Quantization Aligned with LQR Lyapunov Geometry for Networked Control

TODO

***Abstract*— TODO**

***Index Terms*— TODO**

I. Introduction

MULTI-agent systems (MAS) ...

II. Notations and Necessary Tools

TODO

III. Problem Statement

We consider the discrete-time linear time-invariant (LTI) system

$$x_{k+1} = Ax_k + Bu_k, \qquad u_k = -K\hat{x}_k, \tag{1}$$

where $x_k \in \mathbb{R}^n$ is the state, (A, B) is stabilizable, and K is a stabilizing state-feedback gain. The controller receives a quantized state $\hat{x}_k = Q(x_k)$ transmitted over a rate-limited channel.

Objective: Given a fixed bit budget, design a quantization strategy $Q(\cdot)$ that minimizes the degradation of closed-loop performance relative to the ideal (unquantized) controller.

Key question: How should quantization resolution be distributed in the state space *given the closed-loop dynamics induced by the control law*?

IV. Closed-Loop Geometry and Performance Metric

Let $Q \succeq 0$ and $R \succ 0$ be given weighting matrices, and let $S \succ 0$ denote the solution of the discrete-time algebraic Riccati equation associated with (A, B, Q, R). The quadratic function

$$V(x) = x^\top Sx \tag{2}$$

has two fundamental interpretations:

- it is the optimal infinite-horizon LQR value function;
- it is a Lyapunov function for the closed-loop system $x_{k+1} = (A - BK)x_k$.

As a consequence, state estimation errors are not equally important in Euclidean coordinates. The control-relevant distortion induced by quantization is naturally measured by the S-metric

$$d_S(x, \hat{x}) = (x - \hat{x})^\top S(x - \hat{x}). \tag{3}$$

This work was supported by project DECIDE, no. 57/14.11.2022 funded under the PNRR I8 scheme by the Romanian Ministry of Research, Innovation, and Digitisation.

TODO

Implication: Standard uniform or logarithmic quantizers, which allocate resolution based on Euclidean amplitude and axis-aligned coordinates, are generally mismatched to the closed-loop geometry encoded by S.

V. Proposed Idea: LQR-Shaped Quantization

Let $S = L^\top L$ be a Cholesky factorization of the Riccati matrix. Define the shaped coordinates

$$z = Lx. \tag{4}$$

In these coordinates, the value function becomes isotropic:

$$V(x) = x^\top Sx = \|z\|_2^2. \tag{5}$$

Design principle:

- transform the state using $z = Lx$;
- apply a quantizer that minimizes Euclidean distortion in z;
- reconstruct $\hat{x} = L^{-1}\hat{z}$ at the controller.

This construction yields a *control-aware quantizer* whose resolution is aligned with the LQR value geometry.

VI. Algorithm: LQR-Shaped Product K-Means

Offline training:

1) simulate closed-loop trajectories under ideal state feedback;
2) collect samples of shaped states $z_k = Lx_k$;
3) train independent one-dimensional K-means codebooks for each coordinate of z under a fixed bit budget.

Online operation:

$$\hat{z}_{k,i} = \arg\min_{c \in \mathcal{C}_i} |z_{k,i} - c|, \qquad \hat{x}_k = L^{-1}\hat{z}_k, \tag{6}$$

where $\mathcal{C}_i$ denotes the learned codebook for the ith shaped coordinate.

Baselines for comparison:

- uniform scalar quantization;
- logarithmic (μ-law) companding;
- unshaped (Euclidean) K-means quantization.

VII. Performance Analysis

Theorem 1 (Excess LQR cost induced by quantization): Let $A_c = A - BK$ be Schur and let $S \succ 0$ be the Riccati solution. Consider the quantized-feedback system $u_k = -K\hat{x}_k$ with quantization error $e_k = x_k - \hat{x}_k$. Then the excess LQR cost relative to the ideal controller satisfies

$$\Delta J = \sum_{k=0}^{\infty} e_k^\top \Xi e_k, \qquad \Xi \succeq 0, \tag{7}$$

and there exist constants $\underline{\alpha}, \bar{\alpha} > 0$ such that

$$\underline{\alpha} \sum_{k=0}^{\infty} \|e_k\|_S^2 \;\leq\; \Delta J \;\leq\; \bar{\alpha} \sum_{k=0}^{\infty} \|e_k\|_S^2. \tag{8}$$

Interpretation: Closed-loop performance degradation is governed by the accumulation of quantization errors measured in the S-metric. Reducing $\|e_k\|_S^2$ directly reduces excess LQR cost.

Corollary 1 (Uniform quantization): Uniform scalar quantization yields bounded Euclidean error but does not control distortion in the S-metric, resulting in conservative bounds on ΔJ proportional to $\lambda_{\max}(S)$.

Corollary 2 (μ-law companding): μ-law quantization increases resolution near the origin in each coordinate, but remains axis-aligned and does not account for state coupling encoded by S.

Corollary 3 (Cholesky shaping): In shaped coordinates $z = Lx$, Euclidean distortion equals S-metric distortion, i.e., $\|e_k\|_S^2 = \|e_k^z\|_2^2$.

Corollary 4 (Shaped K-means): Product K-means quantization in shaped coordinates minimizes an empirical approximation of $\mathbb{E}\|e_k\|_S^2$, and therefore reduces $\mathbb{E}\Delta J$ when training and operating distributions are matched.

VIII. Case Study

A two-dimensional unstable LTI plant with LQR feedback is considered under a fixed bit budget. Monte Carlo simulations demonstrate that LQR-shaped K-means quantization consistently achieves lower excess LQR cost than uniform, μ-law, and unshaped K-means quantizers. Geometric visualizations confirm alignment between quantization cells and Lyapunov level sets.

IX. Conclusion

This work shows that quantization should be designed in the geometry induced by the closed-loop control objective. Riccati-based shaping provides a principled metric, while data-driven quantization adapts resolution to the relevant state distribution without sacrificing stability guarantees. The approach naturally bridges optimal control, learning, and networked systems, making it well suited for L-CSS/CDC dissemination.

Experiments and conclusions:

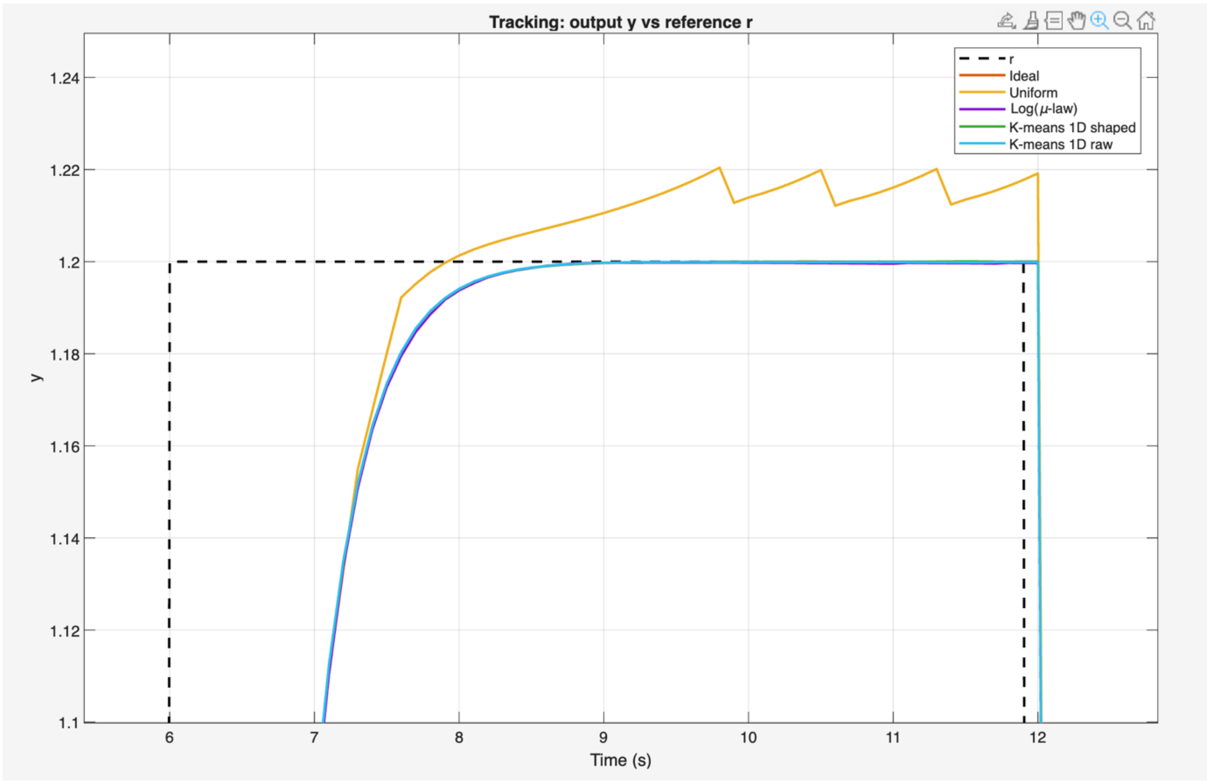
1) Comparison of 2 x 1D Kmeans (one for each state signal) versus 2D Kmeans (meaning to fit a 2D grid from trajectories, which suffers from the curse of dimensionality, takes a lot of time and does not work well) => using separate 1D Kmeans for each state is a lot better

2) For state stabilization, x -> 0, the Kmeans tends to converge to a logarithmic quantizer, as for closed-loop trajectories, which are based on exponentials, more or less, it tends to be the best quantizer to "compensate" the exponentials to logarithms;

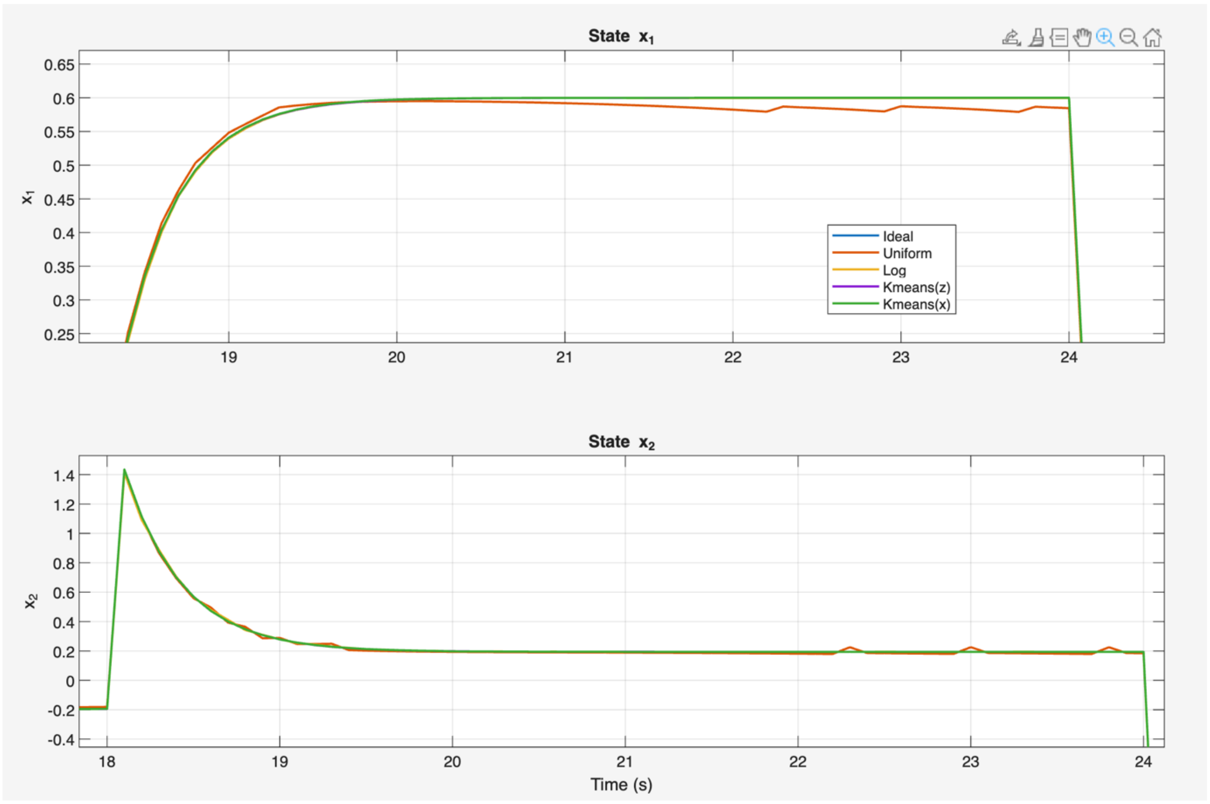
3) I think the Kmeans can be made more efficient from a number of bits' point of view

From the MATLAB file experiment_v2_exploit_comparison.m:

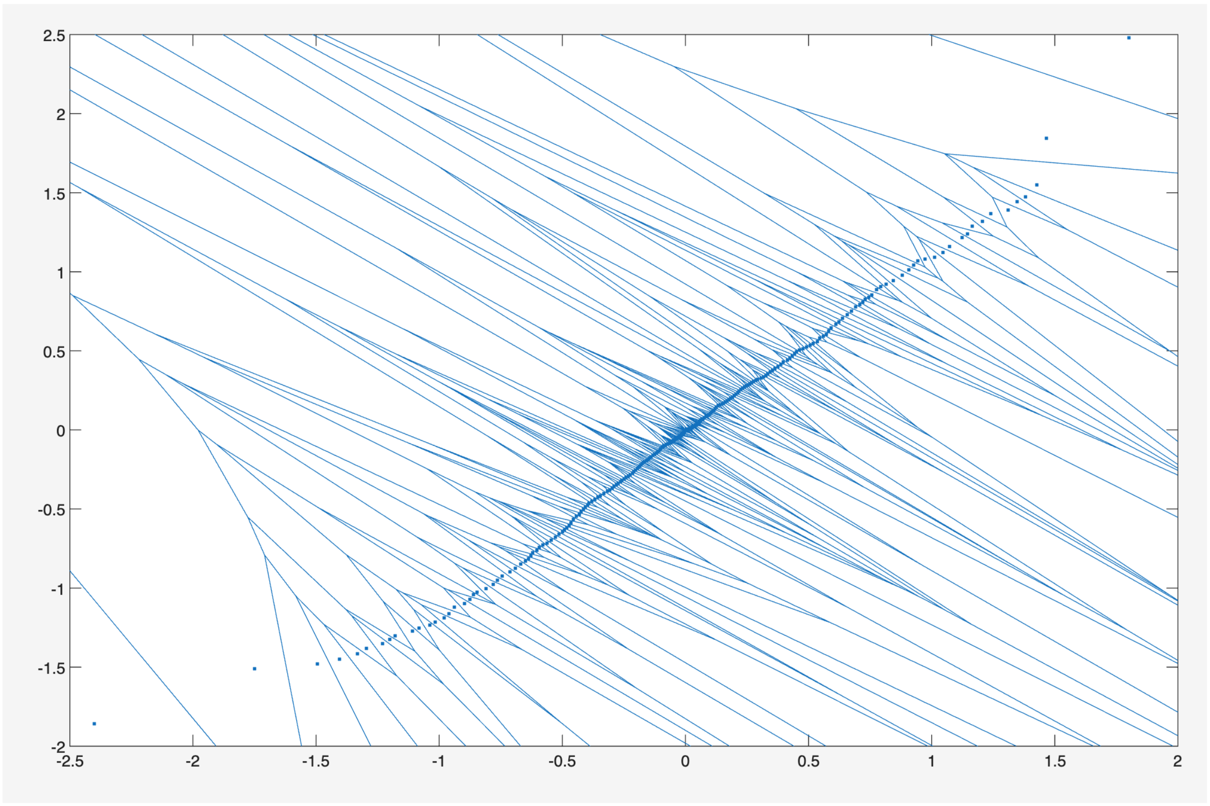
- output tracking (Kmeans kind of works the same as logarithmic)



- state signals (Kmeans kind of works the same as logarithmic)

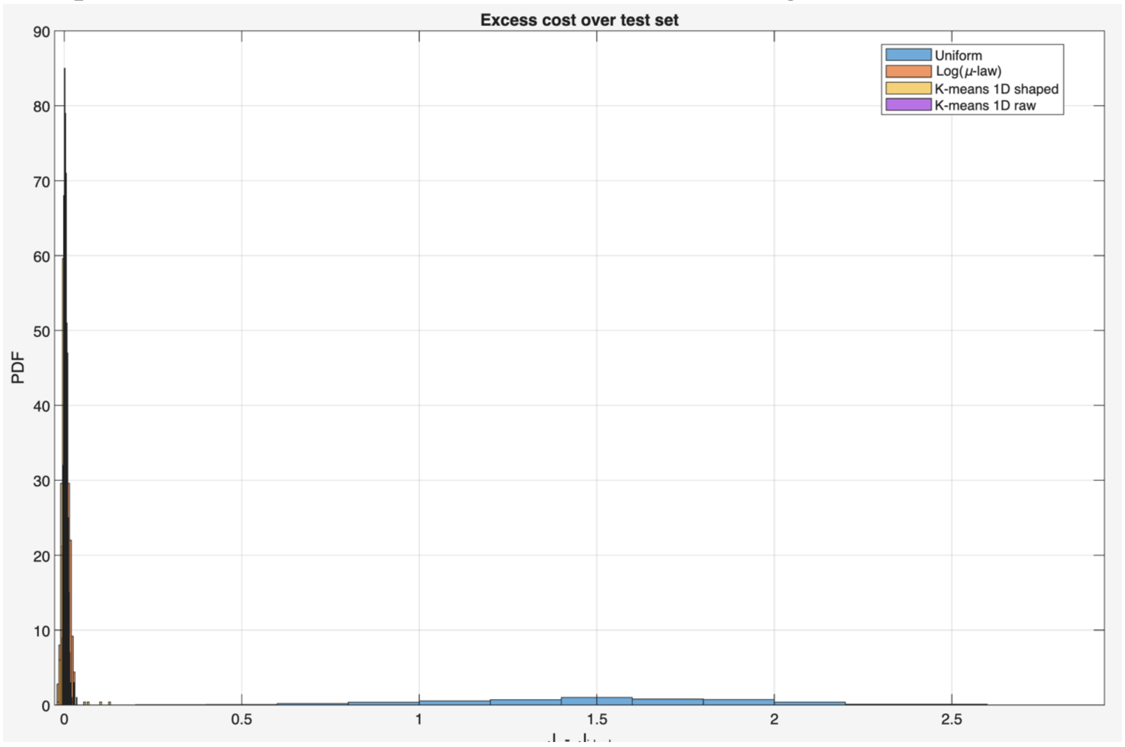


- Voronoi diagram for Kmeans centroids (it sure looks like the Kmeans learns the relevant partitioning of the 2D plane from the system trajectories)

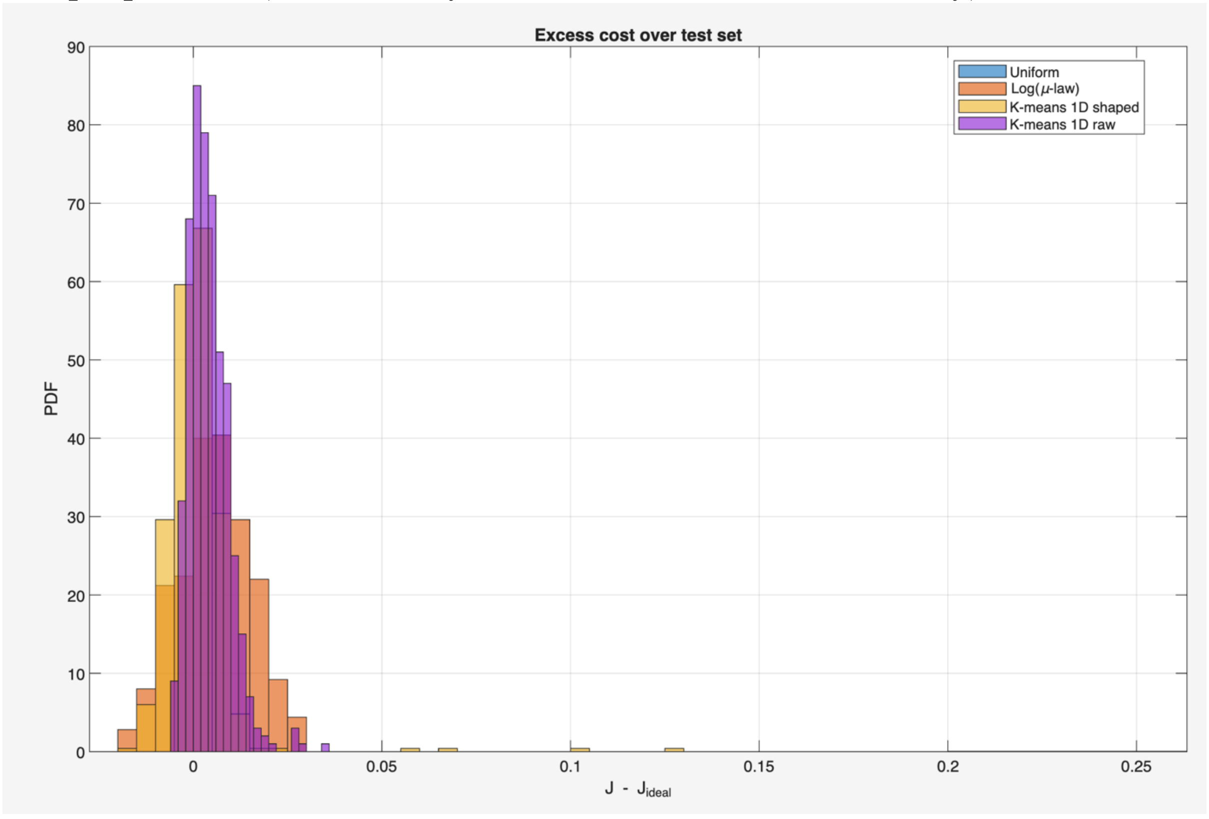


Excess cost of the LQR index for the stabilization to origin problem: J = J_ideal + Delta_J -> Plot of Delta_J distribution of probability using the considered quantizers (MATLAB file experiment_v1_kmeans_1D.m):

- First plot shows that the uniform is on a different league



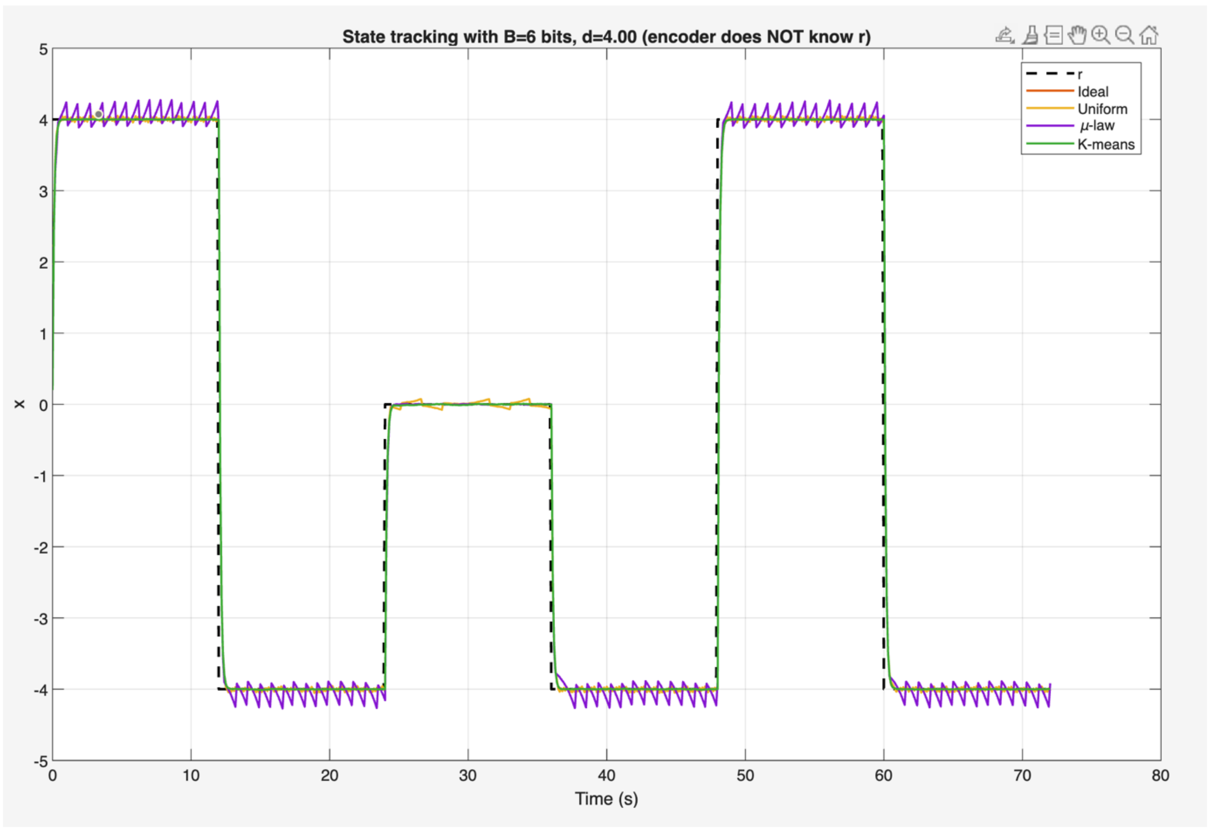
- Second plot shows that they all tend to converge to similar behaviors for this simple problem (Kmeans may be a bit more consistent, let's say):



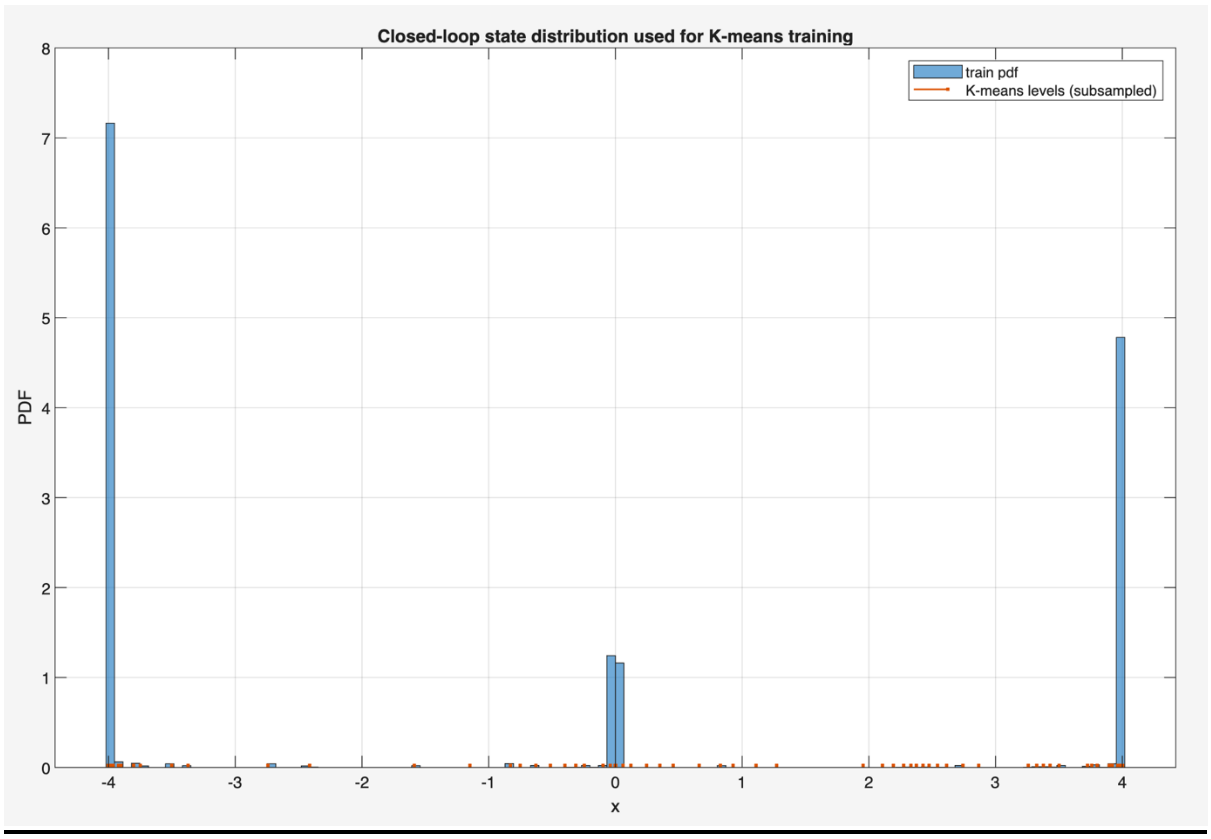
4) Behaviour and excess cost of the LQR index for reference tracking (consider three references: -d, 0, +d, to cover the state space as much as possible). Important conclusion: the logarithmic quantizer works “perfectly” for the origin, but does not work at all to the other extremes, as it doesn’t have quantization steps (precision) for x1 and x2 far from the origin. On the other hand, the Kmeans learns to have good precision around all three references.

- MATLAB file exp_v7_scalar_case_kmeans.m:

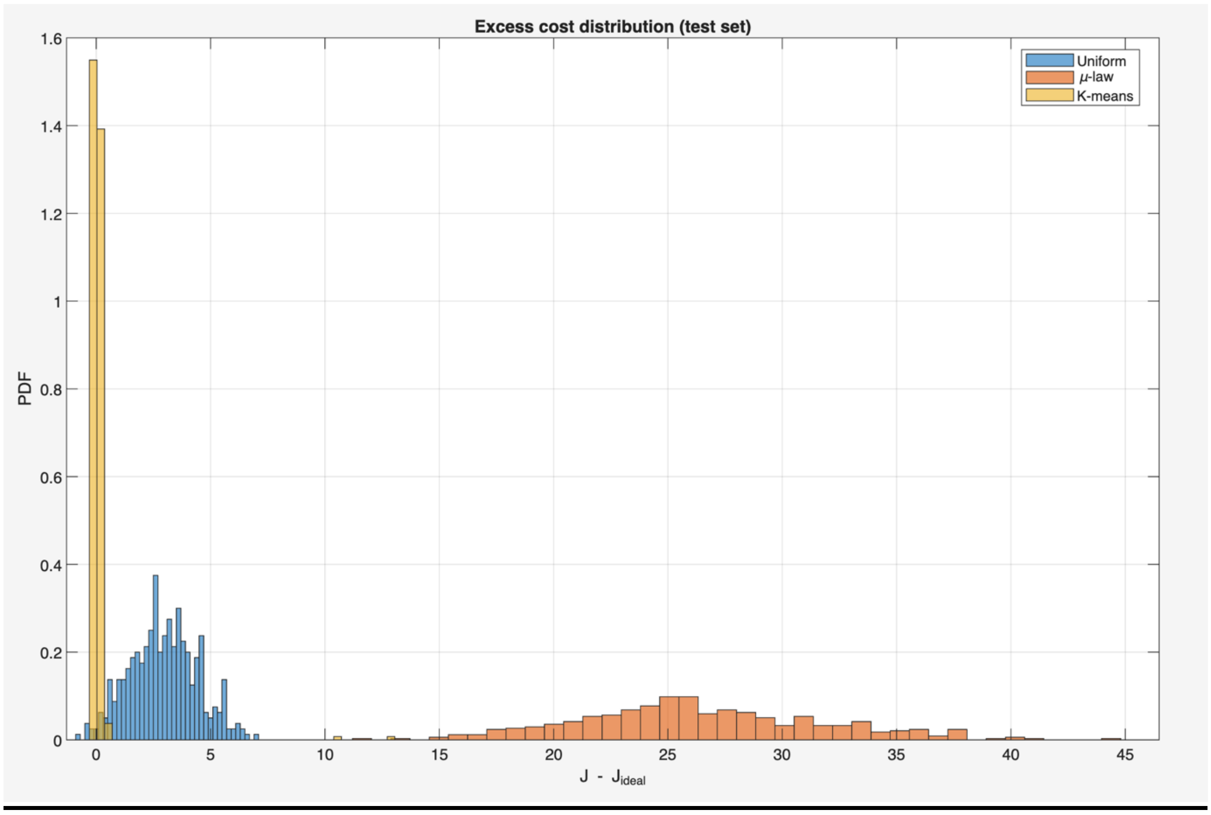
-- good tracking on all three references for Kmeans vs good tracking only around origin for the log quantizer (+same performance everywhere using uniform quantizer)



-- distribution of state signals for the three reference points



- vast improvement of the performance index J when we take multiple references into account



<u>What to exploit next?</u>

I) Quantization for problems where the control trajectories are predefined and improve those cases as much as possible; for example, finite horizon LQR, optimal control problems etc., in which we build a quantizer tailored to that problem => Data-driven quantization

II) I also tried a nonlinear parameter adaptation algorithm (PAA) type solution, as in System Identification, in which I wanted to “learn” the compensation between the ideal command signal and command signal computed using the quantized state signals, i.e., to compensate the quantizer loss of resolution, but, at first glance, it didn’t work too well. It could be experimented with more trials and different setups, but I don’t know if we could provide some guarantees of it, or a more rigorous methodology.

III) ??