Representation and Estimation of the Hyperstate in Dual Control

Friday Seminar

2021-02-26

Christian Rosdahl

Dual Control

- **Goal**: Control dynamical system with uncertainties
- **Certainty Equivalence Control**
  - Requires good parameter estimates
  - No probing
- **Dual Control**:
  Find good long-term control by balancing **exploration and exploitation**

General problem

Given system:

state input

unknown parameters

noises

$$x_{k+1} = f(x_k, u_k, \theta_k, w_k)$$

$$y_k = g(x_k, e_k)$$

measurement signal

Goal: Select u_k to minimize cost J, using $\mathcal{D}_k = \{y_k, u_{k-1}, y_{k-1}, u_{k-2}, \dots\}$

Hyperstate: $\xi_k = \mathbb{P}(x_k, \theta_k \mid \mathcal{D}_k)$

Problem: Find control law $u_k = \pi(\xi_k)$ to minimize cost

$$J(u_{k:k+T-1}, \xi_k) = \mathbb{E}\left\{ \sum_{j=k}^{k+T-1} c(x_{j+1}, u_j) \,\middle|\, \xi_k \right\}$$

Bellman Equation

Problem: Find control law $u_k = \pi(\xi_k)$ to minimize cost J

Value function:

$$V_\tau(\xi_k) = \min_{u_k,\ldots,u_{k+\tau-1}} \mathbb{E}\left\{ \sum_{j=k}^{k+\tau-1} c(x_{j+1}, u_j) \,\middle|\, \xi_k \right\}$$

Action-value function:

$$Q_T(\xi_k, u_k) := \mathbb{E}\{c(x_{k+1}, u_k) + V_{T-1}(\xi_{k+1}) \mid \xi_k\}$$

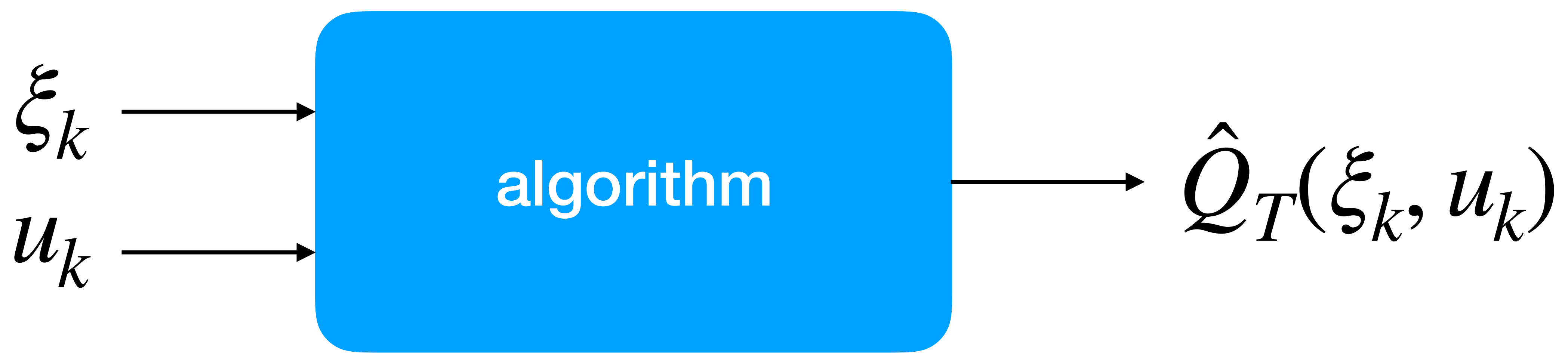
Bellman Equation: $V_\tau(\xi_k) = \min_{u_k} \mathbb{E}\{c(x_{k+1}, u_k) + V_{\tau-1}(\xi_{k+1}) \mid \xi_k\}, \quad \tau = 1,2,\ldots,T$

Solution: $u_k = \pi(\xi_k) := \operatorname*{argmin}_{u_k} Q_T(\xi_k, u_k)$

But the Bellman Equation can't be solved exactly.

End-Goal

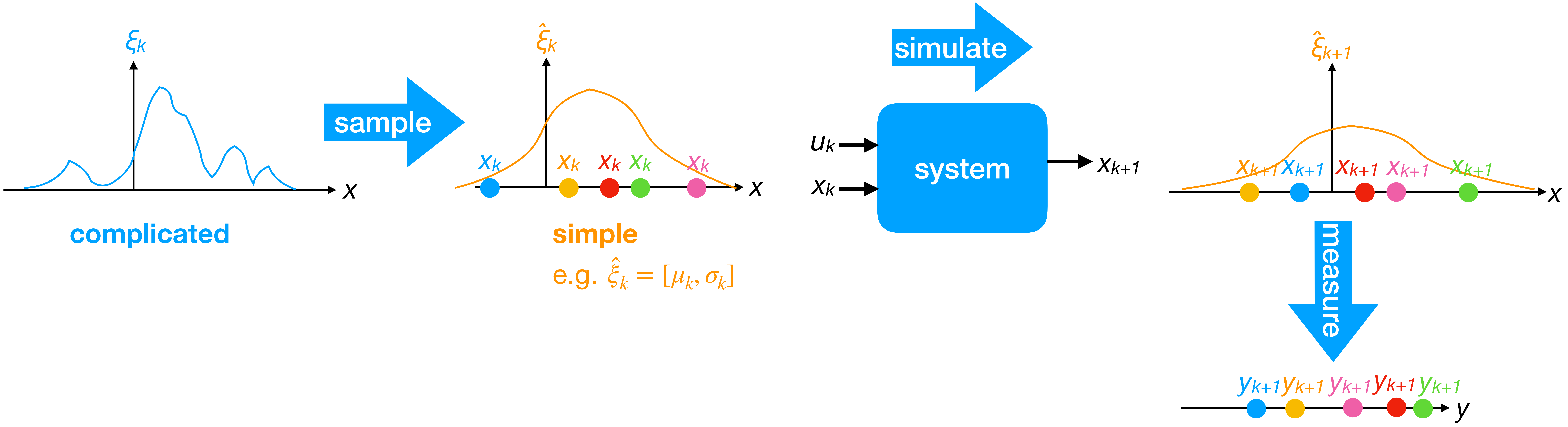
Find algorithm to approximate $Q_T(\xi_k, u_k)$ from ξ_k and u_k



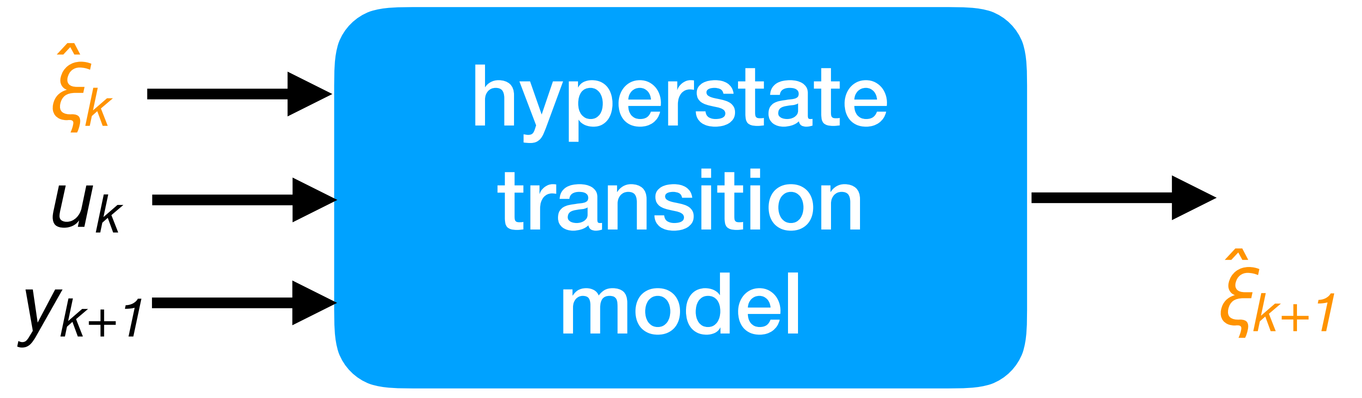
Input can then be chosen as $u_k = \pi(\xi_k) := \underset{u_k}{\operatorname{argmin}} \hat{Q}_T(\xi_k, u_k)$

Hyperstate transition model

We need a model for $\xi_k \rightarrow \xi_{k+1}$



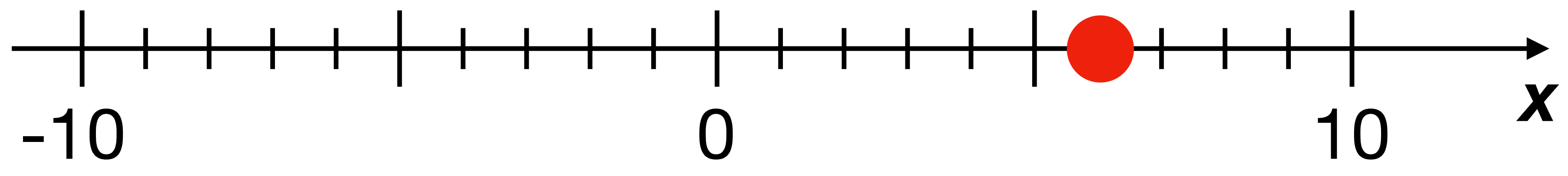
Model:

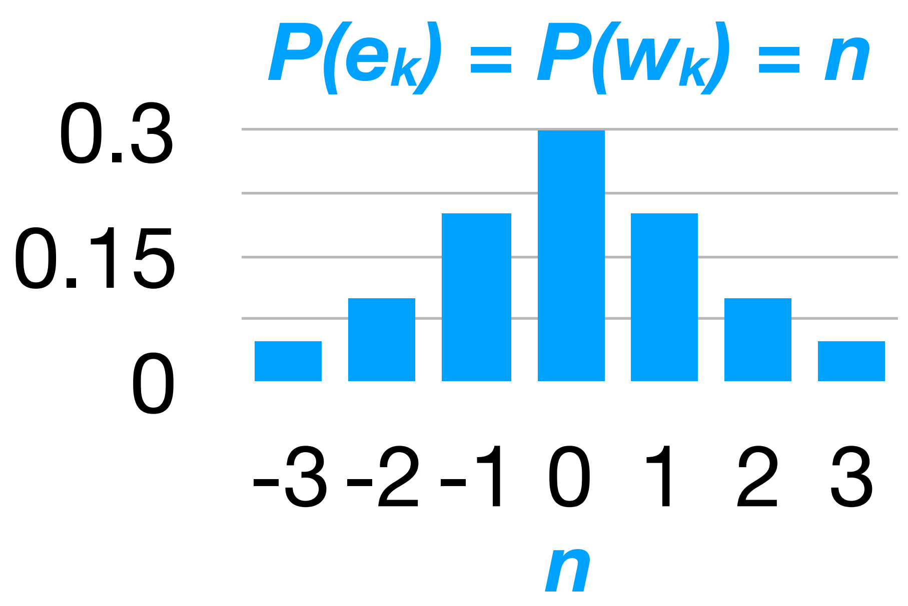


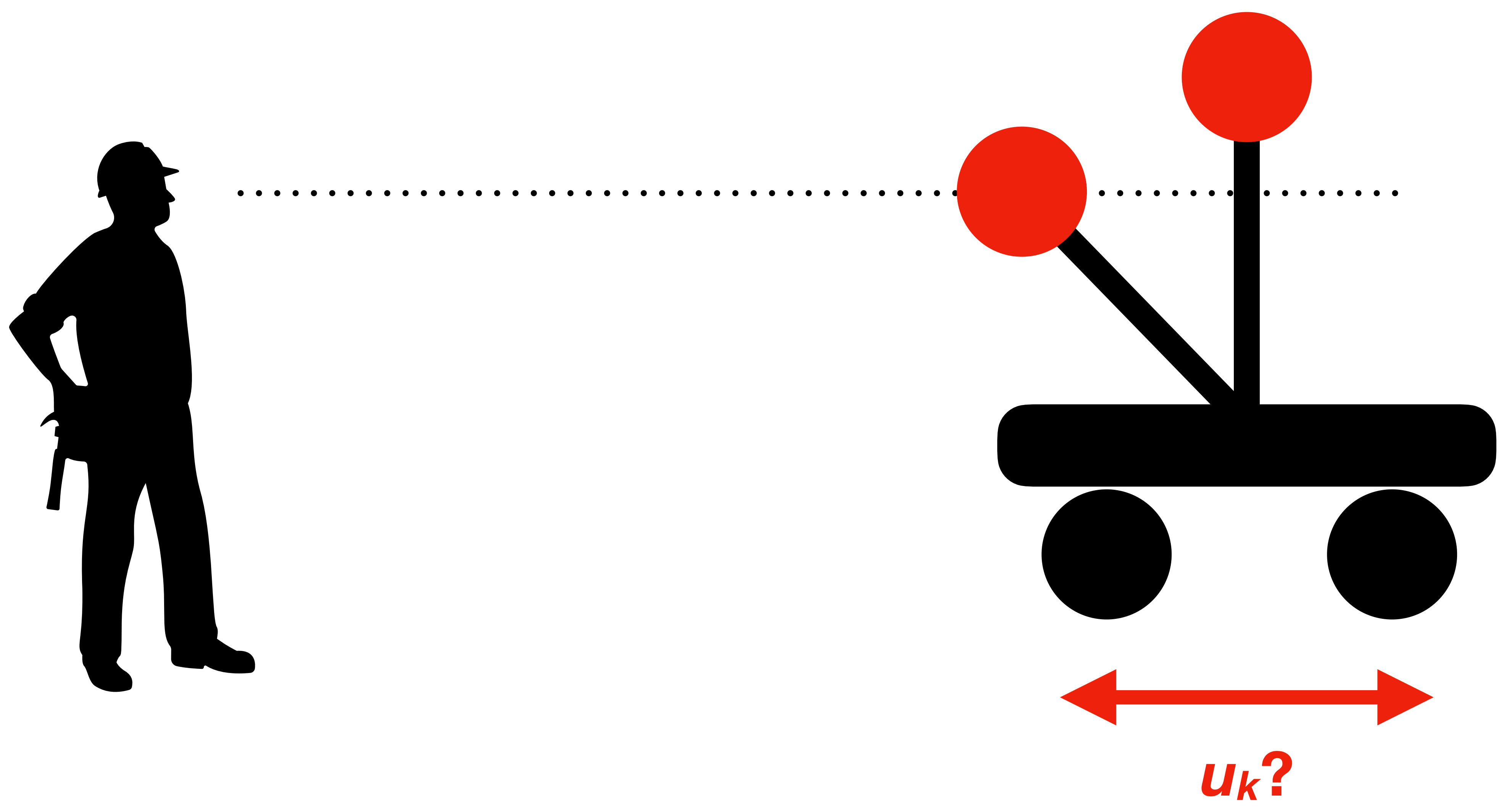
Example system

$$x_{k+1} = \mathrm{sat}_{10}(x_k + u_k + w_k)$$
$$y_k = |x_k| + e_k,$$

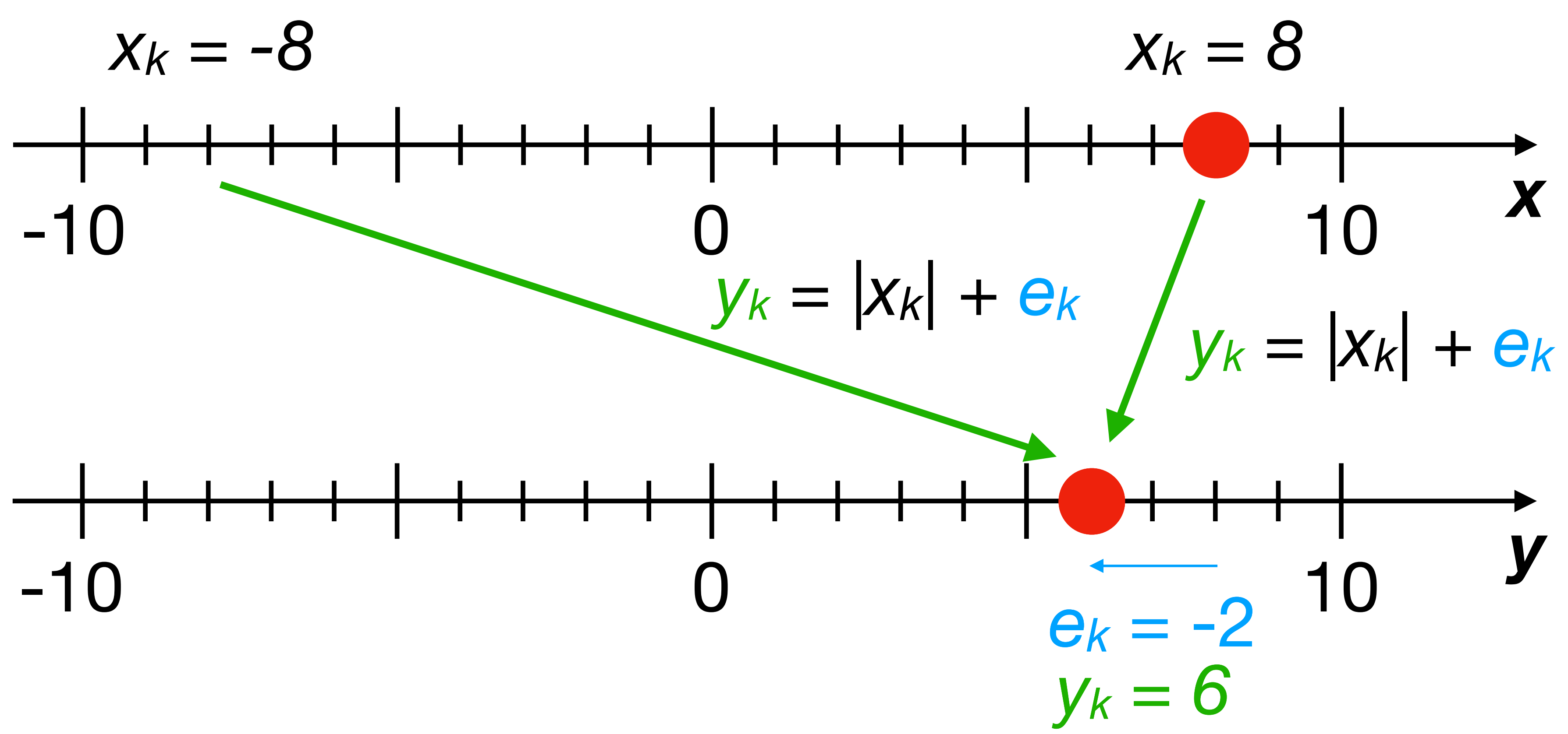
all signals are integers



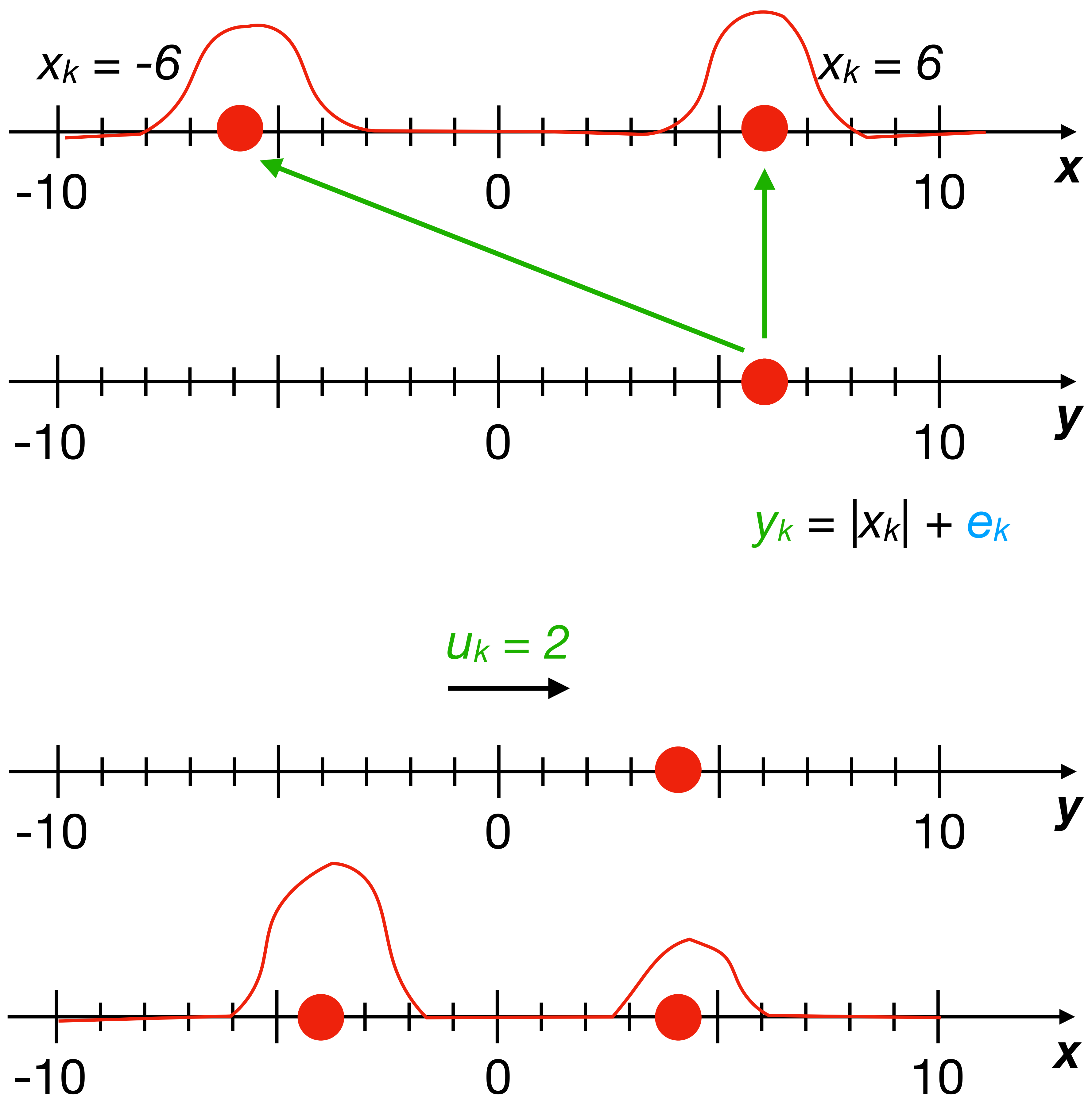




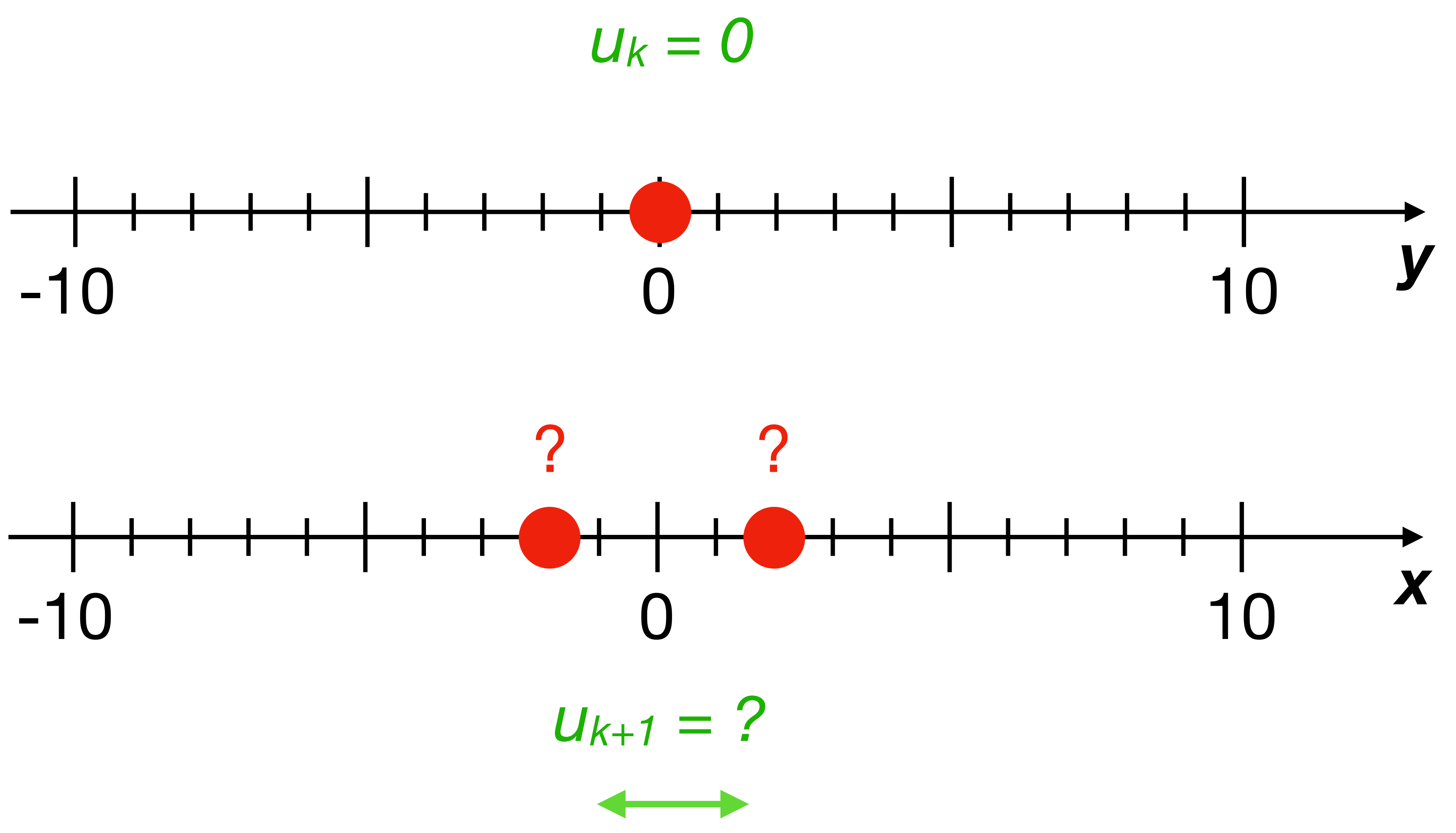
Example system: x_k to y_k



Example system: y_k to x_k



Example system: y_k to x_k



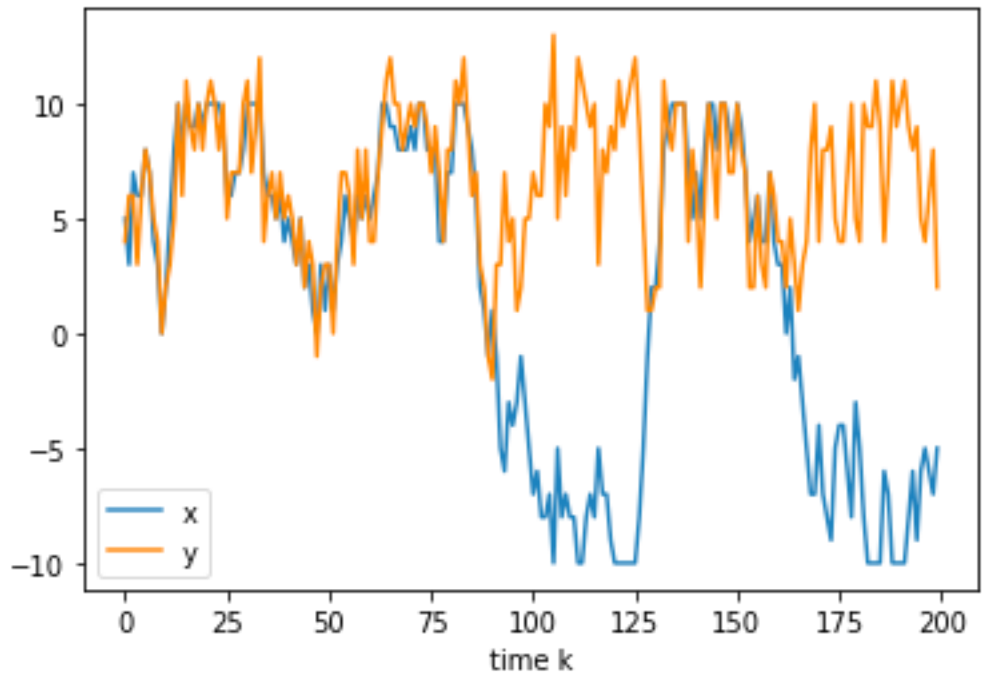
Hyperstate representation

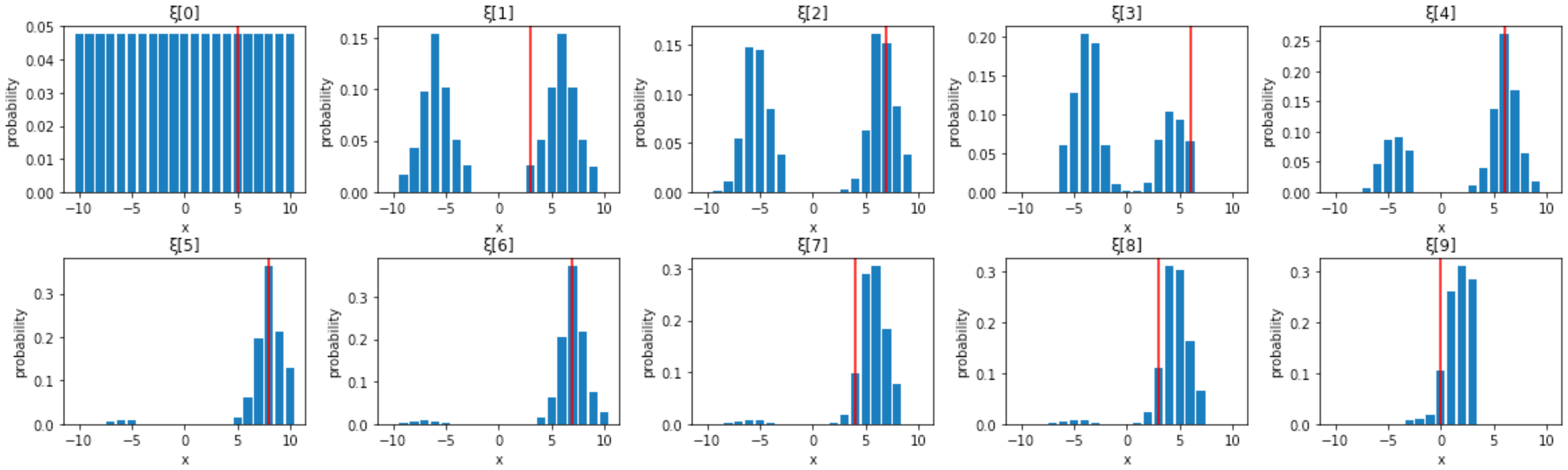
Gaussian mixture model

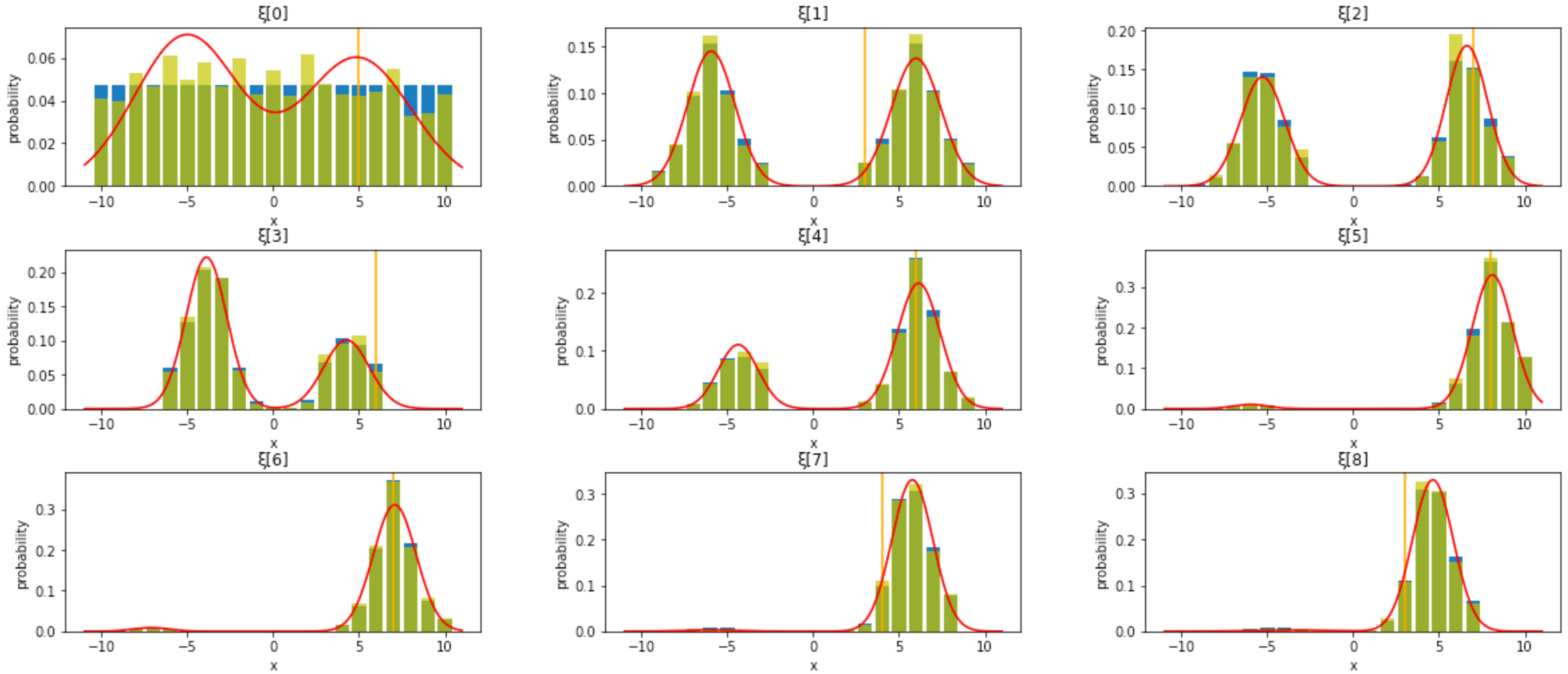
$$\hat{\xi}_k(x;\lambda,\mu,\sigma) = \lambda f(x;\mu_1,\sigma_1) + (1-\lambda)f(x;\mu_2,\sigma_2),$$ where $$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$

Can be represented by the vector $\hat{\xi}_k = [\lambda, \mu_1, \mu_2, \sigma_1, \sigma_2]$

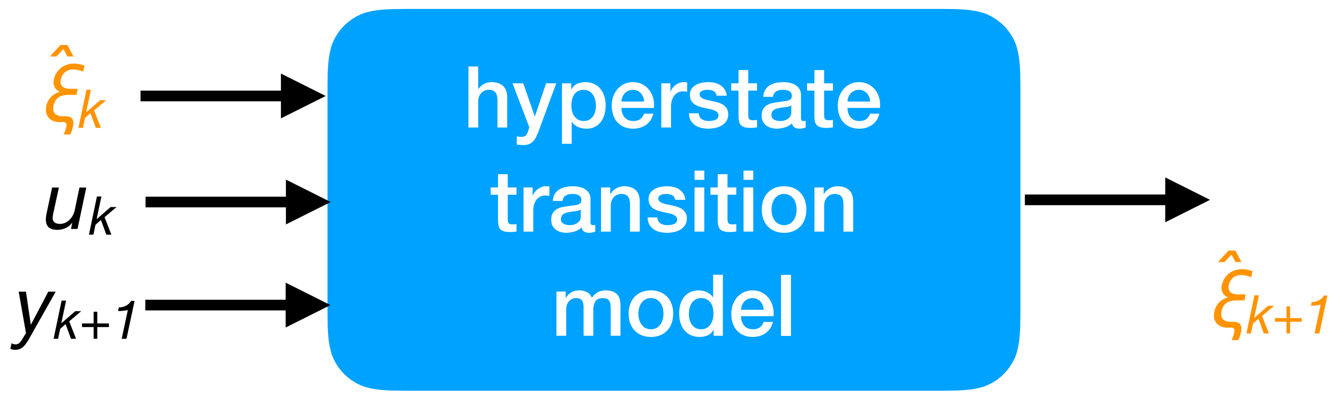
Simulation


x
y
time k


ξ[0]
ξ[1]
ξ[2]
ξ[3]
ξ[4]
ξ[5]
ξ[6]
ξ[7]
ξ[8]
ξ[9]
probability
x


ξ[0]
ξ[1]
ξ[2]
ξ[3]
ξ[4]
ξ[5]
ξ[6]
ξ[7]
ξ[8]
probability
x

Hyperstate transition model

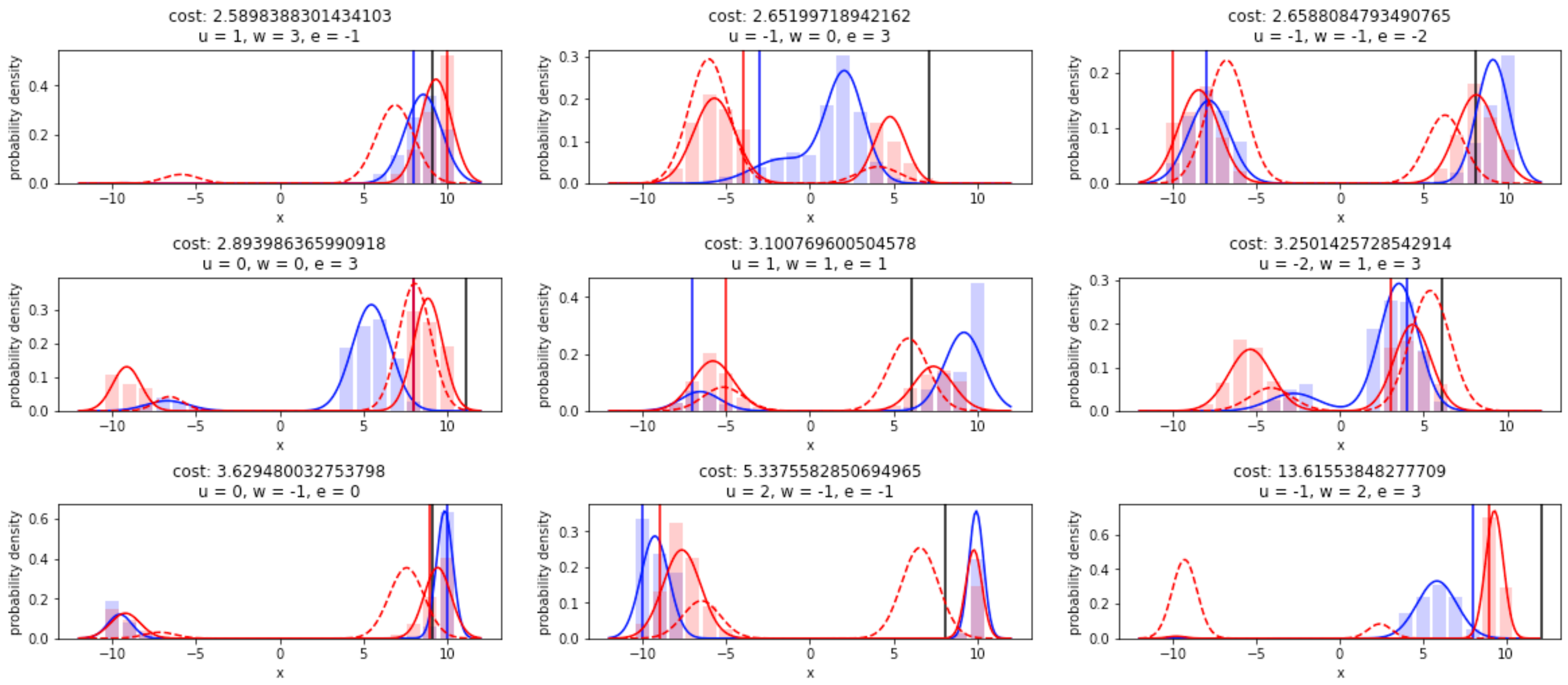


$$\hat{\xi}_k = [\lambda, \mu_1, \mu_2, \sigma_1, \sigma_2]$$

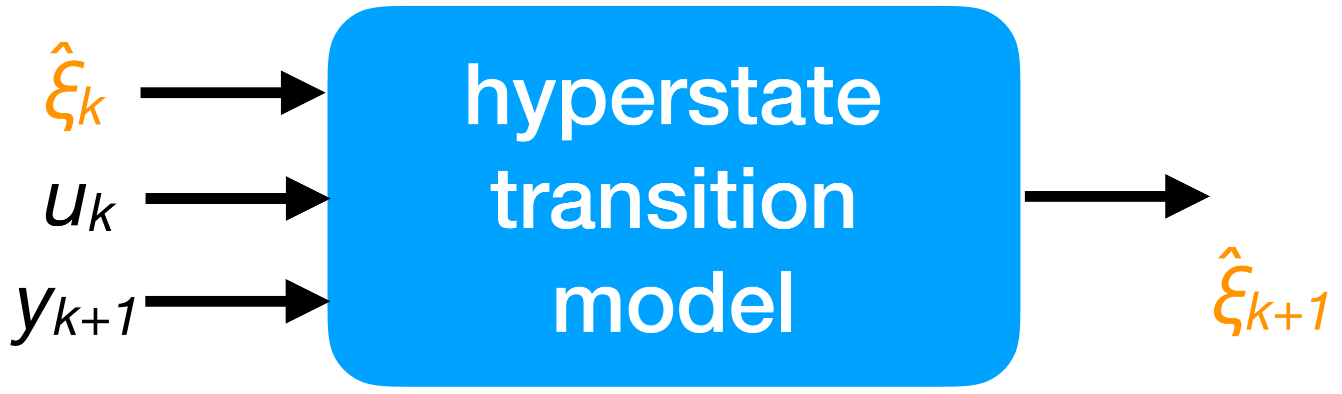
NN with 5 dense layers, 64-node layers

Trained with 5600 samples

Worst cases



Hyperstate transition model

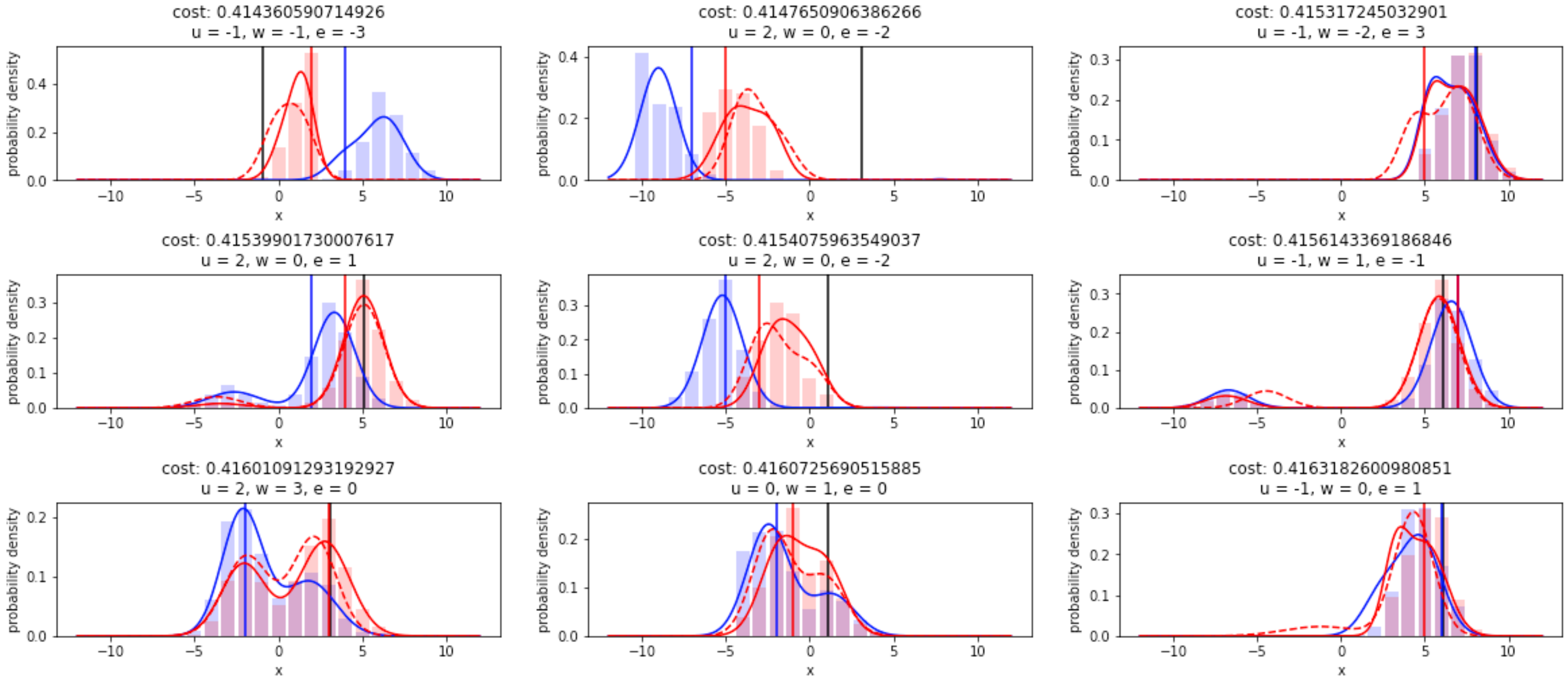


$$\hat{\xi}_k = [\lambda, \mu_1, \mu_2, \sigma_1, \sigma_2]$$

NN with 5 dense layers, 64-node layers

Trained with 5600 samples

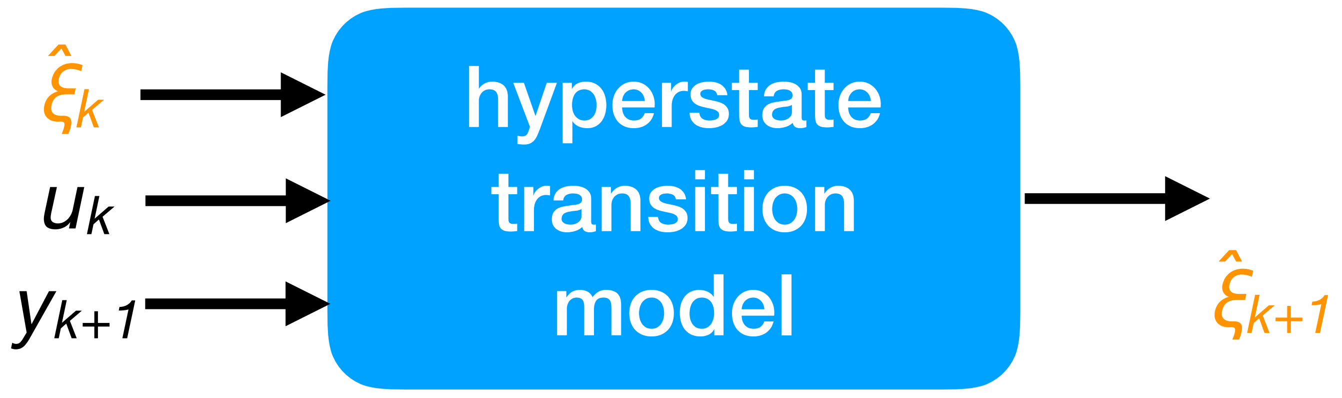
Average cases

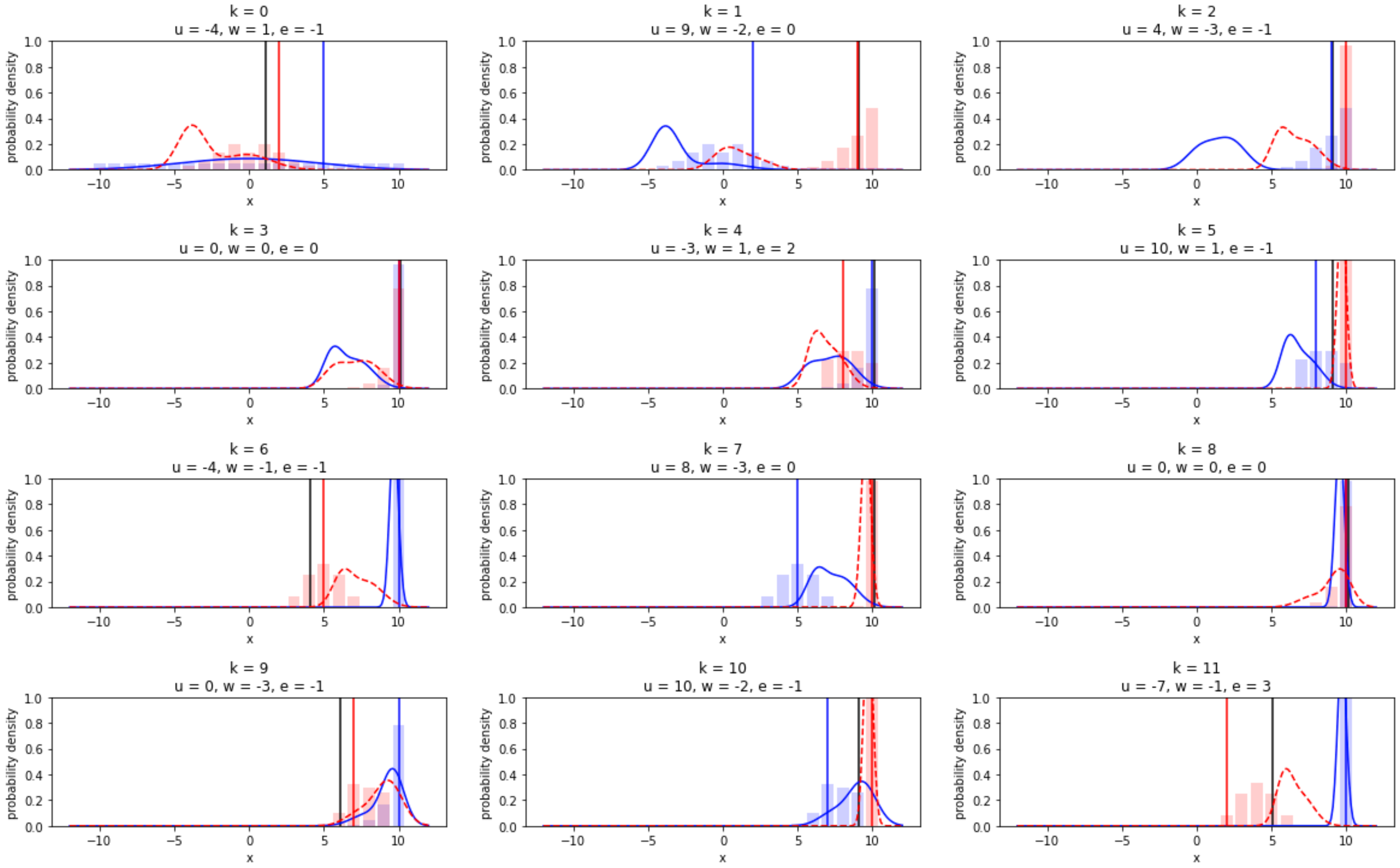


Problem with transition model

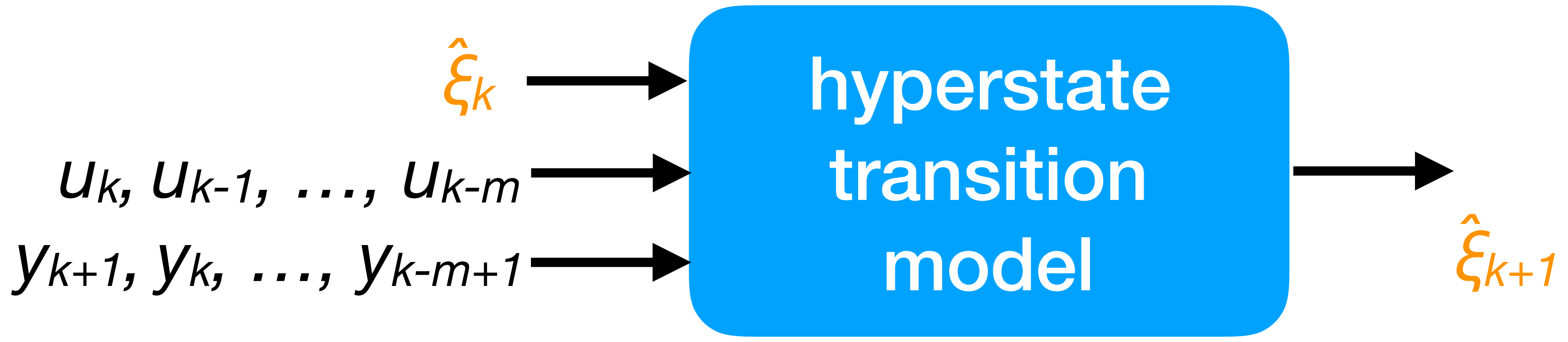
$$\hat{\xi}_k = [\lambda, \mu_1, \mu_2, \sigma_1, \sigma_2]$$

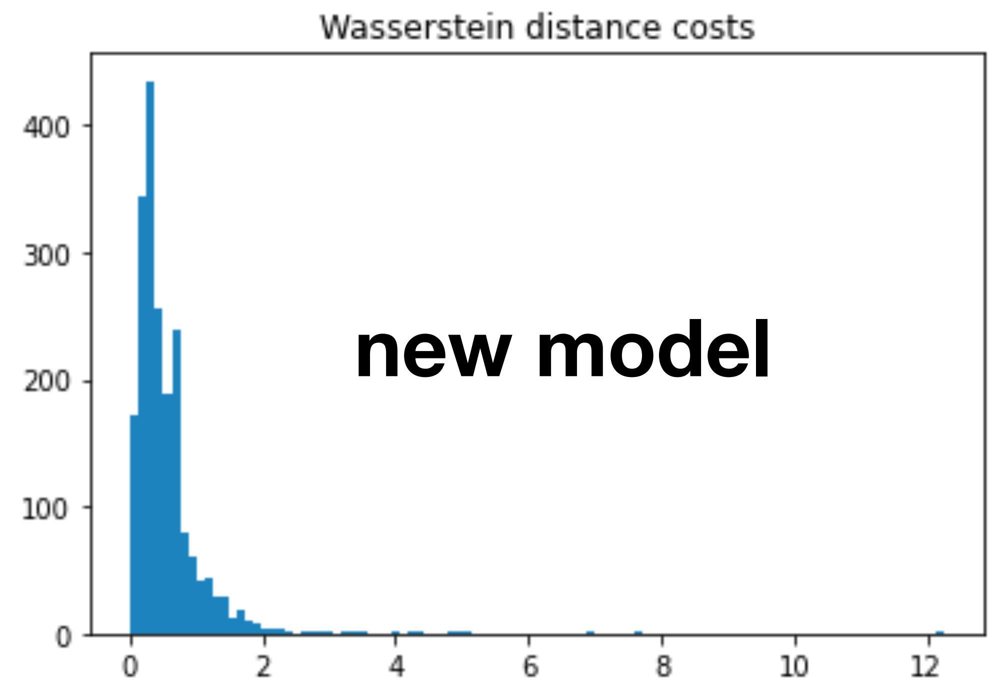
is not exactly known. Previous estimation from model is used.

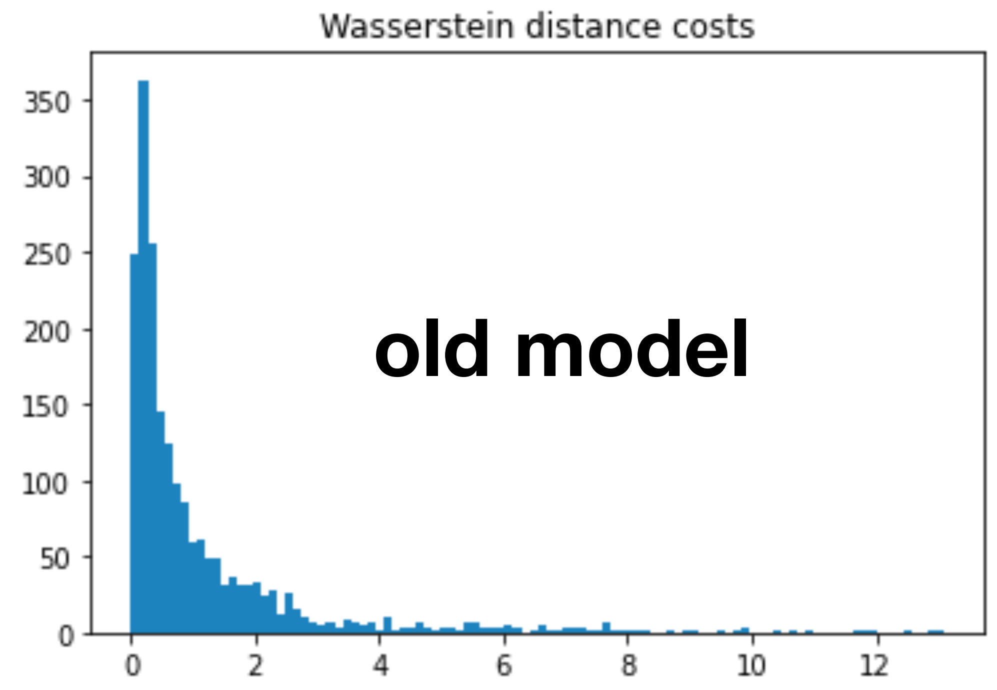


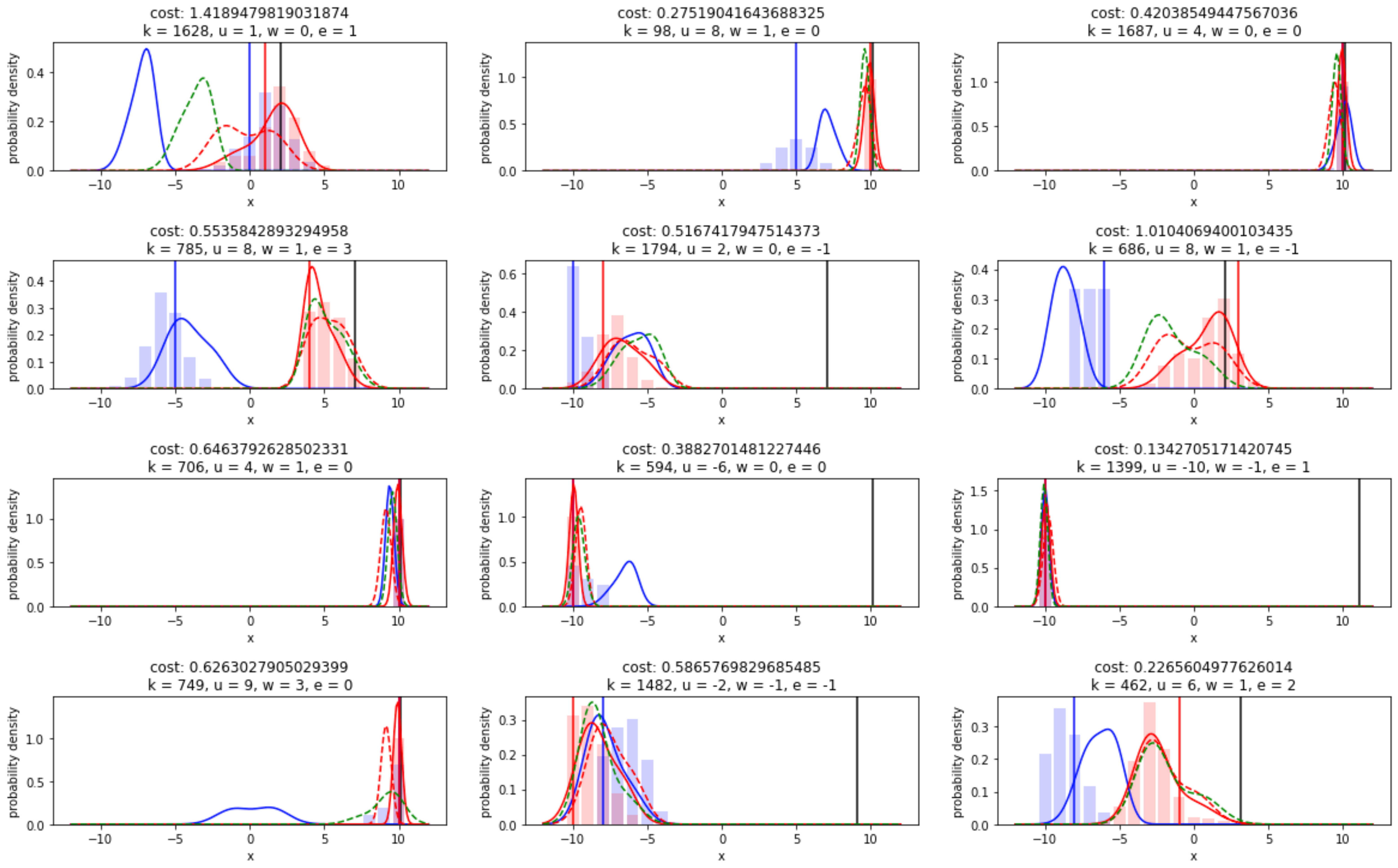


Improved hyperstate model



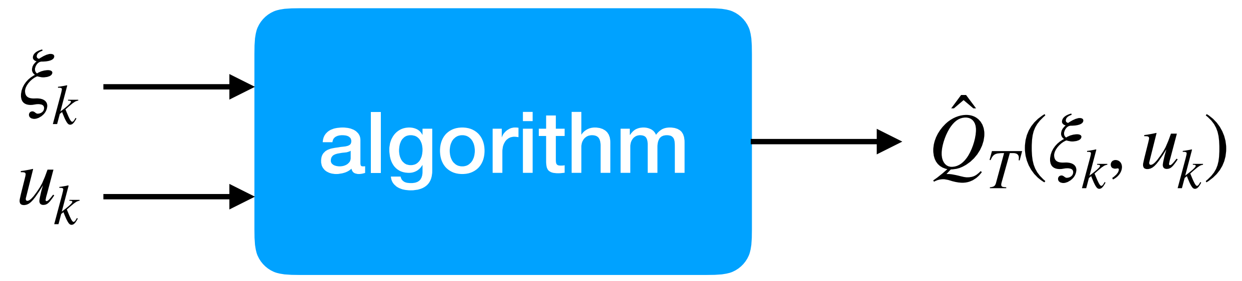




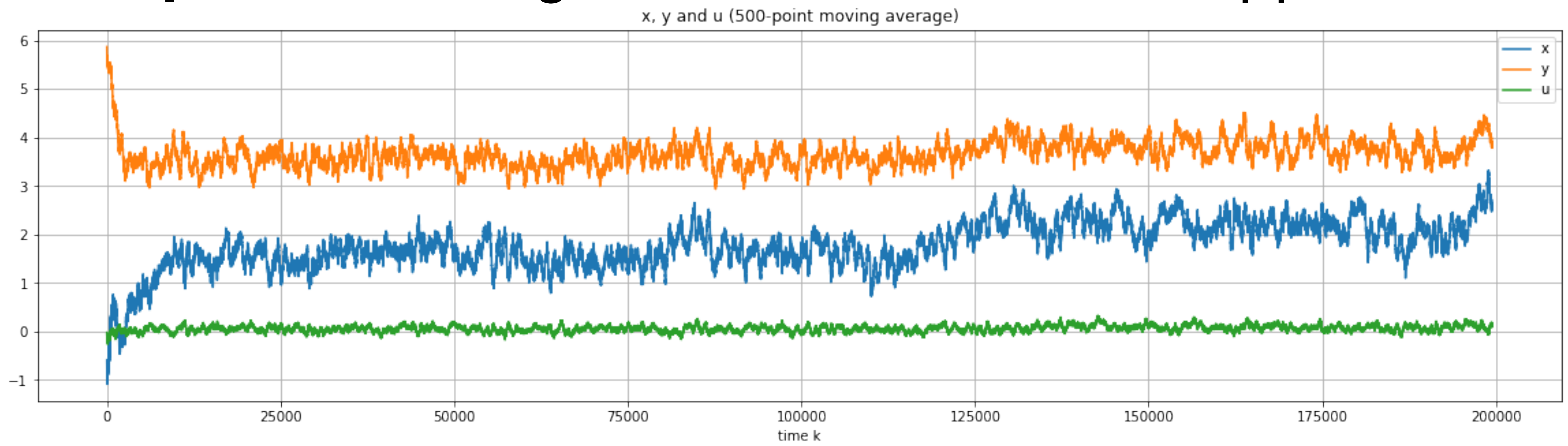


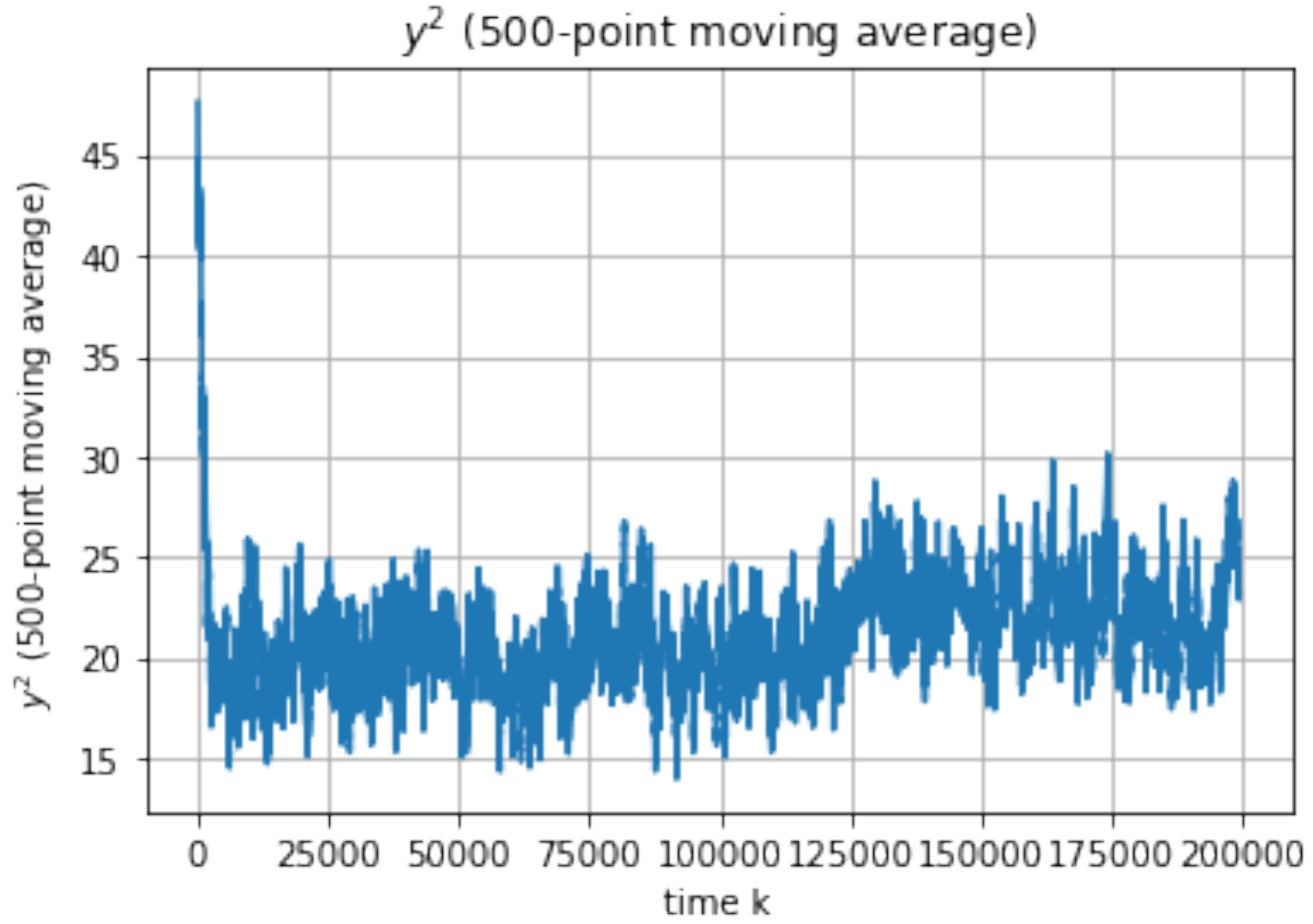
Reinforcement learning

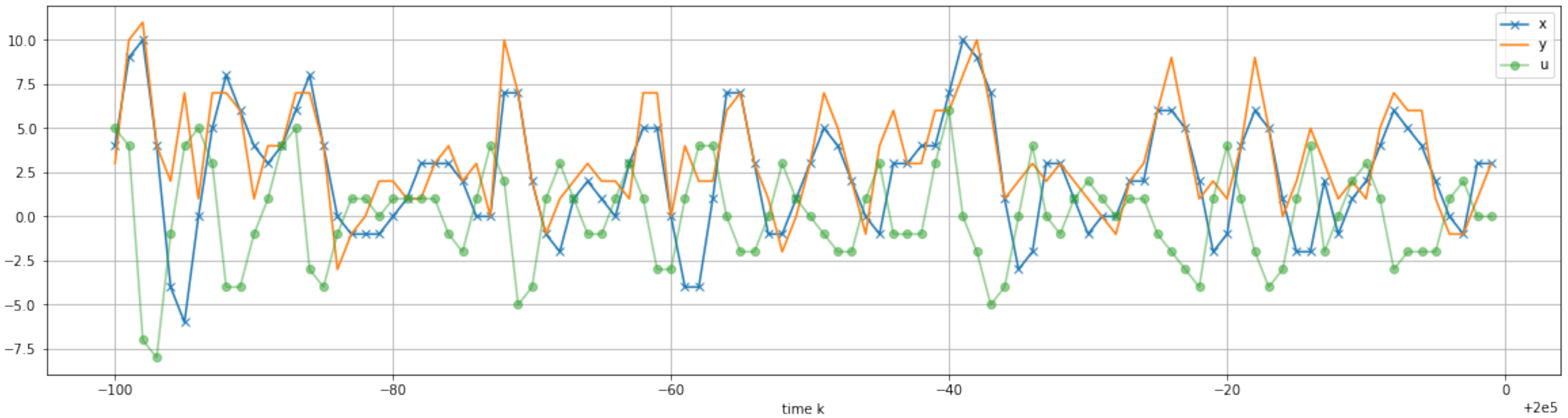
Idea: Use RL with hyperstate estimate $\hat{\xi}_k$ as state to find $\hat{Q}_T(\xi_k, u_k)$

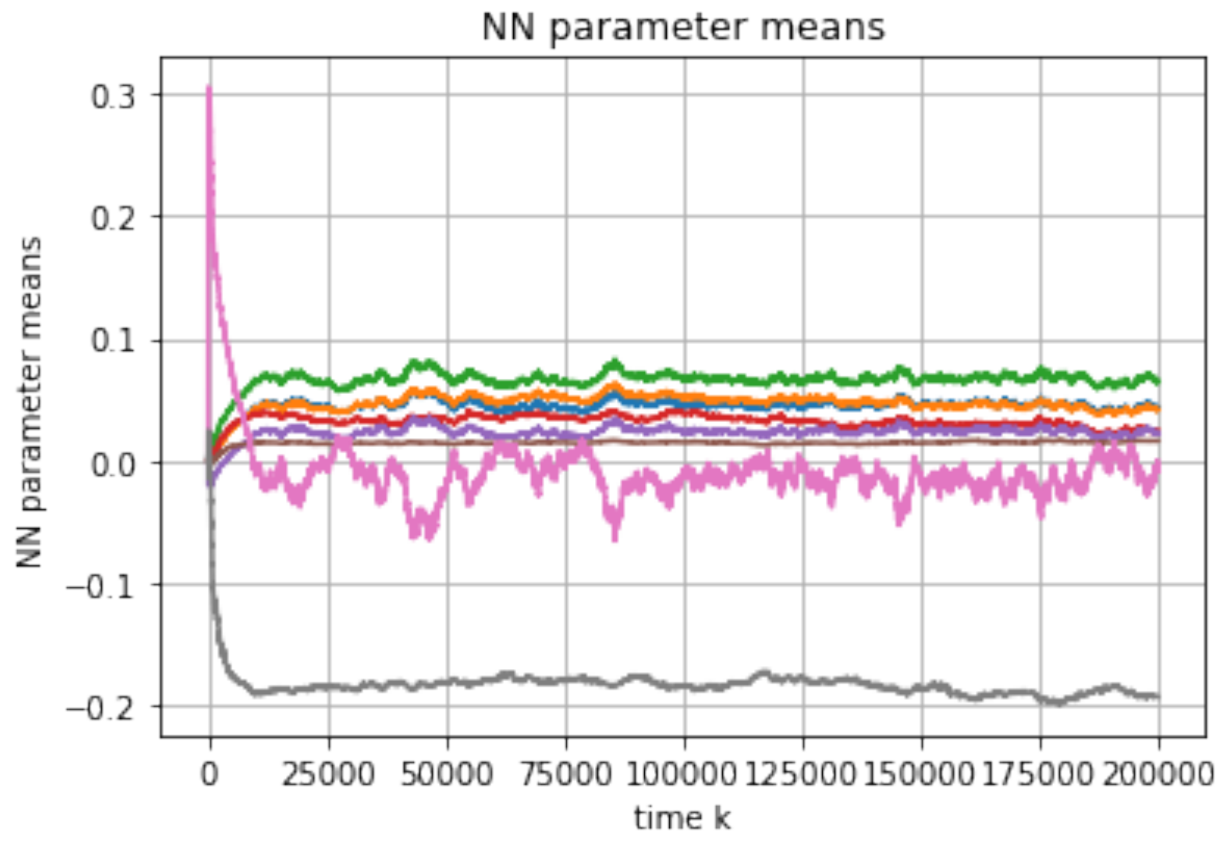


Example: Q-learning with NN as Q-function approximator









Reinforcement learning problems

- Moving target: use *Q*-values to approximate new *Q*-values
- Many tricks needed
- Many parameters to choose

Some nice work tools

For taking notes:

https://www.notion.so/

For version control with Jupyter:

https://github.com/mwouts/jupytext