State Space Models

ELG 5218 - Uncertainty Evaluation in Engineering Measurements and Machine Learning

Miodrag Bolic

University of Ottawa

February 4, 2026

Outline

Notebook: ssm_examples.ipynb

Motivation: The Hidden Structure Problem

Key Questions

- How do we model systems where we only observe *part* of the state?
- How do we handle noisy observations?
- How do we incorporate domain knowledge about dynamics?

Answer (SSM idea): introduce a latent state $\mathbf{z}_t$ and specify a transition model $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ and measurement model $p(\mathbf{y}_t \mid \mathbf{z}_t)$.

Examples:

- Tracking an aircraft: observe radar position, but velocity is hidden
- Economic modeling: observe GDP, but structural factors are latent
- Robotics: observe sensor readings, but true robot state is uncertain

What Makes SSMs Powerful?

1. **Separation of concerns:** dynamics vs. observations
2. **Probabilistic framework:** principled uncertainty quantification
3. **Recursive inference:** efficient online algorithms (Kalman filter, particle filter)
4. **Generality:** unifies many models (ARIMA, HMMs, Dynamic Linear Models)

Core Insight

SSMs let us reason about *what we don't see* from *what we do see*

Classical Time Series: AR(p) Model

Autoregressive model of order p:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t \tag{1}$$

where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$

Limitations:

- Everything is directly observed
- No natural way to incorporate external inputs
- Difficult to handle missing data
- No distinction between "true state" and "noisy measurement"

This is a state space model!

AR(2) model:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2).$$

Step 1: Define the hidden state:

$$\mathbf{z}_t = \begin{bmatrix} y_t \\ y_{t-1} \end{bmatrix}.$$

Step 2: Transition (state dynamics):

$$\mathbf{z}_t = \underbrace{\begin{bmatrix} \phi_1 & \phi_2 \\ 1 & 0 \end{bmatrix}}_{F} \mathbf{z}_{t-1} + \mathbf{q}_t, \qquad \mathbf{q}_t = \begin{bmatrix} \epsilon_t \\ 0 \end{bmatrix}, \qquad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, Q),$$

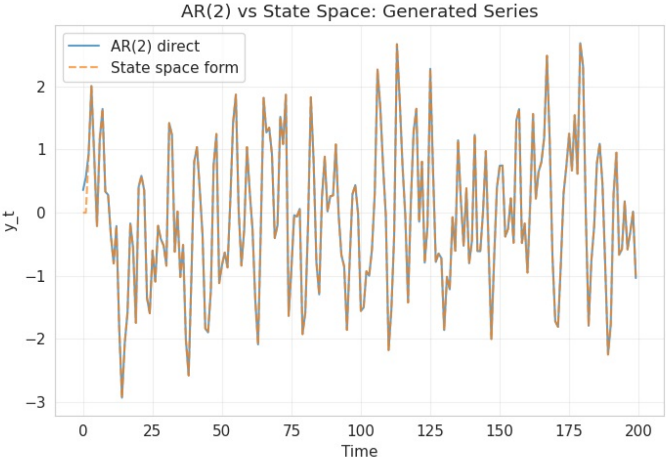
$$Q = \begin{bmatrix} \sigma_\epsilon^2 & 0 \\ 0 & 0 \end{bmatrix}.$$

Step 3: Observation equation:

$$y_t = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{H} \mathbf{z}_t + r_t, \qquad r_t \sim \mathcal{N}(0, R), \qquad R = 0$$

.

Converting AR(2) to State Space Form



General Linear State Space Model (SSM)

Transition Model (Dynamics)

$$\mathbf{z}_t = \mathbf{F}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{b}_t + \mathbf{q}_t, \quad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t) \tag{2}$$

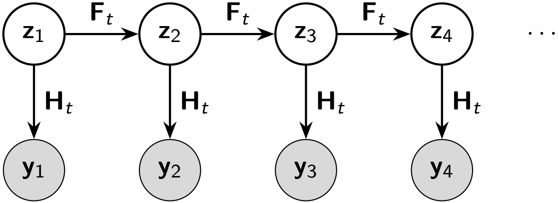
Observation Model (Measurement)

$$\mathbf{y}_t = \mathbf{H}_t \mathbf{z}_t + \mathbf{D}_t \mathbf{u}_t + \mathbf{d}_t + \mathbf{r}_t, \quad \mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t) \tag{3}$$

Notation (Murphy-style):

- $\mathbf{z}_t \in \mathbb{R}^{n_z}$: hidden state or latent variable
- $\mathbf{y}_t \in \mathbb{R}^{n_y}$: observation
- $\mathbf{u}_t \in \mathbb{R}^{n_u}$: control input
- $\mathbf{F}_t$: state transition matrix, $\mathbf{H}_t$: observation matrix
- $\mathbf{Q}_t$: process noise covariance, $\mathbf{R}_t$: measurement noise covariance

Graphical model



Latent states (white) evolve according to dynamics
Observations (gray) are noisy measurements of states

Continuous-Time State Space Model

Many physical systems are naturally described in continuous time:

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{A}_c\mathbf{z}(t) + \mathbf{B}_c\mathbf{u}(t) + \mathbf{w}(t) \tag{4}$$

$$\mathbf{y}(t) = \mathbf{C}_c\mathbf{z}(t) + \mathbf{v}(t) \tag{5}$$

Examples:

- Newton's laws: $\frac{d^2x}{dt^2} = F/m$
- RC circuits: $\frac{dV}{dt} = -\frac{1}{RC}V + \frac{1}{RC}V_{in}$
- Chemical reactions: $\frac{d[A]}{dt} = -k[A]$

Why Discretize?

Practical Reality:

- Computers operate in discrete time
- Sensors sample at fixed intervals (e.g., 100 Hz)
- Numerical algorithms need discrete representations

Our Goal: Transform continuous dynamics into discrete-time form:

$$\mathbf{z}_{t+1} = \mathbf{F}\mathbf{z}_t + \mathbf{B}\mathbf{u}_t + \mathbf{q}_t \tag{6}$$

Key Question

How do we choose **F** and **B** to best approximate continuous dynamics?

Method 1: Euler Discretization

Continuous: $\dot{\mathbf{z}}(t) = \mathbf{A}_c\mathbf{z}(t) + \mathbf{B}_c\mathbf{u}(t)$

Approximate derivative:

$$\frac{\mathbf{z}_{t+1} - \mathbf{z}_t}{\Delta t} \approx \mathbf{A}_c\mathbf{z}_t + \mathbf{B}_c\mathbf{u}_t \tag{7}$$

Rearrange:

$$\mathbf{z}_{t+1} = (\mathbf{I} + \Delta t \cdot \mathbf{A}_c)\mathbf{z}_t + \Delta t \cdot \mathbf{B}_c\mathbf{u}_t \tag{8}$$

Result: $\mathbf{F} = \mathbf{I} + \Delta t \cdot \mathbf{A}_c, \quad \mathbf{B} = \Delta t \cdot \mathbf{B}_c$

Pros: Simple, intuitive
Cons: Can be unstable for large Δt

Method 2: Exact Discretization (Matrix Exponential)

Solve the ODE exactly:

$$\mathbf{z}(t + \Delta t) = e^{\mathbf{A}_c \Delta t}\mathbf{z}(t) + \int_0^{\Delta t} e^{\mathbf{A}_c(\Delta t - \tau)}\mathbf{B}_c\mathbf{u}(t+\tau)d\tau \tag{9}$$

For constant input $\mathbf{u}(t) = \mathbf{u}_t$:

$$\mathbf{F} = e^{\mathbf{A}_c \Delta t} \tag{10}$$

$$\mathbf{B} = \mathbf{A}_c^{-1}(e^{\mathbf{A}_c \Delta t} - \mathbf{I})\mathbf{B}_c \tag{11}$$

Pros: Numerically stable, exact for LTI systems
Cons: Requires matrix exponential computation

In Python: use `scipy.signal.cont2discrete`

Example: Spring-Mass-Damper System

Continuous dynamics:

$$m\ddot{x} + c\dot{x} + kx = F(t) \tag{12}$$

State-space form: Let $\mathbf{z} = [x, \dot{x}]^T$

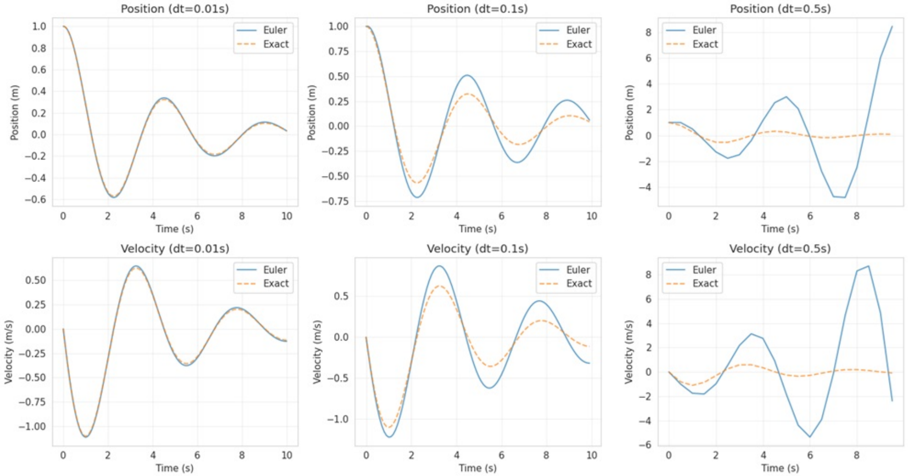
$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} F(t) \tag{13}$$

After discretization (see Python notebook for numerical values):

$$\mathbf{z}_{t+1} = \mathbf{F}\mathbf{z}_t + \mathbf{B}F_t \tag{14}$$

We'll implement this in the notebook!

Example: Euler Discretzation of Spring-Mass-Damper System



Latent Variables: What are they?

Definition (informal)

A **latent variable** is a variable in the model that is **not directly observed** but helps explain the observed data.

Model: observed $\mathbf{y}$ is generated from latent $\mathbf{z}$.

Why introduce them?

Latent variables let us represent **hidden structure**:

- **Clustering/segments**: which group generated the point?
- **Dynamics**: hidden state that evolves over time.
- **Missing/uncertain quantities**: treat unknowns as random variables.

Key idea

Latent variables are **random variables**. We infer them from data:

$$p(\mathbf{z} \mid \mathbf{y}) \quad \text{(posterior over latent variables).}$$

Latent Variables vs Parameters vs Hyperparameters

Three kinds of unknowns in ML models

- **Latent variables $\mathbf{z}$**: hidden *random* quantities that can vary across data points or time.
- **Parameters $\boldsymbol{\theta}$**: fixed (but unknown) model quantities (weights, matrices) that define distributions.
- **Hyperparameters $\boldsymbol{\lambda}$**: settings that control priors/regularization/model capacity (often chosen by CV or set a priori).

How they differ (quick checklist)

Question	Latent variables	Parameters	Hyperparameters
Vary per datapoint / time?	Yes	No	No
Random variables?	Yes	Bayesian: yes / Frequentist: fixed	Fixed
Typical objective / target	Infer $p(\mathbf{z} \mid \mathbf{y}, \boldsymbol{\theta})$	Learn $\hat{\boldsymbol{\theta}}$	Tune / set $\boldsymbol{\lambda}$

Typical inference targets (summary):

$$\underbrace{p(\mathbf{z} \mid \mathbf{y}, \boldsymbol{\theta})}_{\text{infer latent states}} \qquad \underbrace{\hat{\boldsymbol{\theta}}}_{\text{learn parameters}} \qquad \underbrace{\boldsymbol{\lambda}}_{\text{tune/set}}$$

Latent Variables in State Space Models (SSMs)

SSM viewpoint

In an SSM, the latent variable is the **hidden state sequence** $\mathbf{z}_{1:T} = (\mathbf{z}_1, \ldots, \mathbf{z}_T)$, and we observe measurements $\mathbf{y}_{1:T} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$.

What makes SSMs special? (time structure)

SSMs assume:

- **Markov dynamics**: $\mathbf{z}_t$ depends mainly on $\mathbf{z}_{t-1}$ (and input $\mathbf{u}_t$).
- **Conditional independence of observations**: $\mathbf{y}_t$ depends mainly on $\mathbf{z}_t$ (and $\mathbf{u}_t$).

These correspond to the standard transition density $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ and observation likelihood $p(\mathbf{y}_t \mid \mathbf{z}_t)$.

Typical inference tasks in SSMs: **Filtering**: $p(\mathbf{z}_t \mid \mathbf{y}_{1:t})$ (real-time state estimate)

SSMs in Probabilistic Notation

Dynamics as a conditional distribution

Your linear state update

$$\mathbf{z}_t = \mathbf{F}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{b}_t + \mathbf{q}_t, \qquad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$$

is equivalent to:

$$\boxed{p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_t) = \mathcal{N}\big(\mathbf{z}_t;\ \mathbf{F}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{b}_t,\ \mathbf{Q}_t\big)}$$

Measurement as a likelihood

Your observation model

$$\mathbf{y}_t = \mathbf{H}_t \mathbf{z}_t + \mathbf{D}_t \mathbf{u}_t + \mathbf{d}_t + \mathbf{r}_t, \qquad \mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$$

is equivalent to:

$$\boxed{p(\mathbf{y}_t \mid \mathbf{z}_t, \mathbf{u}_t) = \mathcal{N}\big(\mathbf{y}_t;\ \mathbf{H}_t \mathbf{z}_t + \mathbf{D}_t \mathbf{u}_t + \mathbf{d}_t,\ \mathbf{R}_t\big)}$$

SSM as a Full Probabilistic Model (Joint Factorization)

Generative story (with inputs)

Given inputs $\mathbf{u}_{1:T}$:

1. Sample initial state: $\mathbf{z}_0 \sim p(\mathbf{z}_0)$
2. For $t = 1, \ldots, T$:
   - Sample state: $\mathbf{z}_t \sim p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_t)$
   - Sample observation: $\mathbf{y}_t \sim p(\mathbf{y}_t \mid \mathbf{z}_t, \mathbf{u}_t)$

Joint distribution (key SSM factorization)

$$\boxed{p(\mathbf{z}_{0:T}, \mathbf{y}_{1:T} \mid \mathbf{u}_{1:T}) = p(\mathbf{z}_0) \prod_{t=1}^{T} p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_t) \prod_{t=1}^{T} p(\mathbf{y}_t \mid \mathbf{z}_t, \mathbf{u}_t)}$$

Latent Variables Beyond State Space Models

Latent-variable models appear everywhere

- **Mixture models**: latent cluster z_n explains multimodal data.
- **Topic models**: latent topic proportions explain documents.
- **Factor analysis / PCA**: latent factors explain correlations in features.
- **VAEs**: latent code explains data via a neural decoder.
- **Missing data**: treat missing entries as latent variables and infer them.

What we do with latent variables (common inference patterns)

- **Posterior inference**: compute/approximate $p(\mathbf{z} \mid \mathbf{y})$
- **Learning parameters**: maximize $\log p(\mathbf{y} \mid \boldsymbol{\theta})$ or do Bayesian learning
- **Approximate inference**: variational methods, MCMC, particle methods (when exact is hard)

Takeaway

Classic Applications

1. **Tracking and Navigation**
   - GPS receivers: fuse satellite measurements with motion model
   - Aircraft/ship tracking: noisy radar + physics-based dynamics
2. **Signal Processing**
   - Noise reduction in sensor data
   - Smoothing and interpolation
3. **Control Systems**
   - State estimation for feedback control
   - Model Predictive Control (MPC)
4. **Econometrics**
   - Dynamic factor models
   - Estimating unobserved economic indicators

Key Takeaways

1. **SSMs separate dynamics from observations**—models what we believe happens vs. what we see
2. **Time series → SSM:** Add latent states to capture hidden structure
3. **Continuous → Discrete:** Use matrix exponential or Euler for principled discretization
4. **Kalman filter:** Optimal recursive Bayesian inference for linear-Gaussian case
5. **Extensions:** EKF, UKF, particle filters, deep SSMs bridge classical and modern ML

Philosophy

SSMs are about building models with *inductive biases*—using domain knowledge about dynamics and measurement to constrain learning

Acknowledgements/ Document preparation

This document was developed and converted into LaTeX slides and formatted with assistance from ChatGPT (OpenAI) and CoPilot (Microsoft). The instructor modified the source text and verified the final structure and wording.