}}_{\mathbf{s}ii}^{(k+1)}$:

$$\left(\mathbf{e}^{(k+1)} + \mathbf{z}_i^C\right) = (\mathbf{I} - \mathbf{\Delta_R})\left(\mathbf{e}^{(k)} + \mathbf{z}_i^C\right) + \mathbf{\Delta_R}\left(\mathbf{z}_i^C + \mathbf{b}_i^{(k)}\right)$$

Which simplifies to:

$$\mathbf{e}^{(k+1)} = (\mathbf{I} - \mathbf{\Delta R})\mathbf{e}^{(k)} + \mathbf{\Delta R}\mathbf{b}_i^{(k)}$$

Since the spectral radius of $(\mathbf{I} - \mathbf{\Delta_R})$ is strictly less than 1 (as $\mathbf{\Delta_R}$ has entries between 0 and 1), if the bias term $\mathbf{b}_i^{(k)}$ converges to $\mathbf{0}$ as $k \to \infty$, then the expected error $\mathbf{e}^{(k)}$ also converges to $\mathbf{0}$. If $\mathbf{b}_i^{(k)}$ converges to a small, non-zero constant vector $\mathbf{b}_i$, then $\mathbf{e}^{(k)}$ converges to a steady-state error $\mathbf{b}_i$.

The core question for consistency is whether this bias $\mathbf{b}_i^{(k)}$ indeed vanishes asymptotically. In the context of robust M-estimators, which our adaptive weighting scheme resembles, if the underlying noise distribution is symmetric (like Gaussian noise) and the weighting function is symmetric (which $w_{ij}^{(k)}$ is, being based on a squared distance), it is generally expected that the estimator will be asymptotically unbiased. As we repeat the iteration over many batches of training data, the finite-sample bias $\mathbf{b}_i^{(k)}$ should indeed be expected to vanish as k grows very large. Combined with the decreasing variance inherent in repeated averaging, the estimator $\hat{\mathbf{z}}_{\mathbf{s}ii}^{(k)}$ should therefore be consistent for $\mathbf{z}_i^C$.

Consistency of Learned Precision Matrices:

A critical point is that the same set of parameters $\mathbf{S}, \mathbf{\Lambda}, \mathbf{\Lambda}_\Omega, \mathbf{\Lambda}_\Gamma$ parameterizes both the precision matrices (for estimation) and the dynamic evolution (for prediction). This struc-

tural constraint enforces consistency between the linear approximations used for both tasks.

However, we can ask: Given fixed $\hat{\mathbf{z}}_{\mathbf{s}ij}$ sampled from the correct distribution, i.e. $\hat{\mathbf{z}}_i(t_j) \sim \mathcal{N}\left(\mathbf{z}^C(t_i), \mathbf{V}^C(\Delta t_{ij})\right)$, is the expected error $\mathbb{E}\left[\sum_i ||\mathbf{z}_i^{\text{pred}} - \mathbf{z}_{i+1}||_2^2\right]$, where $\mathbf{z}_{i+1} \sim \mathcal{N}(\mathbf{S}e^{\mathbf{\Lambda}(t_{i+1}-t_i)}\mathbf{z}_{\mathbf{s}i}^C, \mathbf{V}_{i+1,i})$, approximately minimized by the true precision matrix $\mathbf{\Lambda}_{Pij}^C$?

Although the optimal weighting matrix $\mathbf{\Lambda}_{ij}^*$ that minimizes the expected squared error of the robust weighted estimator with data-dependent weights generally differs from the true inverse covariance matrix $\mathbf{\Lambda}_{Vij}^{-1}$, it is unlikely to deviate significantly from it. This intuition stems from several key observations: first, the samples $\hat{\mathbf{z}}_{ij}$ are drawn from a distribution with covariance $\mathbf{\Lambda}_{Vij}$, making $\mathbf{\Lambda}_{Vij}^{-1}$ the weighting that achieves minimum mean squared error in the unweighted case. Second, the weighting function $w_{ij} = \frac{1}{1+\mathbf{r}_{ij}^T\mathbf{\Lambda}_{ij}^{-1}\mathbf{r}_{ij}}$ smoothly down-weights outliers or points far from the current estimate, preventing extreme influence but not drastically altering the relative weights for typical samples. A $\mathbf{\Lambda}_{ij}$ far from $\mathbf{\Lambda}_{Vij}$ would tend to bias both the numerator and the reweighting, so the optimal $\mathbf{\Lambda}_{ij}$ should be close to $\mathbf{\Lambda}_{Vij}$. Finally, the loss surface defined by the expected error is locally smooth and strongly convex near $\mathbf{\Lambda}_{Vij}^{-1}$, implying stability of the minimizer under moderate perturbations introduced by the weights, which again suggests that $\mathbf{\Lambda}_{ij}^*$ should remain near $\mathbf{\Lambda}_{Vij}^{-1}$.

7.11 Natural Gradient Interpretation of the AFA Update

The precision-weighted attention mechanism used in AFA can be interpreted as the solution to a score-matching or variational inference problem. Consider the log-likelihood of the state estimate $\mathbf{z}_i^C$ given noisy neighbors:

$$\ell = -\frac{1}{2}\sum_{j=1}^{n}\left(\hat{\mathbf{z}}_{ij} - \mathbf{z}_i^C\right)^\top \mathbf{P}_{ij}^C\left(\hat{\mathbf{z}}_{ij} - \mathbf{z}_i^C\right) + \text{const.}$$

The gradient is:

$$\nabla_{\mathbf{z}_i^C} \ell = 2 \sum_j \mathbf{P}_{ij}^C \left(\mathbf{z}_i^C - \hat{\mathbf{z}}_{ij} \right).$$

We then update our estimate via gradient descent:

$$\mathbf{z}_i^{C(k+1)} = \mathbf{z}_i^{C(k)} - \eta \nabla_{\mathbf{z}_i^C} \ell.$$

Since the loss is quadratic, its Hessian is constant:

$$\nabla^2_{\mathbf{z}_i^C} \mathcal{L} = 2 \sum_j \mathbf{P}_{ij}^C := 2\mathbf{P}_i^C.$$

The Fisher Information Matrix (FIM), is defined as

$$\mathbf{F}_i = \mathbb{E}_{\hat{\mathbf{z}}_{ij} \sim p(\hat{\mathbf{z}}_{ij} | \mathbf{z}_i^C)} \left[\nabla_{\mathbf{z}_i^C} \log p(\hat{\mathbf{z}}_{ij} \mid \mathbf{z}_i^C) \nabla_{\mathbf{z}_i^C} \log p(\hat{\mathbf{z}}_{ij} \mid \mathbf{z}_i^C)^\top \right]$$

Each term contributes:

$$\mathbb{E}\left[\mathbf{P}_{ij}^C (\hat{\mathbf{z}}_{ij} - \mathbf{z}_i^C)(\hat{\mathbf{z}}_{ij} - \mathbf{z}_i^C)^\top \mathbf{P}_{ij}^C \right] = \mathbf{P}_{ij}^C \cdot \left(\mathbf{P}_{ij}^C \right)^{-1} \cdot \mathbf{P}_{ij}^C = \mathbf{P}_{ij}^C$$

Therefore, the total Fisher Information Matrix is:

$$\mathbf{F}_i = \sum_j \mathbf{P}_{ij}^C = \mathbf{P}_i^C$$

The natural gradient update is:

$$\mathbf{z}_i^{C(k+1)} = \mathbf{z}_i^{C(k)} - \eta \mathbf{F}_i^{-1} \nabla_{\mathbf{z}_i^C} \ell.$$

Unfortunately, $\mathbf{z}_i^C$ is unknown. However, we can approximate it using a plug-in estimate $\bar{\mathbf{z}}_i \approx \mathbf{z}_i$:

$$\nabla_{\mathbf{z}_i^C} \ell \approx 2 \sum_j \mathbf{P}_{ij}^C \left(\mathbf{z}_i - \hat{\mathbf{z}}_{ij} \right).$$

Plugging in the expressions for the gradient and FIM, we obtain:

$$\mathbf{z}_i^{(k+1)} = \mathbf{z}_i^{(k)} - 2\eta \left(\sum_j \mathbf{P}_{ij}^C \right)^{-1} \sum_j \mathbf{P}_{ij}^C \left(\mathbf{z}_i^{(k)} - \hat{\mathbf{z}}_{ij} \right).$$

$$= (1 - 2\eta)\mathbf{z}_i^{(k)} + 2\eta\hat{\mathbf{z}}_{ij}^{\text{avg},(k)}$$

To recover the exact weighted-average in one step, we can set $\eta = \frac{1}{2}$. More conservatively, we can set $\eta = \frac{\delta_{R_i}}{2}$, where $\delta_{R_i} \in (0, 1]$, so:

$$\mathbf{z}_i^{(k+1)} = (1 - \delta_{R_i})\mathbf{z}_i^{(k)} + \delta_{R_i}\hat{\mathbf{z}}_{ij}^{\text{avg},(k)}$$

where

$$\hat{\mathbf{z}}_i^{\text{avg}(k)} = \left(\sum_j \mathbf{P}_{ij}^C\right)^{-1} \sum_j \mathbf{P}_{ij}^C \hat{\mathbf{z}}_{ij}^{(k)}, \quad \text{and} \quad \mathbf{z}_i^{(1)} = \mathbf{z}_i,$$

which is exactly the Adaptive Filter Attention (AFA) update rule (before diagonalizing and including the adaptive update to $\mathbf{P}_{ij}^C$).

Note that while the Fisher Information Matrix is computed as a conditional expectation under the original model $p(\hat{\mathbf{z}}_{ij} \mid \mathbf{z}_i^C)$, the gradient is evaluated under a surrogate loss in which $\mathbf{z}_i^C$ is approximated by a learnable plug-in estimate $\mathbf{z}_i$. This results in a hybrid update that makes use of model-informed curvature from the original likelihood while remaining tractable to compute through a variational approximation. Because precision matrices define the local curvature (Fisher information) of Gaussian distributions, updating them aligns with performing natural gradient steps in the space of distributions.

7.12 Variational Interpretation

Recall our probabilistic model in which the true latent state $\mathbf{z}^C \in \mathbb{R}^d$ generates observations $\hat{\mathbf{z}}_j$ according to a Gaussian likelihood:

$$p(\hat{\mathbf{z}}_j \mid \mathbf{z}^C) = \mathcal{N}(\hat{\mathbf{z}}_j \mid \mathbf{z}^C, \mathbf{V}_j^C)$$

where each $\hat{\mathbf{z}}_j$ is a noisy measurement of the same latent $\mathbf{z}^C$, and $\mathbf{V}_j^C$ is the covariance associated with $\hat{\mathbf{z}}_j$ (with precision matrix $\mathbf{P}_j^C := \mathbf{V}_j^{C-1}$).

Assume a Gaussian prior on $\mathbf{z}^C$: $p(\mathbf{z}^C) = \mathcal{N}(\mathbf{z}^C \mid \boldsymbol{\mu}_0, \mathbf{P}_0^{-1})$.

The unnormalized posterior is: $p(\mathbf{z}^C \mid \{\hat{\mathbf{z}}_j\}) \propto p(\mathbf{z}^C) \prod_j p(\hat{\mathbf{z}}_j \mid \mathbf{z}^C)$.

The log-likelihood is:

$$\ell := \log p(\mathbf{z}^C \mid \{\hat{\mathbf{z}}_j\}) = -\frac{1}{2}(\mathbf{z}^C - \boldsymbol{\mu}_0)^T \mathbf{P}_0 (\mathbf{z}^C - \boldsymbol{\mu}_0) - \frac{1}{2} \sum_j (\hat{\mathbf{z}}_j - \mathbf{z}^C)^T \mathbf{P}_j^C (\hat{\mathbf{z}}_j - \mathbf{z}^C) + \text{const.}$$

Regrouping terms to find a quadratic form in $\mathbf{z}^C$:

$$\begin{aligned} \ell &= -\frac{1}{2}(\mathbf{z}^C)^T \left(\mathbf{P}_0 + \sum_j \mathbf{P}_j^C \right) \mathbf{z}^C + (\mathbf{z}^C)^T \left(\mathbf{P}_0 \boldsymbol{\mu}_0 + \sum_j \mathbf{P}_j^C \hat{\mathbf{z}}_j \right) + \text{const.} \\ &= -\frac{1}{2}(\mathbf{z}^C)^T \mathbf{P} \mathbf{z}^C + (\mathbf{z}^C)^T \mathbf{b} + \text{const.} \end{aligned}$$

where $\mathbf{P} = \mathbf{P}_0 + \sum_j \mathbf{P}_j^C$ and $\mathbf{b} = \mathbf{P}_0 \boldsymbol{\mu}_0 + \sum_j \mathbf{P}_j^C \hat{\mathbf{z}}_j$.

Completing the square, this is:

$$\ell = -\frac{1}{2}(\mathbf{z}^C - \hat{\mathbf{z}}^C)^T \mathbf{P} (\mathbf{z}^C - \hat{\mathbf{z}}^C) + \text{const.}$$

where $\hat{\mathbf{z}}^C := \mathbf{P}^{-1}\mathbf{b}$. Hence, the posterior is Gaussian:

$$p(\mathbf{z}^C \mid \{\hat{\mathbf{z}}_j\}) = \mathcal{N}\left(\mathbf{z}^C \mid \hat{\mathbf{z}}^C, \mathbf{P}^{-1}\right)$$

with posterior mean $\hat{\mathbf{z}}^C = \mathbf{P}^{-1}\left(\mathbf{P}_0 \boldsymbol{\mu}_0 + \sum_j \mathbf{P}_j^C \hat{\mathbf{z}}_j\right)$ and posterior precision $\mathbf{P} = \mathbf{P}_0 + \sum_j \mathbf{P}_j^C$, which is the Maximum A Posteriori (MAP) solution.

Let us now relax the assumption of a Gaussian prior on $\mathbf{z}^C$. We can approximate the posterior using variational inference. Define a variational distribution:

$$q(\mathbf{z}^C) = \mathcal{N}(\mathbf{z}^C \mid \hat{\mathbf{z}}^C, \mathbf{P}^{-1}).$$

The evidence lower bound (ELBO) is given by:

$$\mathcal{L}(q) = \mathbb{E}_{q(\mathbf{z}^C)}\left[\sum_j \log p(\hat{\mathbf{z}}_j \mid \mathbf{z}^C)\right] - \text{KL}\left(q(\mathbf{z}^C)||p(\mathbf{z}^C)\right)$$

The expected log-likelihood is:

$$\mathbb{E}_{q(\mathbf{z}^C)}\left[\sum_j \log p(\hat{\mathbf{z}}_j \mid \mathbf{z}^C)\right] = -\frac{1}{2}\sum_j \left[\text{Tr}(\mathbf{P}_j^C\mathbf{P}^{-1}) + (\hat{\mathbf{z}}_j - \hat{\mathbf{z}}^C)^T\mathbf{P}_j^C(\hat{\mathbf{z}}_j - \hat{\mathbf{z}}^C)\right] + \text{const.}$$

The KL divergence term is:

$$\text{KL}\left[q(\mathbf{z}^C)\|p(\mathbf{z}^C)\right] = \frac{1}{2}\left[\text{Tr}(\mathbf{P}_0\mathbf{P}^{-1}) + (\hat{\mathbf{z}}^C - \boldsymbol{\mu}_0)^T\mathbf{P}_0(\hat{\mathbf{z}}^C - \boldsymbol{\mu}_0) - d + \log\frac{|\mathbf{P}^{-1}|}{|\mathbf{P}_0^{-1}|}\right].$$

If the prior precision is uninformative, $\mathbf{P}_0 \to 0$, then

$$\mathbf{P} = \sum_j \mathbf{P}_j^C, \quad \hat{\mathbf{z}}^C = \mathbf{P}^{-1}\sum_j \mathbf{P}_j^C\hat{\mathbf{z}}_j,$$

which reduces to the MLE solution: a precision-weighted average of the noisy observations $\hat{\mathbf{z}}_j$.

Alternatively, we can formalize a plug-in approximation by using a Dirac delta distribution centered at a learnable estimate $\mathbf{z}_i$ as our variational distribution:

$$q(\mathbf{z}_i^C) := \delta(\mathbf{z}_i^C - \mathbf{z}_i),$$

This corresponds to a maximum likelihood estimate of $\mathbf{z}_i$, minimizing a precision-weighted squared loss. The variational loss is then:

$$\mathcal{L}(\mathbf{z}_i) := \mathbb{E}_{q(\mathbf{z}_i^C)}\left[\sum_j \left(\mathbf{z}_i^C - \hat{\mathbf{z}}_{ij}\right)^\top \mathbf{P}_{ij}^C \left(\mathbf{z}_i^C - \hat{\mathbf{z}}_{ij}\right)\right] - \mathbb{H}[q(\mathbf{z}_i^C)].$$

Since q is a delta function, the entropy term $\mathbb{H}[q] = 0$, and the expectation reduces to:

$$\mathcal{L}(\mathbf{z}_i) = \sum_j \left(\mathbf{z}_i - \hat{\mathbf{z}}_{ij}\right)^\top \mathbf{P}_{ij}^C \left(\mathbf{z}_i - \hat{\mathbf{z}}_{ij}\right),$$

This recovers a quadratic loss in $\mathbf{z}_i$, and minimizing with respect to $\mathbf{z}_i$ again recovers the MLE.

The above motivates a combined update rule that jointly updates the posterior mean estimate $\hat{\mathbf{z}}^{C,(k)}$ and the prior precision estimate $\hat{\mathbf{P}}_0^{(k)}$ as the model propagates through layers k (whereas in our prior formulation, we iteratively updated the mean but left the prior precision fixed at each layer). This enables recursive refinement of both the latent state estimate and its uncertainty in a feedforward manner.

At layer k, we compute the following:
Precision update from residuals:

$$\hat{\mathbf{P}}_j^{C,(k)} = \hat{\mathbf{P}}_j^{C,(k)} + \mathbf{r}_j^{(k)} \left(\mathbf{r}_j^{(k)}\right)^T$$

Posterior precision:

$$\hat{\mathbf{P}}^{(k)} = \hat{\mathbf{P}}_0^{(k)} + \sum_j \hat{\mathbf{P}}_j^{C,(k)}$$

Posterior mean:

$$\hat{\mathbf{z}}^{C,(k)} = (\hat{\mathbf{P}}^{(k)})^{-1} \left(\hat{\mathbf{P}}_0^{(k)} \boldsymbol{\mu}_0^{(k)} + \sum_j \hat{\mathbf{P}}_j^{C,(k)} \hat{\mathbf{z}}_j^{(k)} \right)$$

Update priors for next layer:

$$\boldsymbol{\mu}_0^{(k+1)} = \hat{\mathbf{z}}^{C,(k)}, \quad \hat{\mathbf{P}}_0^{(k+1)} = \hat{\mathbf{P}}^{(k)}.$$

Assuming all precision matrices are diagonal matrices $\mathbf{\Lambda}_P^{(k)}$, the posterior updates at layer k become elementwise:

$$\mathbf{\Lambda}_P^{(k)} = \mathbf{\Lambda}_{P_0}^{(k)} + \sum_j \mathbf{\Lambda}_{P_j}^{(k)}$$

$$\hat{\mathbf{z}}^{C,(k)} = \frac{\mathbf{\Lambda}_{P_0}^{(k)} \odot \boldsymbol{\mu}_0^{(k)} + \sum_j \mathbf{\Lambda}_{P_j}^{(k)} \odot \hat{\mathbf{z}}_j^{(k)}}{\mathbf{\Lambda}_{P_0}^{(k)} + \sum_j \mathbf{\Lambda}_{P_j}^{(k)}}$$

However, since we're not getting new independent information per layer, it may make more sense to keep the precision fixed and perform iterative refinement of the mean under fixed uncertainty, i.e. our previous iteration.

7.13 Properties of the Mahalanobis-based Similarity Score

Consider the similarity score:

$$w_{ij} = \frac{\mathbf{P}_{ij}}{1 + \mathbf{r}_{ij}^\top \mathbf{P}_{ij} \mathbf{r}_{ij}}.$$

Taking the logarithm, we have:

$$s_{ij} := \log w_{ij} = \log \mathbf{P}_{ij} - \log\left(1 + \mathbf{r}_{ij}^\top \mathbf{P}_{ij} \mathbf{r}_{ij}\right).$$

Exponentiating and normalizing yields a softmax distribution. Using the approximation $\log(1 + x) \approx x$ for $x \ll 1$, we get:

$$s_{ij} \approx \log \mathbf{P}_{ij} - \mathbf{r}_{ij}^\top \mathbf{P}_{ij} \mathbf{r}_{ij}.$$

Exponentiating,

$$w_{ij} \approx \mathbf{P}_{ij} \exp\left(-\mathbf{r}_{ij}^\top \mathbf{P}_{ij} \mathbf{r}_{ij}\right),$$

which corresponds to a Gaussian kernel similarity.

Consider the simpler case when $\mathbf{P} = \mathbf{I}$:

$$w_{ij} = \frac{\mathbf{1}}{1 + \mathbf{r}_{ij}^T \mathbf{r}_{ij}}, \quad \text{and} \quad q_{ij} = w_{ij} \Big/ \sum_{j=1}^{m} w_{ij}$$

Here, we compare the properties of q_{ij} as an attention score with those of softmax. Even before normalization, $p_{ij} \in (0, 1]$ for any value of $\mathbf{r}_{ij}$, and is monotonic decreasing as the norm of the residual increases. In contrast to softmax, this form of similarity score is related to Euclidean distance rather than probability, and falls of quadratically rather than exponentially.

Let $a_{ij} = \mathbf{r}_{ij}^T \mathbf{r}_{ij}$. We can compute:

$$\sum_{j=1}^{m} w_{ij} = \sum_{j=1}^{m} \frac{1}{1 + a_{ij}} = \frac{\sum_{j=1}^{m} \prod_{k \neq j} (1 + a_{ik})}{\prod_k (1 + a_{ik})}$$

$$\left(\sum_{j=1}^{m} w_{ij} \right)^{-1} = \frac{\prod_k (1 + a_{ik})}{\sum_{j=1}^{m} \prod_{k \neq j} (1 + a_{ik})}$$

Letting $a_{ij} = c, \forall j$,

$$q_{ij} = \frac{\prod_k (1 + c)}{\sum_{j=1}^{m} \prod_{k \neq j} (1 + c)} \frac{1}{1 + c} = \frac{1}{m}$$

while on the other hand, if $a_{il} \ll a_{ij}, \forall j \neq l$,

$$q_{ij} \approx \frac{\prod_k (1 + a_{ik})}{\prod_{k \neq l} (1 + a_{ik})} \frac{1}{1 + a_{ij}} = \frac{1 + a_{il}}{1 + a_{ij}}$$

So, this metric behaves similarly to the softmax function: If one residual $\mathbf{r}_{il}$ is much smaller than the others, then the output $\mathbf{y}_i$ is close to the dominant term $\mathbf{x}_l$. If all residuals are similar in size and not too large, then the output is an approximately uniform weighting of the inputs. However, the key difference is in how the weighting behaves when all residuals are small (on the order of 1 or less). Softmax always picks a dominant term if the differences between inputs are large in relative terms, no matter the absolute scale. This similarity score does not: If all residuals are relatively small (even if one is much smaller than the others), then the attention distribution remains roughly uniform.

7.14 The Attention Mechanism in Matrix Form

Let $\mathbf{X}_Q, \mathbf{X}_K, \mathbf{X}_V \in \mathbb{R}^{n \times m}$ each be a collection of n embedding vectors of dimension m stacked horizontally, that is:

$$\mathbf{X}_Q = \begin{bmatrix} - & \mathbf{x}_{\mathbf{q}1}^T & - \\ - & \mathbf{x}_{\mathbf{q}2}^T & - \\ & \vdots & \\ - & \mathbf{x}_{\mathbf{q}n}^T & - \end{bmatrix}, \quad \mathbf{X}_K = \begin{bmatrix} - & \mathbf{x}_{\mathbf{k}1}^T & - \\ - & \mathbf{x}_{\mathbf{k}2}^T & - \\ & \vdots & \\ - & \mathbf{x}_{\mathbf{k}n}^T & - \end{bmatrix}, \quad \mathbf{X}_V = \begin{bmatrix} - & \mathbf{x}_{\mathbf{v}1}^T & - \\ - & \mathbf{x}_{\mathbf{v}2}^T & - \\ & \vdots & \\ - & \mathbf{x}_{\mathbf{v}n}^T & - \end{bmatrix}.$$

Define weight matrices $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{m \times m}$, and let $\mathbf{Q} = \mathbf{X}_Q\mathbf{W}_Q$, $\mathbf{Q} = \mathbf{X}_K\mathbf{W}_K$, $\mathbf{V} = \mathbf{X}_V\mathbf{W}_V$. For simplicity, we will work with $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times m}$:

$$\mathbf{Q} = \begin{bmatrix} - & \mathbf{q}_1^T & - \\ - & \mathbf{q}_2^T & - \\ & \vdots & \\ - & \mathbf{q}_n^T & - \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} - & \mathbf{k}_1^T & - \\ - & \mathbf{k}_2^T & - \\ & \vdots & \\ - & \mathbf{k}_n^T & - \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} - & \mathbf{v}_1^T & - \\ - & \mathbf{v}_2^T & - \\ & \vdots & \\ - & \mathbf{v}_n^T & - \end{bmatrix}.$$

The attention matrix is:

$$\mathbf{Q}\mathbf{K}^T = \begin{bmatrix} \mathbf{q}_1^T\mathbf{k}_1 & \mathbf{q}_1^T\mathbf{k}_2 & \dots & \mathbf{q}_1^T\mathbf{k}_n \\ \mathbf{q}_2^T\mathbf{k}_1 & \mathbf{q}_2^T\mathbf{k}_2 & \dots & \mathbf{q}_2^T\mathbf{k}_n \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{q}_n^T\mathbf{k}_1 & \mathbf{q}_n^T\mathbf{k}_2 & \dots & \mathbf{q}_n^T\mathbf{k}_n \end{bmatrix}.$$

Ignoring, for now, the softmax operation:

$$\mathbf{Y} = \left(\mathbf{Q}\mathbf{K}^T\right)\mathbf{V} = \begin{bmatrix} (\mathbf{q}_1^T\mathbf{k}_1)v_{11} + (\mathbf{q}_1^T\mathbf{k}_2)v_{21} + \dots & (\mathbf{q}_1^T\mathbf{k}_1)v_{12} + (\mathbf{q}_1^T\mathbf{k}_2)v_{22} + \dots & \dots \\ (\mathbf{q}_2^T\mathbf{k}_1)v_{11} + (\mathbf{q}_2^T\mathbf{k}_2)v_{21} + \dots & (\mathbf{q}_2^T\mathbf{k}_1)v_{12} + (\mathbf{q}_2^T\mathbf{k}_2)v_{22} + \dots & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}.$$

The outputs are a weighted sum of the values: $\mathbf{y}_i = \sum_{j=1}^{m} p_{ij}\mathbf{v}_j$, where $p_{ij} = \mathbf{q}_j^T\mathbf{k}_i$.

Defining a residual $\mathbf{r}_{ij} = \mathbf{x}_j - \mathbf{x}_i$, the output of self-attention is:

$$\mathbf{y}_i = \sum_j p_{ij}(\mathbf{x}_i + \mathbf{r}_{ij}) = \mathbf{x}_i + \sum_j p_{ij}\mathbf{r}_{ij}$$

That is, the output is the original vector, adjusted by some distance along the direction of each residual.

Consider a matrix generalization of attention, where instead of weighting the inputs by scalars, we multiply by a matrix (so there are m^2 matrices, one for every input-output pair):

$$\mathbf{y}_i = \left(\sum_{j=1}^{m} \tilde{\mathbf{P}}_{ij}\right)^{-1} \sum_{j=1}^{m} \tilde{\mathbf{P}}_{ij}\mathbf{x}_j$$

where

$$\tilde{\mathbf{P}}_{ij} = \left(\mathbf{P}_{ij} + \mathbf{r}_{ij}\mathbf{r}_{ij}^T\right)^{-1} = \mathbf{P}_{ij} - \frac{\mathbf{P}_{ij}\mathbf{r}_{ij}\mathbf{r}_{ij}^T\mathbf{P}_{ij}}{1 + \mathbf{r}_{ij}^T\mathbf{P}_{ij}\mathbf{r}_{ij}}$$

and where every $\mathbf{P}_{ij}$ is a positive definite matrix. In the case where $\tilde{\mathbf{P}}_{ij} = \frac{\mathbf{\Lambda}_{ij}}{1+\mathbf{r}_{ij}^T\mathbf{\Lambda}_{ij}\mathbf{r}_{ij}}$, where $\mathbf{\Lambda}_{ij}$ is a diagonal matrix, the inverse becomes a row-wise normalization, and we have an element-wise generalization of attention.

7.15 Attention as an Adaptive Filter

Consider a loss function using a precision weighted estimator:

$$L = \frac{1}{2}\mathbb{E}\left[\|\mathbf{y} - \bar{\mathbf{x}}\|_2^2\right] = \frac{1}{2}\mathbb{E}\left[\left\|\mathbf{y} - \left(\sum_k \mathbf{P}_k\right)^{-1} \sum_j \mathbf{P}_j\hat{\mathbf{x}}_j\right\|_2^2\right]$$

Observe that we can pull $\left(\sum_k \mathbf{P}_k\right)^{-1}$ outside:

$$L = \frac{1}{2}\mathbb{E}\left[\left\|\left(\sum_k \mathbf{P}_k\right)^{-1}\left(\sum_k \mathbf{P}_k\mathbf{y} - \sum_j \mathbf{P}_j\hat{\mathbf{x}}_j\right)\right\|_2^2\right]$$

$$= \frac{1}{2}\mathbb{E}\left[\left\|\sum_j \mathbf{W}_j\left(\mathbf{y} - \hat{\mathbf{x}}_j\right)\right\|_2^2\right], \quad \text{where} \quad \mathbf{W}_j := \left(\sum_k \mathbf{P}_k\right)^{-1}\mathbf{P}_j$$

,

$$= \frac{1}{2}\mathbb{E}\left[\left(\sum_j \mathbf{W}_j\mathbf{e}_j\right)^T\left(\sum_j \mathbf{W}_j\mathbf{e}_j\right)\right]$$

where $\mathbf{e}_j := \mathbf{y} - \hat{\mathbf{x}}_j$ is the error between the target signal $\mathbf{y}$ and the estimate $\hat{\mathbf{x}}_j$. Note that, unlike in adaptive filtering, we don't actually know $\mathbf{e}_j$, because we don't know $\mathbf{y}$.

If the errors are uncorrelated,

$$\mathbb{E}\left[(\mathbf{W}_i\mathbf{e}_i)^T(\mathbf{W}_j\mathbf{e}_j)\right] = \mathbb{E}\left[\mathbf{e}_i^T\mathbf{W}_i^T\mathbf{W}_j\mathbf{e}_j\right] = \mathbb{E}\left[\text{tr}\left(\mathbf{W}_i^T\mathbf{W}_j\mathbf{e}_j\mathbf{e}_i^T\right)\right] = \text{tr}\left(\mathbf{W}_i^T\mathbf{W}_j\mathbb{E}\left[\mathbf{e}_j\mathbf{e}_i^T\right]\right) = 0$$

Hence, the expectation simplifies:

$$\mathbb{E}\left[\left(\sum_k \mathbf{W}_k\mathbf{e}_k\right)^T\left(\sum_j \mathbf{W}_j\mathbf{e}_j\right)\right] = \mathbb{E}\left[\sum_j (\mathbf{W}_j\mathbf{e}_j)^T(\mathbf{W}_j\mathbf{e}_j)\right]$$

In adaptive filtering, we update our weights in order to match a target signal. Given a loss function:

$$L = \frac{1}{2}\mathbb{E}\left[\sum_j (\mathbf{W}_j\mathbf{e}_j)^T(\mathbf{W}_j\mathbf{e}_j)\right]$$

we compute the gradient:

$$\frac{dL}{d\mathbf{W}_j} = \mathbb{E}\left[\mathbf{W}_j\mathbf{e}_j\mathbf{e}_j^T\right] = \mathbf{W}_j\mathbb{E}\left[\mathbf{e}_j\mathbf{e}_j^T\right]$$

We replace the expectation with an empirical estimate. Since we only have a single sample, this becomes:

$$\frac{dL}{d\mathbf{W}_j} \approx \mathbf{W}_j\mathbf{e}_j\mathbf{e}_j^T$$

And apply an LMS-like gradient update to $\mathbf{W}_j$:

$$\mathbf{W}_j \leftarrow \mathbf{W}_j - \mu\nabla_{\mathbf{W}_j}L = \mathbf{W}_j - \mu\mathbf{W}_j\mathbf{e}_j\mathbf{e}_j^T = \mathbf{W}_j(\mathbf{I} - \mu\mathbf{e}_j\mathbf{e}_j^T)$$

And then normalize:

$$\mathbf{W}_j \leftarrow \left(\sum_k \mathbf{W}_k\right)^{-1} \mathbf{W}_j$$

Setting $\mu = \frac{1}{1+\mathbf{e}^T\mathbf{e}}$ (as in normalized LMS), we recognize the above gradient update as multiplying by the Sherman Morrison formula for the identity.

Now, consider a penalty term of the form: $\sum_j \text{tr}\left(\mathbf{W}_j(\mathbf{I}-\mathbf{P}_j)\mathbf{W}_j^T\right)$.
The gradient of this penalty term is: $\mathbf{W}_j(\mathbf{I}-\mathbf{P}_j) + \mathbf{W}_j(\mathbf{I}-\mathbf{P}_j)^T$.
If $\mathbf{P}_j$ is symmetric (as is any precision matrix), this is: $2\mathbf{W}_j(\mathbf{I}-\mathbf{P}_j)$. Hence, if we define a new loss function of the form:

$$L_P = \frac{1}{2}\sum_j \left(\mathbf{W}_j\mathbf{P}_j\mathbf{e}_j\right)^T\left(\mathbf{W}_j\mathbf{P}_j\mathbf{e}_j\right) + \frac{\alpha}{2}\sum_j \text{tr}\left(\mathbf{W}_j(\mathbf{I}-\mathbf{P}_j)\mathbf{W}_j^T\right)$$

The gradient becomes:

$$\frac{dL_P}{d\mathbf{W}_j} = \mathbf{W}_j\mathbf{P}_j\mathbf{e}_j\mathbf{e}_j^T\mathbf{P}_j^T + \alpha\mathbf{W}_j(\mathbf{I}-\mathbf{P}_j)$$

Hence,

$$\mathbf{W}_j \leftarrow \mathbf{W}_j - \mu\nabla_{\mathbf{W}_j}L = \mathbf{W}_j - \mu\mathbf{W}_j\mathbf{P}_j\mathbf{e}_j\mathbf{e}_j^T\mathbf{P}_j^T - \mu\alpha\mathbf{W}_j(\mathbf{I}-\mathbf{P}_j)$$

Letting $\alpha = 1/\mu$, this is:

$$\mathbf{W}_j \leftarrow \mathbf{W}_j\left(\mathbf{P}_j - \mu\mathbf{P}_j\mathbf{e}_j\mathbf{e}_j^T\mathbf{P}_j^T\right)$$

And setting $\mu = \frac{1}{1+\mathbf{e}_j^T\mathbf{P}_j\mathbf{e}_j}$, we recognize this as multiplying by the full Sherman Morrison formula. So, we've shown that right multiplying by the Sherman Morrison formula is equivalent to a single step of gradient descent, with an appropriate penalty term in the loss function and an adaptive step size. Roughly, the penalty term encourages $\mathbf{W}_j$ to align with $\mathbf{P}_j$ while penalizing large weights, acting similarly to ridge regression but with anisotropic scaling.

We do not have access to the error signal $\mathbf{e}_j$, but if we take $\mathbf{e}_j \approx \mathbf{r}_j$, then we have the following iterative algorithm (where we set $\mathbf{P}_j = \mathbf{I}, \forall j$ for simplicity):

Compute $\mathbf{W}_{ij}^{(t)}$ in parallel for all i:

$$\mathbf{W}_{ij}^{(t)} = \mathbf{W}_{ij}^{(t-1)} \left(\mathbf{I} - \frac{\mathbf{r}_{ij}^{(t)} \mathbf{r}_{ij}^{(t)T}}{1 + \mathbf{r}_{ij}^{(t)T} \mathbf{r}_{ij}^{(t)}} \right)$$

where: $\mathbf{r}_{ij}^{(t)} = \mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}$, and $\mathbf{W}_{ij}^{(0)} = \mathbf{I}$. Then, normalize $\mathbf{W}_{ij}^{(t)}$:

$$\bar{\mathbf{W}}_{ij}^{(t)} = \left(\sum_j \mathbf{W}_{ij}^{(t)} \right)^{-1} \mathbf{W}_{ij}^{(t)}$$

Then, compute the updated estimates:

$$\mathbf{x}_i^{(t+1)} = \sum_j \bar{\mathbf{W}}_{ij}^{(t)} \mathbf{x}_j^{(t)}$$

But this adaptive filtering algorithm is simply a forward pass of multi-layer matrix attention, where $\mathbf{x}_i^{(t)}$ corresponds to the ith output of the tth layer. Hence, multi-layer matrix attention, when constructed with matrix-valued weights depending on residuals and normalized properly, can be viewed as performing a form of adaptive filtering — it adapts based on the input residuals and precision matrices at inference time, without explicitly modifying learned parameters via gradient descent.

This algorithm is difficult to analyze because the update matrices are time-varying and data-dependent. Although each Sherman-Morrison term is symmetric and contractive, their product $\mathbf{W}_{ij}^{(t)}$ is generally non-symmetric and non-commutative, making standard spectral or Lyapunov analyses inapplicable.

Instead, consider a simpler iteration:

$$\mathbf{W}_{ij}^{(t)} = \mathbf{W}_{ij}^{(t-1)} \left(\frac{\mathbf{\Lambda}_{ij}^{(t)}}{1 + \mathbf{r}_{ij}^{(t)T} \mathbf{\Lambda}_{ij}^{(t)} \mathbf{r}_{ij}^{(t)}} \right)$$

where: $\mathbf{r}_{ij}^{(t)} = \mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}$, each $\mathbf{\Lambda}_{ij}^{(t)}$ is diagonal and positive, $\sum_j \bar{\mathbf{W}}_{ij}^{(t)} = \mathbf{I}$ (row-normalization), and $\mathbf{W}_{ij}^{(0)} = \mathbf{I}$. Then, compute the updated estimates:

$$\mathbf{x}_i^{(t+1)} = \alpha \mathbf{x}_i^{(t)} + (1-\alpha) \sum_j \bar{\mathbf{W}}_{ij}^{(t)} \mathbf{x}_j^{(t)}, \quad \text{where} \quad \alpha \in (0, 1]$$

This is a form of consensus algorithm with data-dependent weights. Let $\mathbf{x}_i^{(t)} = [x_{i,1}^{(t)}, \ldots, x_{i,m}^{(t)}]^T$ denote the vector estimates at iteration t, and define the dimension-wise average $\bar{x}_d^{(t)} = \frac{1}{n} \sum_{i=1}^n x_{i,d}^{(t)}$, and the disagreement error $e_{i,d}^{(t)} := x_{i,d}^{(t)} - \bar{x}_d^{(t)}$. The Lyapunov function measuring total disagreement in dimension d is

$$L_d(t) = \frac{1}{2} \sum_{i=1}^n \left(e_{i,d}^{(t)}\right)^2.$$

Because the weights $\bar{\mathbf{W}}_{ij}^{(t)}$ are diagonal and row-normalized, the update decouples dimension-wise:

$$x_{i,d}^{(t+1)} = \alpha x_{i,d}^{(t)} + (1-\alpha) \sum_j \bar{w}_{ij,d}^{(t)} x_{j,d}^{(t)}, \quad \text{where} \quad \bar{w}_{ij,d}^{(t)} > 0, \quad \text{and} \quad \sum_j \bar{w}_{ij,d}^{(t)} = 1$$

This scalar-weighted averaging with positive row-stochastic weights is a classic consensus iteration. $L_d(t)$ is nonnegative and strictly decreasing unless consensus is reached in dimension d, guaranteeing convergence of the errors to a finite limit. Importantly, the limiting values $x_{i,d}^{(\infty)}$ may differ across agents because the weights are only row-stochastic and possibly asymmetric. If the weights were doubly stochastic or symmetric, the limit would coincide for all agents, yielding consensus.

Bibliography

[AHL23] Abdul Fatir Ansari, Alvin Heng, Andre Lim, and Harold Soh. "Neural Continuous-Discrete State Space Models for Irregularly-Sampled Time Series.", 2023.

[Ant97] Panos J. Antsaklis. *Linear systems / Panos J. Antsaklis, Anthony N. Michel.* McGraw-Hill,, New York, 1997.

[BBH18] Maximilian Behr, Peter Benner, and Jan Heiland. "Solution Formulas for Differential Sylvester and Lyapunov Equations.", 2018.

[BCB16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate.", 2016.

[BCK15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. "Weight Uncertainty in Neural Networks.", 2015.

[BHD25] Long Minh Bui, Tho Tran Huu, Duy Dinh, Tan Minh Nguyen, and Trong Nghia Hoang. "Revisiting Kernel Attention with Correlated Gaussian Process Representation.", 2025.

[BJT16] Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. "Dynamic Filter Networks.", 2016.

[BLX25] Aviv Bick, Kevin Y. Li, Eric P. Xing, J. Zico Kolter, and Albert Gu. "Transformers to SSMs: Distilling Quadratic Knowledge to Subquadratic Models.", 2025.

[BO15] Justin Bayer and Christian Osendorfer. "Learning Stochastic Recurrent Networks.", 2015.

[BRV23] Jose Agustin Barrachina, Chengfang Ren, Gilles Vieillard, Christele Morisseau, and Jean-Philippe Ovarlez. "Theory and Implementation of Complex-Valued Neural Networks.", 2023.

[BS23] Giovanni Araujo Bacochina and Rodrigo Clemente Thom de Souza. "Element-Wise Attention Layers: an option for optimization.", 2023.

[BSZ24] Ali Behrouz, Michele Santacatterina, and Ramin Zabih. "MambaMixer: Efficient Selective State Space Models with Dual Token and Channel Selection.", 2024.

[CC23] Jen-Tzung Chien and Yi-Hsiang Chen. "Learning Continuous-Time Dynamics With Attention." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **45**(2):1906–1918, 2023.

[CLD22] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. "Rethinking Attention with Performers.", 2022.

[CLS09] Federico S. Cattivelli, Cassio G. Lopes, and Ali H. Sayed. "Diffusion Strategies for Distributed Kalman Filtering: Formulation and Performance Analysis." In *Proceedings of the 43rd Asilomar Conference on Signals, Systems and Computers*, pp. 908–912. IEEE, 2009.

[CRB19] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. "Neural Ordinary Differential Equations.", 2019.

[CS10] Federico S. Cattivelli and Ali H. Sayed. "Diffusion LMS Strategies for Distributed Estimation." *IEEE Transactions on Signal Processing*, **58**(3):1035–1048, 2010.

[CTT24] Yingyi Chen, Qinghua Tao, Francesco Tonin, and Johan A. K. Suykens. "Self-Attention through Kernel-Eigen Pair Sparse Variational Gaussian Processes.", 2024.

[DCW24] Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. "Shifting Attention to Relevance: Towards the Predictive Uncertainty Quantification of Free-Form Large Language Models." In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5050–5063, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[DFT91] John Comstock Doyle, Bruce A. Francis, and Allen R. Tannenbaum. *Feedback Control Theory*. Prentice Hall Professional Technical Reference, 1991.

[DG24] Tri Dao and Albert Gu. "Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality.", 2024.

[DM17] Subhro Das and José M. F. Moura. "Consensus+Innovations Distributed Kalman Filter With Optimized Gains." *IEEE Transactions on Signal Processing*, **65**(2):467–481, 2017.

[Fen25] Guoxin Feng. "Element-wise Attention Is All You Need.", 2025.

[GB24] Gautam Goel and Peter Bartlett. "Can a Transformer Represent a Kalman Filter?", 2024.

[GBM21] Prasad Gabbur, Manjot Bilkhu, and Javier Movellan. "Probabilistic Attention for Interactive Segmentation.", 2021.

[GD24] Albert Gu and Tri Dao. “Mamba: Linear-Time Sequence Modeling with Selective State Spaces.”, 2024.

[GDE20] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Re. “HiPPO: Recurrent Memory with Optimal Polynomial Projections.”, 2020.

[GGD22] Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. “It’s Raw! Audio Generation with State-Space Models.”, 2022.

[GGR22] Albert Gu, Karan Goel, and Christopher Ré. “Efficiently Modeling Long Sequences with Structured State Spaces.”, 2022.

[GSR18] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. “Neural Processes.”, 2018.

[GWJ22] Hongji Guo, Hanjing Wang, and Qiang Ji. “Uncertainty-Guided Probabilistic Transformer for Complex Action Recognition.” In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20020–20029, 2022.

[GZ22] Nicholas Geneva and Nicholas Zabaras. “Transformers for modeling physical systems.” *Neural Networks*, **146**:272–289, February 2022.

[Ham24] M. M. Hammad. “Comprehensive Survey of Complex-Valued Neural Networks: Insights into Backpropagation and Activation Functions.”, 2024.

[Hay02] Simon Haykin. *Adaptive filter theory*. Prentice Hall, Upper Saddle River, NJ, 4th edition, 2002.

[HLK18] Jay Heo, Hae Beom Lee, Saehoon Kim, Juho Lee, Kwang Joon Kim, Eunho Yang, and Sung Ju Hwang. “Uncertainty-Aware Attention for Reliable Interpretation and Prediction.”, 2018.

[HLQ23] Zhongzhan Huang, Mingfu Liang, Jinghui Qin, Shanshan Zhong, and Liang Lin. "Understanding Self-attention Mechanism via Dynamical System Perspective.", 2023.

[ICE24a] Georgios Ioannides, Aman Chadha, and Aaron Elkins. "Gaussian Adaptive Attention is All You Need: Robust Contextual Representations Across Multiple Modalities." *CoRR*, **abs/2401.11143**, 2024.

[ICE24b] Georgios Ioannides, Aman Chadha, and Aaron Elkins. "Gaussian Adaptive Attention is All You Need: Robust Contextual Representations Across Multiple Modalities.", 2024.

[Jaz70] Andrew H. Jazwinski. *Stochastic processes and filtering theory*. Number 64 in Mathematics in science and engineering. Acad. Press, New York, NY [u.a.], 1970.

[JJK21] Sheo Yon Jhin, Minju Jo, Taeyong Kong, Jinsung Jeon, and Noseong Park. "ACE-NODE: Attentive Co-Evolving Neural Ordinary Differential Equations.", 2021.

[JSH21] Sheo Yon Jhin, Heejoo Shin, Seoyoung Hong, Solhee Park, and Noseong Park. "Attentive Neural Controlled Differential Equations for Time-series Classification and Forecasting.", 2021.

[Kal60] Rudolph Emil Kalman. "A New Approach to Linear Filtering and Prediction Problems." *Transactions of the ASME–Journal of Basic Engineering*, **82**(Series D):35–45, 1960.

[KDH17] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. "Structured Attention Networks.", 2017.

[Kir04] Donald. E Kirk. *Optimal Control Theory: An Introduction*. Dover, New York, 2004.

[KL24] Kyung Geun Kim and Byeong Tak Lee. "Self-attention with temporal prior: can we learn more from the arrow of time?" *Frontiers in Artificial Intelligence*, **7**, August 2024.

[KMF20] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. "Neural Controlled Differential Equations for Irregular Time Series.", 2020.

[KMS19] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. "Attentive Neural Processes.", 2019.

[KSS15] Rahul G. Krishnan, Uri Shalit, and David Sontag. "Deep Kalman Filters.", 2015.

[KSS16] Rahul G. Krishnan, Uri Shalit, and David Sontag. "Structured Inference Networks for Nonlinear State Space Models.", 2016.

[KVP20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention.", 2020.

[LZW23] Shahar Lutati, Itamar Zimerman, and Lior Wolf. "Focus Your Attention (with Adaptive IIR Filters).", 2023.

[Mac92] David J. C. MacKay. "A practical Bayesian framework for backpropagation networks." *Neural Comput.*, **4**(3):448–472, May 1992.

[MG20] Javier R. Movellan and Prasad Gabbur. "Probabilistic Transformers.", 2020.

[NAT24] Stefan K. Nielsen, Laziz U. Abdullaev, Rachel S. Y. Teo, and Tan M. Nguyen. "Elliptical Attention.", 2024.

[NBD21] Alexander Norcliffe, Cristian Bodnar, Ben Day, Jacob Moss, and Pietro Liò. "Neural ODE Processes.", 2021.

[Nea96] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996.

[NGG22] Eric Nguyen, Karan Goel, Albert Gu, Gordon W. Downs, Preey Shah, Tri Dao, Stephen A. Baccus, and Christopher Ré. "S4ND: Modeling Images and Videos as Multidimensional Signals Using State Spaces.", 2022.

[NGS25] Mayank Nautiyal, Stela Arranz Gheorghe, Kristiana Stefa, Li Ju, Ida-Maria Sintorn, and Prashant Singh. "PARIC: Probabilistic Attention Regularization for Language Guided Image Classification from Pre-trained Vison Language Models.", 2025.

[RSL21] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. "Hopfield Networks is All You Need.", 2021.

[SAD24] Jerome Sieber, Carmen Amo Alonso, Alexandre Didier, Melanie N. Zeilinger, and Antonio Orvieto. "Understanding the differences in Foundation Models: Attention, State Space Models, and Recurrent Neural Networks.", 2024.

[Say13] Ali H. Sayed. "Diffusion Adaptation over Networks.", 2013.

[SDH23] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. "Retentive Network: A Successor to Transformer for Large Language Models.", 2023.

[Shi22] Alexander Shim. "A Probabilistic Interpretation of Transformers.", 2022.

[SJK21] Kyungwoo Song, Yohan Jung, Dongjun Kim, and Il-Chul Moon. "Implicit Kernel Attention.", 2021.

[SLP23] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. "RoFormer: Enhanced Transformer with Rotary Position Embedding.", 2023.

[SMM23] Arne Schmidt, Pablo Morales-Álvarez, and Rafael Molina. "Probabilistic Attention based on Gaussian Processes for Deep Multiple Instance Learning.", 2023.

[SP05] Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2005.

[Spe08] Jason Lee Speyer. *Stochastic Processes, Estimation, and Control*. Society for Industrial and Applied Mathematics, USA, 2008.

[SSK21] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-Based Generative Modeling through Stochastic Differential Equations.", 2021.

[Ste87] Samuel D. Stearns. *Fundamentals of adaptive signal processing*, p. 246–288. Prentice-Hall, Inc., USA, 1987.

[SUV18] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. "Self-Attention with Relative Position Representations.", 2018.

[SWF22] Karthik Abinav Sankararaman, Sinong Wang, and Han Fang. “BayesFormer: Transformer with Uncertainty Estimation.”, 2022.

[SWL23] Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. “Simplified State Space Layers for Sequence Modeling.”, 2023.

[SYC18] Shang-Yu Su, Pei-Chieh Yuan, and Yun-Nung Chen. “Dynamically Context-Sensitive Time-Decay Attention for Dialogue Modeling.”, 2018.

[Tab21] Marco Taboga. “Multivariate normal distribution - Maximum Likelihood Estimation.” Lectures on probability theory and mathematical statistics. Kindle Direct Publishing. Online appendix., 2021.

[TBY19] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. “Transformer Dissection: A Unified Understanding of Transformer’s Attention via the Lens of Kernel.”, 2019.

[TBZ18] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. “Deep Complex Networks.”, 2018.

[VSP23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention Is All You Need.”, 2023.

[Zha21] Zhiyong Zhang. “A Note on Wishart and Inverse Wishart Priors for Covariance Matrix.” *Journal of Behavioral Data Science*, **1**(2):119–126, 2021.