Finite Blocklength Entropy-Achieving Coding for Linear System Stabilization

Yirui Cong, *Member, IEEE,* Xiangyun Zhou, *Senior Member, IEEE,* and Rodney A. Kennedy, *Fellow, IEEE*

***Abstract*—In this paper, we consider the minimum data rate problem for linear system stabilization under noiseless communication channels. Previous results indicated that having a data rate very approaching the entropy bound leads to large delays and data buffer sizes. In analogy, the entropy bound in Shannon's source coding theorem in traditional information theory displays this behavior, where the data rate can be arbitrarily close to the entropy bound but only at the cost of boundlessly enlarging the blocklength. However, in this work, we show the analogy is not strict. We prove that it is possible to stabilize a linear system at a rate equal to the entropy bound with zero delay, that is, where each system state is encoded and decoded within one time unit. We establish a set of sufficient conditions for guaranteeing zero-delay entropy-achieving codes. Following this we design an entropy-achieving code with finite blocklength satisfying the set of sufficient conditions, where the codeword length is uniformly bounded.**

***Index Terms*—Linear system control, stabilization, minimum data rate, entropy-achieving code, finite blocklength, zero delay.**

I. Introduction

A. Motivation

The minimum data rate problems for stabilizing Networked Control Systems (NCS) have been studied nearly 20 years with the earliest results appearing in [1] and [2]. By achieving the minimum data rate (i.e., the entropy bound), the NCS is able to most efficiently utilize the limited communication bandwidth. However, there is a drawback in that for existing entropy-bound-achieving methods the blocklength[1] goes to infinity when the data rate approaches the entropy bound (see [3], [4] for examples). As a result, if a data rate gets closer to the minimum data rate, each codeword needs the information from more time units to be encoded, and the NCS will face two serious problems: (i) the system delay is unacceptably large, and (ii) the data buffer for codeword storage is also unacceptably large. Unfortunately, it is impossible to reduce the blocklength while approaching the entropy bound using the methods in the literature, even for the simplest scalar (one-dimensional) linear systems. Even though some efforts were made for reducing the cost caused by finite blocklength coding (e.g., [5]), it is still not able to achieve the entropy bound in general. For this reason, the minimum data rate problems have been studied incompletely for stabilizing NCSs. This incompleteness comes from the fact that the encoding and decoding methods used are informed from Shannon's source coding theorem (see Chapter 5 in [6]), where the encoder encodes every k system states x_t (a.k.a. source symbols) to a packet as follows[2]

$$\underbrace{x_1 x_2 \cdots x_k}_{k}\underbrace{x_{k+1} x_{k+2} \cdots x_{2k}}_{k}\cdots \tag{1}$$

In Shannon's source coding theorem, it is well known that the larger the blocklength k is, even though the closer the data rate will get to the entropy bound, the longer the codeword will be. This fact implies that if we still follow the framework of Shannon's source coding theorem, any attempt to reduce the blocklength will eventually become a compromise so that the data rate is pushed away from the entropy bound. Therefore, it is necessary to jump out of the conventional framework and find a completely different way to study the minimum data rate for stabilizing NCSs.

In this work, we examine the linear system stabilization problem where the entropy bound can be achieved (without any gap) by using finite blocklength codes. Actually, if abandoning the view "encoding more system states is the only way to reduce the data rate", we can achieve this goal. Surprisingly, encoding only one system state at each time unit, i.e., the blocklength is $k = 1$ in (1), is enough to achieve this goal, which means the system delay can be zero for entropy-achieving codes.

B. Related Work

Generally speaking, in source coding design[3] for stabilization, the data rate reflects the ability to eliminate the system uncertainties: if the uncertainties gradually vanish under a sufficiently high data rate while the control law asymptotically pushes the system towards a fixed point (or a given region), then the system will be stabilized. For all those data rates that can guarantee system stabilization, there is a tight lower bound called the entropy bound, and minimum data rate problems seek this entropy bound and/or propose the coding method to achieve this bound.

Y. Cong is with the College of Intelligence Science and Technology, National University of Defense Technology, China.

X. Zhou and R. A. Kennedy are with the Research School of Electrical, Energy and Materials Engineering, the Australian National University, Australia.

The work of X. Zhou was supported by the Australian Research Council's Discovery Projects under Grant DP180104062. This work was also partially supported by the National Natural Science Foundation of China under Grants 61801494 and 61876187.

[1]In this paper, the blocklength is the number of system states in adjacent time units to encode a codeword. The length of one time unit refers to the sample period of a NCS.

[2]Since the existing works are based on Markov systems, encoding the last symbol (e.g., x_k, which contains the uncertainty from x_1 to x_k) is equivalent to encoding all the symbols in one block.

[3]The channel is assumed to be noiseless.

For discrete-time MIMO linear control systems with/without bounded disturbances, the minimum data rate for stabilization was obtained in [7], where the asymptotic observability was utilized to calculate a lower bound of minimum data rate which was then proved to be tight. In [8], an identical lower bound was derived for stabilizing linear stochastic systems in the mean square sense, and any data rate strictly greater than this lower bound can be achieved by using finite-dimensional coder-controller with periodic alphabet. Surprisingly, this lower bound (given in [7], [8]) has the same form as the topological entropy for linear mappings first introduced in 1965 by [9], and the topological entropy measures the average number of distinguishable orbits over time for a control-free dynamical system [10], [11]. Actually, the minimum data rate for stabilization equaling the topological entropy is a coincidence, since the former corresponds to the orbits related to stabilization (i.e., towards a fixed point or region) rather than all the orbits generated by an uncontrolled system as the latter corresponds to. In other words, the minimum data rate for stabilization reflects the local property of a mapping, while the topological entropy represents the global property. The reason why this coincidence happens for linear control systems is that: the uncertainties are always generated with the same speed regardless of where the uncertainty region is, because local property means global property in linear mappings. But for nonlinear systems, it is not the case. To describe the minimum data rate for stabilizing nonlinear control system, we need another kind of topological entropy, and this entropy was established in [12], named Topological Feedback Entropy (TFE). In [12], the TFE provided a strict way to describe the minimum data rate for stabilizing discrete-time linear/nonlinear control systems, and as an example the TFE for the locally uniformly asymptotically stabilization was calculated. Afterwards, a topological entropy called the invariance entropy [13], [14] was defined to determine the minimum data rate for stabilizing continuous-time control systems. Different from the discrete-time control systems, the control law is designed as a continuous-time function (rather than a constant value during a sampling interval), and it is immediately updated when receiving a new codeword [15], just like the event-triggered control strategy. The invariance entropy also works on discrete-time control systems, and the comparison with the TFE was fully discussed in [16].

All above studies are about the minimum data rate for all possible classes of quantizers. But if one only considers the logarithmic quantizers (a very important class of quantizers [17]), what should the minimum data rate for stabilization be? In [18], this problem was completely solved for discrete-time linear control systems.

Thanks to these seminal studies, the foundation of the minimum data rate stabilization theory was well established. However, as stated in Section I-A, the coding methods they used are all with the form of (1). That means, practically speaking, when approaching the entropy bound, the data rate results in an unacceptable system delay and an unacceptably large data buffer size for real-time control systems. Mathematically speaking, all the existing results just consider the case that $R > H$, where R is the data rate of a designed coding method and H is the entropy bound, and these works are not able to study the case $R = H$.

C. Our Contributions

In this work, we study the source coding theory for stabilizing discrete-time MIMO linear control systems, and propose a finite blocklength entropy-bound-achieving code for stabilization. The main contributions are: We establish a zero-delay framework of encoding-decoding designs where each system state is encoded and decoded within one time unit, i.e., the blocklength is 1. More importantly, in this framework we propose a set of sufficient conditions for entropy-achieving codes that once an encoding-decoding design satisfies this condition, then its data rate achieves the entropy bound without any gap. Based on this set of sufficient conditions, an entropy-achieving code with zero delay is designed, where the lengths of all the codewords are uniformly bounded.

D. Paper Organization

Section II gives the preliminaries of this paper, and it has four subsections: Firstly, the system model is given in Section II-A. Secondly, in Section II-B a coding-quantizing framework is established. Thirdly, a zero-delay encoding-decoding framework is proposed in Section II-C. Finally, in Section II-D the problem description is provided. In Section III, we propose a set of sufficient conditions for zero-delay entropy-achieving codes as a criterion for coding designs. Based on this criterion, an entropy-achieving code with bounded codeword length is designed in Section IV. Concluding remarks are given in Section V.

E. Notation

Throughout this paper, $\mathbb{R}$, $\overline{\mathbb{Z}}_+$, and $\mathbb{Z}_+$ denote the sets of real numbers, non-negative integers, and positive integers, respectively. $\mathbb{R}^n$ stands for the n-dimensional Euclidean space. $\mu_n(\mathcal{A})$ is the Lebesgue measure on a measurable set $\mathcal{A} \subseteq \mathbb{R}^n$. $D(\mathcal{B})$ denotes the diameter of a compact set $\mathcal{B} \subseteq \mathbb{R}^n$ with $D(\mathcal{B}) = \max_{x,y\in\mathcal{B}} \|x - y\|$. Symbol 0 represents the number zero or zero matrices (including vectors) with proper dimensions. The limit superior and limit inferior are denoted by $\overline{\lim}$ and $\underline{\lim}$, respectively. For a square matrix $A \in \mathbb{R}^{n\times n}$, the spectrum is $\mathrm{spec}(A)$, and the spectral radius is $\rho(A)$. $|\cdot|$ denotes the magnitude of a complex number. If not specified, $\|\cdot\|$ refers to the Euclidean norm. For $a \in \mathbb{R}$, $(a)^+$ returns $\max\{0, a\}$.

II. System Model and Problem Description

A. System Model

Consider the fully observed, time-invariant, linear system Σ:

$$x_{t+1} = Ax_t + Bu_t, \quad t \in \overline{\mathbb{Z}}_+, \tag{2}$$

where the state is $x_t \in \mathcal{X} = \mathbb{R}^n$, and the input is $u_t \in \mathcal{U} = \mathbb{R}^m$. The system matrix and input matrix are $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$, respectively. Without loss of generality, we assume the magnitudes of all eigenvalues in A are not less than 1, and pair (A, B) is controllable [4], [7].[4] The feedback control system is shown as Fig. 1, where the initial state x_0 is deterministically unknown in a compact set $\overline{\mathcal{X}}_0 \subset \mathcal{X}$ with $\mu_n(\overline{\mathcal{X}}_0) > 0$. Initially, state x_0 is unknown to the encoder-decoder pair, but $\overline{\mathcal{X}}_0$ is known to the encoder-decoder pair.

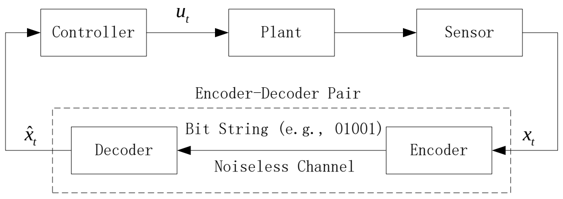


Fig. 1. Networked control system with a noiseless channel, where the dynamics of the plant are governed by (2).

At each time $t \in \overline{\mathbb{Z}}_+$, the encoder obtains the state x_t from the sensor and the decoder passes the quantized state $\widehat{x}_t$ to the controller. The controller applies the control signal $u_t = \psi_t(\widehat{x}_t)$ to the plant which affects the state x_{t+1} at time $t+1$ through (2), where $\psi_t : \mathcal{X} \to \mathcal{U}$ is known to the encoder-decoder pair. We can see that, without the encoder-decoder pair (in this case, $\widehat{x}_t = x_t$), the system in Fig. 1 becomes a classic discrete-time linear control system. With the encoder-decoder pair, the system in Fig. 1 is referred to as the quantized system.

At each time t, the encoder encodes the information from state x_t to a codeword, i.e., a bit string (see Fig. 1), and sends it to the decoder through a noiseless channel. After receiving this codeword, the decoder decodes it to get the quantized state $\widehat{x}_t$, which is used by the controller. This process (from $t = 0$ towards ∞) is called the encoding-decoding process with blocklength 1, because each codeword depends only on the system state at one time. Since the codeword is encoded and decoded within one time unit, the system delay is zero, i.e., any $\widehat{x}_t$ can be obtained by the controller at time t. In this paper, we focus on how to design the encoding-decoding process and the control law to stablize the quantized system of (2) with minimum data rate and blocklength 1 (i.e., zero delay).

B. Coding-Quantizing Framework

This framework tells how a point a in a compact set $\mathcal{A}$, called coding range, is encoded to a codeword, and how it is quantized. Briefly speaking, the main idea is to use a set of compact sets to cover $\mathcal{A}$. Then, each compact set is endowed with a unique bit string. For each compact set, we select one point as its representative, as the quantized value.

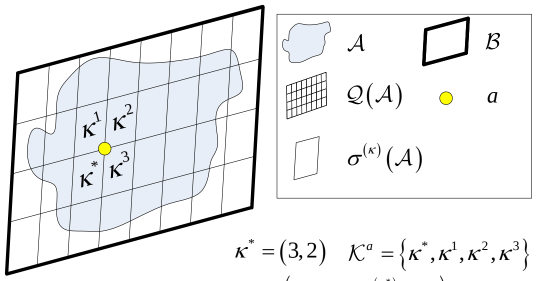
In general $\mathcal{A}$ may be hard to succinctly characterize so we introduce an easy-to-describe superset $\mathcal{B}$ and divide it into some regions (which may include parts not in $\mathcal{A}$). In fact these divisions are a cover, and Definition 1 furnishes the necessary notations and terminology, illustrated in Fig. 2.

Definition 1 (Coding Cover, Coding Cell, and Coding Scale)**.** *Given a coding range $\mathcal{A} \subset \mathbb{R}^n$, which is a compact set, the coding cover $\mathcal{Q}(\mathcal{A})$ is a finite set of compact sets $\sigma^{(\kappa)}(\mathcal{A})$ with the index set $\mathcal{K} \ni \kappa$ and the bijective $\kappa \mapsto \sigma^{(\kappa)}(\mathcal{A})$, when satisfying the following three conditions:*

(i) $\mu_n\left(\sigma^{(\kappa)}(\mathcal{A})\right) > 0$, $\forall \kappa \in \mathcal{K}$;
(ii) $\bigcup_{\kappa \in \mathcal{K}} \sigma^{(\kappa)}(\mathcal{A}) \supseteq \mathcal{A}$;
(iii) $\mu_n\left(\sigma^{(\kappa')}(\mathcal{A}) \bigcap \sigma^{(\kappa'')}(\mathcal{A})\right) = 0$, $\forall \kappa', \kappa'' \in \mathcal{K}$ $(\kappa' \neq \kappa'')$.

We call $\sigma^{(\kappa)}(\mathcal{A})$ as the coding cell *of $\mathcal{A}$ with index κ. We define $\overline{D} := \max_{\kappa \in \mathcal{K}} D\left(\sigma^{(\kappa)}(\mathcal{A})\right)$ as the* coding scale *of coding cover $\mathcal{Q}(\mathcal{A})$.*

[4]Since the magnitudes of all eigenvalues in A are not less than 1, pair (A, B) is controllable means it is stabilizable.


κ^1
κ^2
κ^*
κ^3
$\mathcal{A}$
$\mathcal{B}$
$\mathcal{Q}(\mathcal{A})$
a
$\sigma^{(\kappa)}(\mathcal{A})$
$\kappa^* = (3,2)$
$\mathcal{K}^a = \{\kappa^*, \kappa^1, \kappa^2, \kappa^3\}$

$$c(\mathcal{Q}(\mathcal{A}), a) = \overline{c}\left(\mathcal{Q}(\mathcal{A}), \sigma^{(\kappa^*)}(\mathcal{A})\right) = 01010$$

Fig. 2. Illustrations of coding cover and codeword. The shaded area is the coding range $\mathcal{A}$ and the area $\mathcal{B}$ enveloped by the thick lines is a superset of $\mathcal{A}$. The coding cover $\mathcal{Q}(\mathcal{A})$ (the grided parallelogram) is the set of all coding cells $\sigma^{(\kappa)}(\mathcal{A})$ (the parallelogram) whose joint (i.e., the area $\mathcal{B}$) contains $\mathcal{A}$. For point a, it falls into four coding cells $\sigma^{(\kappa)}(\mathcal{A})$ $(\kappa \in \mathcal{K}^a = \{\kappa^*, \kappa^1, \kappa^2, \kappa^3\})$, and the encoder-decoder pair has a consensus on choosing $\sigma^{(\kappa^*)}(\mathcal{A})$ as the coding cell that point a located in. Then, the codeword is chosen as the same as the cell code of $\sigma^{(\kappa^*)}(\mathcal{A})$. Since $\kappa^* = (3, 2)$, the codeword is 01010, where the first two bits 01 represent 2 (since it starts from 1 rather than 0), and for the same reason the last three bits 010 stand for 3.

In Definition 1, a coding cover needs to satisfy three conditions: (i) every coding cell has a positive volume in $\mathbb{R}^n$; (ii) any point in coding range $\mathcal{A}$ must fall in at least one coding cell; and (iii) any overlap between two coding cells is with zero volume. The coding scale reflects the worst-case diameter for the coding cells in the coding cover. An illustrative example of the coding cover is given in Fig. 2.

Note that the encoder and the decoder should have a consensus on the coding range $\mathcal{A}$ and the coding cover $\mathcal{Q}(\mathcal{A})$.

Definition 2 (Selected Coding Cell)**.** *Given a coding cover $\mathcal{Q}(\mathcal{A})$ and a point $a \in \mathcal{A}$, the selected coding cell of a is a mapping $\mathcal{S} : \mathcal{A} \to \mathcal{Q}(\mathcal{A})$ such that*

$$\mathcal{S}(a) = \begin{cases} \sigma^{(\kappa)}(\mathcal{A}), \text{if } a \in \sigma^{(\kappa)}(\mathcal{A}) \text{ holds for only one } \kappa \in \mathcal{K}, \\ \sigma^{(\kappa^*)}(\mathcal{A}), \text{if } a \in \sigma^{(\kappa)}(\mathcal{A}) \text{ holds for more than one } \kappa \in \mathcal{K}, \end{cases} \quad (3)$$

where the encoder and the decoder have a consensus on choosing κ^*.

The role of selected coding cell is indeed to classify all the points in set $\mathcal{A}$ into different cases (i.e., different coding cells). It builds up a link between any point $a \in \mathcal{A}$ and coding cells $\sigma^{(\kappa)}(\mathcal{A}) \in \mathcal{Q}(\mathcal{A})$. Based on this link, the encoder can classify a given point $a \in \mathcal{A}$ to the coding cell $\mathcal{S}(a)$ which corresponds to a unique bit string, called codeword (see Definition 3). Through decoding this codeword, the decoder knows which coding cell is selected by the decoder, i.e., the decoder can obtain $\mathcal{S}(a)$.

Based on the selected coding cell in Definition 2, we define the codebook and the codeword as follows.

Definition 3 (Codebook and Codeword)**.** *Given a coding cover $\mathcal{Q}(\mathcal{A})$ with $\#\mathcal{Q}(\mathcal{A}) > 1$ (or equivalently $\#\mathcal{K} > 1$), each $\sigma^{(\kappa)}(\mathcal{A})$ is endowed with a unique string of $L = \log_2\lceil \#\mathcal{Q}(\mathcal{A})\rceil$ bits, called the cell codeword*

$$\bar{c}\left(\mathcal{Q}(\mathcal{A}), \sigma^{(\kappa)}(\mathcal{A})\right) = b_1 b_2 \dots b_L, \tag{4}$$

where $b_l \in \{0,1\}$ ($l \in \{1,\dots,L\}$) is the l^{th} bit of the codeword, and L is the codeword length. All cell codewords form the codebook $\mathcal{C}(\mathcal{Q}(\mathcal{A}))$, i.e.,

$$\mathcal{C}(\mathcal{Q}(\mathcal{A})) = \left\{\bar{c}\big(\mathcal{Q}(\mathcal{A}), \sigma^{(\kappa)}(\mathcal{A})\big) \colon \kappa \in \mathcal{K}\right\}. \tag{5}$$

Then, the codeword of point $a \in \mathcal{A}$ is

$$c(\mathcal{Q}(\mathcal{A}), a) := \bar{c}\left(\mathcal{Q}(\mathcal{A}), \mathcal{S}(a)\right), \tag{6}$$

where $\mathcal{S}(a)$ is the selected coding cell (see Definition 2).

If $\#\mathcal{Q}(\mathcal{A}) = 1$ (or equivalently $\#\mathcal{K} = 1$), the codebook is $\mathcal{C}(\mathcal{Q}(\mathcal{A})) = \emptyset$, and there are no generated codewords of $\forall a \in \mathcal{A}$. In this case, the codeword length is $L = \log_2 \#\mathcal{Q}(\mathcal{A}) = 0$.

In Definition 3, different coding cells have different cell codewords, and all the cell codewords form the codebook. According to condition (ii) in Definition 1, any point a in coding range $\mathcal{A}$ must fall into at least one coding cell, and this guarantees that any $a \in \mathcal{A}$ must have a codeword. From (6), we know that this codeword is exactly the cell codeword of selected coding cell $\mathcal{S}(a)$.

Through decoding a codeword, the decoder can have a consensus on $\mathcal{S}(a)$ with the encoder, even though the decoder does not know point a. For control systems, furthermore, the quantized value of point a is needed by the controller (see Fig. 1), which is selected from $\mathcal{S}(a)$.

Before defining the quantized value, we introduce the definitions of the representative, the representative set, and the representative mapping.

Definition 4 (Representative, Representative Set, and Representative Mapping)**.** *Given a coding cover $\mathcal{Q}(\mathcal{A})$, point $\beta^{(\kappa)}$ is a representative of coding cell $\sigma^{(\kappa)}(\mathcal{A})$, if $\beta^{(\kappa)} \in \sigma^{(\kappa)}(\mathcal{A})$. A representative set is the set of the representatives of all coding cells in the coding cover, i.e.,*

$$\mathcal{B}(\mathcal{Q}(\mathcal{A})) = \left\{\beta^{(\kappa)} \colon \kappa \in \mathcal{K}\right\}. \tag{7}$$

Then, the representative mapping is $p\colon \mathcal{Q}(\mathcal{A}) \to \mathcal{B}(\mathcal{Q}(\mathcal{A}))$ such that

$$p(\sigma^{(\kappa)}(\mathcal{A})) = \beta^{(\kappa)}. \tag{8}$$

Definition 4 says that any point in a coding cell can be its representative. Once the representatives of all the coding cells are selected, these representatives form the representative set. Then, the representative mapping, which must be common to the encoder and decoder, matches each coding cell with its representative. We can now define the quantized value of any point $a \in \mathcal{A}$.

Definition 5 (Quantized Value)**.** *Given a coding cover $\mathcal{Q}(\mathcal{A})$ and a representative mapping p, the quantized value of any point $a \in \mathcal{A}$ is $\widehat{a} = p(\mathcal{S}(a))$.*

C. Zero-Delay Encoding-Decoding Framework

In this subsection, we establish the zero-delay encoding-decoding framework where each codeword is encoded and decoded in only one time unit. This contrasts with existing frameworks which use multiple time units to generate one codeword (see Section I-A).

Now, we start describing the encoding-decoding process from time $t = 0$. At time $t = 0$, the very parts to be designed are the coding cover $\mathcal{Q}_0(\overline{\mathcal{X}}_0)$, the codebook $\mathcal{C}_0(\mathcal{Q}_0(\overline{\mathcal{X}}_0))$, and the representative mapping $p_0\colon \mathcal{Q}_0(\overline{\mathcal{X}}_0) \to \mathcal{B}(\mathcal{Q}_0(\overline{\mathcal{X}}_0))$. The encoding-decoding process at time $t = 0$ has three design stages.

In stage 1, the coding range $\overline{\mathcal{X}}_0$ is determined for time $t = 0$: Since both the encoder and the decoder know $\overline{\mathcal{X}}_0$, the coding range is determined automatically.

In stage 2, the codeword is encoded and decoded: At time $t = 0$, the encoder obtains x_0 from the sensor (see Fig. 1). Through the selected coding cell (see Definition 2), the encoder knows which coding cell the state x_0 falls in. We label this coding cell as $\widehat{\mathcal{X}}_0$, and call it the selected coding cell at time $t = 0$, i.e., $\widehat{\mathcal{X}}_0 = \mathcal{S}_0(x_0)$. By Definition 3, the codeword $c(\mathcal{Q}_0(\overline{\mathcal{X}}_0), x_0)$ is then encoded via $c(\mathcal{Q}_0(\overline{\mathcal{X}}_0), x_0) = \bar{c}(\mathcal{Q}_0(\overline{\mathcal{X}}_0), \widehat{\mathcal{X}}_0)$ by the encoder. Afterwards, the encoder sends this codeword to the decoder. Through decoding this codeword, the decoder knows the selected coding cell $\widehat{\mathcal{X}}_0$, since one codeword uniquely corresponds to one coding cell.

In stage 3, the quantized state $\widehat{x}_0$ is determined: Since both the encoder and the decoder know $\widehat{\mathcal{X}}_0$, they calculate the same quantized state $\widehat{x}_0$ via $\widehat{x}_0 = p_0(\widehat{\mathcal{X}}_0)$, where p_0 is the representative mapping at $t = 0$. After deriving $\widehat{x}_0$, the decoder passes $\widehat{x}_0$ to the controller which generates the control signal $u_0 = \psi_0(\widehat{x}_0)$.

At time $t = 1$, the coding range $\overline{\mathcal{X}}_1$ is determined by both encoder and decoder in stage 1 through

$$\overline{\mathcal{X}}_1 = \{\bar{x}_1 \colon \bar{x}_1 = A\widetilde{x}_0 + B\psi_0(\widehat{x}_0), \quad \widetilde{x}_0 \in \widehat{\mathcal{X}}_0\}. \tag{9}$$

Then, the remaining stages for time $t = 1$ follow those at time $t = 0$, (with subscript 1).

Proceeding similarly, we can establish the whole encoding-decoding process for $t \in \mathbb{Z}_+$. Next, we give the rigorous description of the encoding-decoding process. The following definitions strictly describe the coding range and the selected coding cell at each time t.

Definition 6 (Coding Range at t)**.** *At time $t \in \mathbb{Z}_+$, the coding range is*

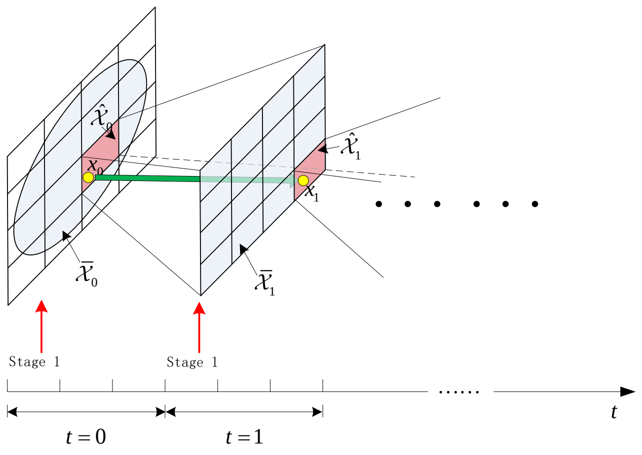
$$\overline{\mathcal{X}}_t = \{\bar{x}_t \colon \bar{x}_t = A\widetilde{x}_{t-1} + B\psi_{t-1}(\widehat{x}_{t-1}), \quad \widetilde{x}_{t-1} \in \widehat{\mathcal{X}}_{t-1}\}. \tag{10}$$

The coding range includes all possible states that can be reached by system (2) and $\widehat{\mathcal{X}}_{t-1}$. At time $t = 0$, we define $\overline{\mathcal{X}}_0$ (the range of initial state) as the coding range.

Definition 7 (Selected Coding Cell at t)**.** *At time $t \in \overline{\mathbb{Z}}_+$, the selected coding cell $\widehat{\mathcal{X}}_t$ is the selected coding cell (see Definition 2) of state x_t, i.e., $\widehat{\mathcal{X}}_t = \mathcal{S}_t(x_t)$.*

From Definition 6 and Definition 7, we can observe that the coding range and the selected coding cell are defined iteratively in time. For example, $\overline{\mathcal{X}}_t$ relies on $\widehat{\mathcal{X}}_{t-1}$, and conversely, $\widehat{\mathcal{X}}_t$ is dependent on the coding range $\overline{\mathcal{X}}_t$ (see Fig. 3).

Remark 1 (Distinguishing four notations of system state x_t, $\widetilde{x}_t$, $\overline{x}_t$ and $\widehat{x}_t$)**.** *The actual state of system (2) at time t is x_t, which is known to the encoder, but cannot be obtained by the decoder. Even though the decoder does not know x_t, it can know which coding cell x_t belongs to, and we use $\widetilde{x}_t$ to represent a point in this selected coding cell, i.e., in $\widehat{\mathcal{X}}_t$. For $\overline{x}_t$, it composes the coding range $\overline{\mathcal{X}}_t$. As for $\widehat{x}_t$, it is the quantized value of x_t.*



$\widehat{\mathcal{X}}_0$
x_0
$\overline{\mathcal{X}}_0$
$\widehat{\mathcal{X}}_1$
x_1
$\overline{\mathcal{X}}_1$
Stage 1
Stage 1
t
$t=0$
$t=1$

Fig. 3. Coding ranges and selected coding cells. The shaded ellipse in the left-hand side is $\overline{\mathcal{X}}_0$ which the initial state x_0 lies in. At time $t = 0$, in stage 1, the coding range $\overline{\mathcal{X}}_0$ is determined. At time $t = 0$, in stage 2, the selected coding cell $\widehat{\mathcal{X}}_0$ is determined by the state x_0 at the encoder and the decoder sides, respectively, where the coding range $\overline{\mathcal{X}}_0$ is 'divided' by the coding cover into 16 congruent parallelograms and $\widehat{\mathcal{X}}_0$ is one of these parallelograms. At time $t = 1$, the coding range is determined through (10) at the encoder and the decoder sides, respectively, and the selected coding cell is derived similar to that at $t = 0$. This process goes iteratively as time goes by.

Now, we strictly describe the encoding-decoding process at each $t \in \overline{\mathbb{Z}}_+$, for a given design of the coding cover $\mathcal{Q}_t(\overline{\mathcal{X}}_t)$, the codebook $\mathcal{C}_t(\mathcal{Q}_t(\overline{\mathcal{X}}_t))$, and the representative mapping $p_t \colon \mathcal{Q}_t(\overline{\mathcal{X}}_t) \to \mathcal{B}(\mathcal{Q}_t(\overline{\mathcal{X}}_t))$. This is summarized in Table I, which includes three stages:

Stage 1) **Determining coding range.** Both the encoder and the decoder determine the coding range $\overline{\mathcal{X}}_t$ through Definition 6.

Stage 2) **Encoding and decoding.** The encoder derives the selected coding cell $\widehat{\mathcal{X}}_t$ through Definition 7. Then, the encoder encodes $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t)$ via $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t) = \overline{c}(\mathcal{Q}_t(\overline{\mathcal{X}}_t), \widehat{\mathcal{X}}_t)$, and sends the codeword $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t)$ to the decoder. Through decoding this codeword, the decoder obtains the selected coding cell $\widehat{\mathcal{X}}_t$.

Stage 3) **Generating quantized state.** Both the encoder and the decoder use the representative mapping p_t to derive the quantized state $\widehat{x}_t$. The quantized state is used for the control signal $u_t = \psi_t(\widehat{x}_t)$ on the decoder side, and this signal is known to both encoder and decoder.

D. Data Rate and Problem Description

Conventionally, the data rate is defined in the asymptotically time-average sense (see also [4], [12])

$$R := \varliminf_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \log_2 M_t, \tag{11}$$

where M_t is the cardinality of the coding alphabet at time t, which means the noiseless channel transmits $\log_2 M_t$ bits in the t^{th} time unit.

In the zero-delay encoding-decoding framework, each codeword is decoded in one time unit, and thus $\log_2 M_t$ equals the codeword length $L_t = \lceil \log_2 \#\mathcal{Q}_t(\overline{\mathcal{X}}_t) \rceil$. For the rigorousness of our results, we specify the data rate definition in the zero-delay encoding-decoding framework as follows. Given a system Σ defined in (2), initial set $\overline{\mathcal{X}}_0$, coding-cover sequence $(\mathcal{Q}_t(\overline{\mathcal{X}}_t))_{t\in\overline{\mathbb{Z}}_+} =: \mathfrak{Q}$, codebook sequence $\mathfrak{C} =: (\mathcal{C}_t(\mathcal{Q}_t(\overline{\mathcal{X}}_t)))_{t\in\overline{\mathbb{Z}}_+}$, representative mapping sequence $\Pi = (p_t)_{t\in\overline{\mathbb{Z}}_+}$, and control law sequence $\Psi = (\psi_t)_{t\in\overline{\mathbb{Z}}_+}$, the data rate is expressed as

$$R(\Sigma, \overline{\mathcal{X}}_0, \mathfrak{Q}, \mathfrak{C}, \Pi, \Psi) := \varliminf_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} L_t. \tag{12}$$

Next, we give the definition of stability for quantized system.

Definition 8 (Stability)**.** *Given system Σ, initial set $\overline{\mathcal{X}}_0$, coding-cover sequence $\mathfrak{Q}$, codebook sequence $\mathfrak{C}$, representative mapping sequence Π, and control law sequence Ψ, if $\forall x_0 \in \overline{\mathcal{X}}_0$, the system state x_t satisfies $\lim_{t\to\infty} x_t = 0$, then the quantized system is stable; Otherwise, it is unstable.*

This paper focuses on how to design coding-cover sequence $\mathfrak{Q}$, codebook sequence $\mathfrak{C}$, representative mapping sequence Π, and control law sequence Ψ (i.e., the coding cover, codebook, representative mapping, and control law at each time $t \in \overline{\mathbb{Z}}_+$) to stabilize the quantized system of (2) with minimum data rate under our zero-delay encoding-decoding framework.

III. Entropy-Achieving Analysis

In this section, we propose a set of sufficient conditions for an entropy-achieving design and with finite codeword length. That means the rate will be equal to the entropy[5]

$$H(A) = \sum_{\lambda\in\mathrm{spec}(A)} \log_2 |\lambda|, \tag{13}$$

and the codeword length L_t ($t \in \overline{\mathbb{Z}}_+$) is bounded.

[5]This entropy is known as the topological feedback entropy [12], or the invariance entropy [13], [14], and it is numerically equal to the topological entropy [9] for linear mappings.

TABLE I
ENCODING-DECODING PROCESS IN EACH TIME UNIT

	Encoder	Decoder
Stage 1)	1. determine $\overline{\mathcal{X}}_t$ via Definition 6	1. determine $\overline{\mathcal{X}}_t$ via Definition 6
Stage 2)	1. determine $\widehat{\mathcal{X}}_t$ via x_t obtained from the sensor	
	2. encode $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t)$	
	3. send $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t)$	1. receive $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t)$
		2. decode $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t)$ to determine $\widehat{\mathcal{X}}_t$
Stage 3)	1. calculate $\widehat{x}_t = p_t(\widehat{\mathcal{X}}_t)$	1. calculate $\widehat{x}_t = p_t(\widehat{\mathcal{X}}_t)$
		2. pass $\widehat{x}_t$ to the controller

Firstly, we give an important lemma to lower bound all data rates for linear system stabilizations. This lower bound is exactly the entropy $H(A)$.

Lemma 1 (Lower Bound). *For any linear system Σ, and any initial set $\overline{\mathcal{X}}_0 \subset \mathcal{X}$, the data rate for stabilization is lower bounded by*

$$R(\Sigma, \overline{\mathcal{X}}_0, \mathfrak{Q}^{\mathrm{s}}, \mathfrak{C}^{\mathrm{s}}, \Pi^{\mathrm{s}}, \Psi^{\mathrm{s}}) \geq H(A), \tag{14}$$

where the superscript s *in $\mathfrak{Q}^{\mathrm{s}}$, $\mathfrak{C}^{\mathrm{s}}$, Π^{s}, and Ψ^{s} means the coding-cover sequence, the codebook sequence, the representative mapping sequence, and the control law sequence to stabilize system* (2).

Proof: The proof follows a standard volume argument, which can be found in e.g., [7] and [4]. ■

Remark 2. *The entropy $H(A)$ represents the rate of generated uncertainties by a given system Σ; while the data rate $R(\Sigma, \overline{\mathcal{X}}_0, \mathfrak{Q}^{\mathrm{s}}, \mathfrak{C}^{\mathrm{s}}, \Pi^{\mathrm{s}}, \Psi^{\mathrm{s}})$ reflects the speed to eliminate those uncertainties. Lemma 1 tells that to stabilize a system, the speed of eliminating uncertainties cannot be slower than the speed of generating uncertainties.*

Lemma 1 gives the fundamental limit of data rate for stabilizations, which is similar to the Shannon entropy in source coding theory (see [6]). In the literature, this similarity inspired the researchers to use the same idea from source coding theory to derive entropy-achieving codes. To be more specific, all the existing results package multiple time units' system states into one codeword [see (1) in Section I-A] and approach the entropy bound through enlarging the blocklength. As a result, the existing methods can only achieve rates $R(\Sigma, \overline{\mathcal{X}}_0, \mathfrak{Q}^{\mathrm{s}}, \mathfrak{C}^{\mathrm{s}}, \Pi^{\mathrm{s}}, \Psi^{\mathrm{s}}) > H(A)$. When the data rate equals the entropy bound $H(A)$, the blocklength is infinite, which means the codeword length and the system delay are also infinite.

Indeed, it is possible to achieve the entropy bound with finite blocklength, even in the zero-delay framework proposed in Section II-C. We give a sufficient set of conditions for entropy-achieving codes in Theorem 1. Before that, we give a special class of coding covers as follows.

Definition 9 (Perfect Coding Cover). *A coding cover $\mathcal{Q}(\mathcal{A})$ is perfect if the following two conditions hold:*

(i) $\forall \kappa', \kappa'' \in \mathcal{K}_t,\ \mu_n\left(\sigma^{(\kappa')}(\mathcal{A})\right) = \mu_n\left(\sigma^{(\kappa'')}(\mathcal{A})\right)$;

(ii) $\bigcup_{\kappa \in \mathcal{K}} \sigma^{(\kappa)}(\mathcal{A}) = \mathcal{A}$.

Remark 3. *In Definition 9, condition (i) means all the coding cells have the same Lebesgue measure. Condition (ii) implies a perfect coding cover partitions*[6] *the set $\mathcal{A}$.*

Theorem 1 (Entropy-Achieving Criterion). *A quantized system is stabilized with data rate*

$$R(\Sigma, \overline{\mathcal{X}}_0, \mathfrak{Q}^{\mathrm{s}}, \mathfrak{C}^{\mathrm{s}}, \Pi^{\mathrm{s}}, \Psi^{\mathrm{s}}) = H(A), \tag{15}$$

and uniformly bounded codeword length, if the following five conditions are satisfied in system designs:

(i) $\psi_t(\widehat{x}_t) = -K\widehat{x}_t$ *such that* $\rho(A - BK) < 1$;

(ii) $\lim_{t\to\infty} \overline{D}_t = 0$ *[$\overline{D}_t$ is the coding scale of coding cover $\mathcal{Q}_t(\overline{\mathcal{X}}_t)$];*

(iii) $\exists \underline{t} \in \mathbb{Z}_+$, s.t., $\mathcal{Q}_t(\overline{\mathcal{X}}_t)$ *is perfect for all* $t \geq \underline{t}$;

(iv) $\exists L$: $\forall t \in \mathbb{Z}_+$, s.t., $\log_2 \#\mathcal{Q}_t(\overline{\mathcal{X}}_t) \in \mathbb{Z}_+ \bigcap [0,\ L]$;

(v) $\lim_{t\to\infty} \frac{1}{t} \log_2 \mu_n(\sigma_{t-1}^{(\kappa_{t-1})}(\overline{\mathcal{X}}_{t-1})) = 0$.

Proof: See Appendix A. ■

Remark 4 (Illustrations of the Five Conditions). *In Theorem 1, condition (i) means that the control law to stabilize a quantized system is just as to stabilize a linear system using linear feedback. Condition (ii) requires the coding scale of $\mathcal{Q}_t(\overline{\mathcal{X}}_t)$ converges to 0 as t goes to infinity, which means the decoder asymptotically observes the system state x_t. If conditions (i) and (ii) are satisfied, then the quantized system is stabilized (see the proof of Theorem 1).*

Condition (iii) means that once t is large enough, the coding cover must be perfect, which means the coding range $\overline{\mathcal{X}}_t$ is equally partitioned. As for condition (iv), the numbers of designed coding cells should be integer powers of 2 and bounded for all $t \in \mathbb{Z}_+$. The most important condition is (v), and it is related to the convergence rate/speed of coding scale $\overline{D}_t$. To be more specific, $\overline{D}_t$ cannot converge exponentially fast (i.e., $\lim_{t\to\infty} \frac{1}{t}\log_2 \overline{D}_t < 0$ cannot hold), otherwise condition (v) will be violated:

$$\lim_{t\to\infty} \frac{\log_2 \mu_n(\sigma_{t-1}^{(\kappa_{t-1})}(\overline{\mathcal{X}}_{t-1}))}{t} \leq \lim_{t\to\infty} \frac{\log_2 \varpi_n \overline{D}_t^n}{t} < 0, \tag{16}$$

where ϖ_n is the volume of the n-dimensional ball of radius 1.[7] *If conditions (iii)–(v) are satisfied, then the data rate is $H(A)$ [i.e., (15) holds], and the codeword length is bounded.*

[6]Rigourously speaking, a perfect coding cover is not a partition of $\mathcal{A}$, since the intersection of two coding cells can be non-empty, even though the intersection are with measure zero [see condition 3) in Definition 1].

[7]We note that $\varpi_n = 2\pi^{\frac{n}{2}}/\Gamma(\frac{n}{2})$, where $\Gamma(\cdot)$ is the Gamma function.

Note that any system design satisfying the five conditions in Theorem 1 will result in a finite blocklength entropy-achieving code whose blocklength is 1, i.e., with zero delay. In Section IV, we design a zero-delay entropy-achieving code by using Theorem 1.

Before giving this design in Section IV, we examine the convergence rate of x_t in Theorem 1. Through condition (v) in Theorem 1 and the corresponding discussions in Remark 4, one can observe that x_t cannot converge exponentially fast, since the worst case $\|x_t\|$ cannot be smaller than half of the coding scale $\overline{D}_t$ (see Appendix B for a detailed proof). Actually, when the data rate is equal to the entropy, no entropy-achieving code can achieve exponential convergence. This result is given in Proposition 1, which can be regarded as the converse of Theorem 1 w.r.t. condition (v).

Proposition 1 (Rate Gap of Exponential Convergence)**.** *If x_t converges exponentially fast, i.e., $\exists \alpha \geq \|x_0\|, \beta > 1$ such that*

$$\|x_t\| \leq \alpha\beta^{-t}, \quad t \in \overline{\mathbb{Z}}_+, \tag{17}$$

then the data rate R defined by (11) satisfies

$$R \geq H(A) + n\log_2\beta, \tag{18}$$

where $n\log_2\beta$ is the rate gap of exponential convergence.

Proof: See Appendix C. ■

Remark 5. *The result in Proposition 1 holds for all possible encoding-decoding processes, not only for the cases in the zero-delay encoding-decoding framework. This result tells us that there is a non-zero rate gap if one wants x_t to converge exponentially fast, i.e., $\beta > 1$. A faster convergence rate, i.e., a larger β, requires a larger rate gap. In other words, it is impossible to have exponential convergence while having zero rate gap to the entropy. Our result in Theorem 1 makes $\beta = 1$ as guaranteed by condition (v).*

IV. Entropy-Achieving Synthesis

In this section, we propose the method to stabilize the quantized system of (2) with data rate $H(A)$ in zero-delay encoding-decoding framework. From Section IV-A to Section IV-D, we give the details about how the coding cover, the codebook, and the representative mapping are designed at each $t \in \overline{\mathbb{Z}}_+$ in our method. Based on above designs, we design the detailed encoding-decoding process at the encoder and decoder sides in Section IV-E.

Note that there are two novel aspects in our design method:

- We directly design the coding cover in the original (state) space by introducing the parallelepiped-shape coding cells (see Section IV-A). Thus, there is no need to make any transformation between the original space and the transformed space in each time unit.
- We introduce the edge-length constraint which can accurately control the convergence process of coding cells w.r.t. time (see Section IV-E). Therefore, condition (v) in Theorem 1 can be satisfied. In contrast, the existing methods in the literature, e.g., [3], [4], [7], cannot satisfy condition (v) in Theorem 1.

To facilitate the readers to understand the proposed method, we would like to mention that: (i) A high-level description of the proposed encoding and decoding designs is given in Table I. The encoder design has three stages, namely, determining coding range, encoding, and generating quantized state. The decoder design also has three corresponding stages, namely, determining coding range, decoding, and generating quantized state. (ii) The detailed designs of encoder and decoder are described in Algorithm 1 and Algorithm 2, respectively, in Section IV-E. Detailed explanations are also given for each stage in the encoder/decoder design after each algorithm.

A. Coding Cover Design

Most of previous studies design hyper-rectangular coding cells in a transformed space related to the real Jordan form, and then transform these coding cells back to the original space (e.g., [4], [7], [12]). We call these indirect designs, since this forward and back transformations have to be done in every time unit. In contrast, our coding cell design is directly made in the original space, i.e., no transformations related to the real Jordan form are required in every time unit.

Now, we give a description of a n-dimensional parallelepiped generated by $n+1$ points.

Definition 10 (Parallelepiped)**.** $\forall v^{(0)}, \ldots, v^{(n)} \in \mathbb{R}^n$ *such that* $v^{(1)} - v^{(0)}, \ldots, v^{(n)} - v^{(0)}$ *are linearly independent, then the parallelepiped generated by these $n+1$ points is*

$$\mathcal{P}(v^{(0)}, \ldots, v^{(n)}) = \Big\{ \sum_{i=1}^{n} \theta_i (v^{(i)} - v^{(0)}) + v^{(0)} \colon \theta_1, \ldots, \theta_n \in [0,1] \Big\}. \tag{19}$$

We call $v^{(0)}, \ldots, v^{(n)}$ the generating points of the parallelepiped in (19)*.*

In contrast to the traditional definition of the parallelepiped, which depends on n linearly independent vectors (see [19]), Definition 10 requires $n+1$ generating points so that the coding cell design can be simplified.

Lemma 2 describes the form of transformed parallelepiped by invertible affine mappings.

Lemma 2 (Parallelepiped Under Invertible Affine Transformation)**.** *Let $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ be any invertible affine mapping, then the transformed parallelepiped is $\varphi(\mathcal{P}(v^{(0)}, \ldots, v^{(n)})) = \mathcal{P}(\varphi(v^{(0)}), \ldots, \varphi(v^{(n)}))$.*

Proof: See Appendix D. ■

Remark 6 (Two Facts of Transformed Parallelepiped)**.** *From Lemma 2, we can observe that any parallelepiped is still parallelepiped after applying any invertible affine mapping. Furthermore, the transformed parallelepiped has an easy-to-compute form w.r.t. Definition 10 by simply transforming each $v^{(i)}$ in $\mathcal{P}(v^{(0)}, \ldots, v^{(n)})$ to $\varphi(v^{(i)})$.*

Remark 7 (Preservation of Parallelepiped-Shape)**.** *Remark 6 implies that if the selected coding cell $\widehat{\mathcal{X}}_t$ at time t is a parallelepiped $\mathcal{P}(v^{(0)}, \ldots, v^{(n)})$, then the coding range $\overline{\mathcal{X}}_{t+1}$ at time $t+1$ is also a parallelepiped $\mathcal{P}(\varphi(v^{(0)}), \ldots, \varphi(v^{(n)}))$,* *where $\varphi(v^{(i)}) = Av^{(i)} + Bu_t$ for $i \in \{1,\ldots,n\}$. Therefore, it is important to discuss how to get the parallelepiped-shaped $\widehat{\mathcal{X}}_t$ for all $t \in \overline{\mathbb{Z}}_+$. This is given as follows: At time $t = 0$, even though $\overline{\mathcal{X}}_0$ is generally not a parallelepiped, we can find $n+1$ generating points $\overline{v}_0^{(0)}, \ldots, \overline{v}_0^{(n)}$ such that $\overline{\mathcal{X}}_0 \subseteq \mathcal{P}(\overline{v}_0^{(0)}, \ldots, \overline{v}_0^{(n)})$. To satisfy condition (iii) in Theorem 1, we partition $\mathcal{P}(\overline{v}_0^{(0)}, \ldots, \overline{v}_0^{(n)})$ into congruent parallelepiped-shaped coding cells with edge length $d_0^{(i)}$, where $\|\overline{v}_0^{(i)} - \overline{v}_0^{(0)}\|/d_0^{(i)} := \overline{\kappa}_0^{(i)} \in \mathbb{Z}_+$, for $i \in \{1,\ldots,n\}$. The symbol $\overline{\kappa}_0^{(i)}$ is the maximum number of partitioning sets w.r.t. the axis $\overline{v}_0^{(i)} - \overline{v}_0^{(0)}$, and is always a power of 2. The index set of the coding cells has the following form*

$$\mathcal{K}_0 = \Big\{\kappa_0 = \big(\kappa_0^{(0)}, \ldots, \kappa_0^{(n)}\big) \colon \kappa_0^{(i)} \in \{1, \ldots, \overline{\kappa}_0^{(i)}\}, \quad i \in \{1,\ldots,n\}\Big\}. \tag{20}$$

Now, each coding cell $\sigma_0^{(\kappa_0)}(\overline{\mathcal{X}}_0)$ can be uniquely determined by index $\kappa_0 \in \mathcal{K}_0$, and the coding cover can be expressed as $\mathcal{Q}_0(\overline{\mathcal{X}}_0) = \{\sigma_0^{(\kappa_0)}(\overline{\mathcal{X}}_0) \colon \kappa_0 \in \mathcal{K}_0\}$. A 2-dimensional example for the partition is given in Fig. 4. Since each coding cell $\sigma_0^{(\kappa_0)}(\overline{\mathcal{X}}_0)$ ($\kappa_0 \in \mathcal{K}_0$) is a parallelepiped, the selected coding cell $\widehat{\mathcal{X}}_0$ is parallelepiped-shaped. At time $t = 1$, according to Lemma 2, we can derive $\overline{\mathcal{X}}_1 \subseteq \mathcal{P}(\overline{v}_1^{(0)}, \ldots, \overline{v}_1^{(n)})$, where $\overline{v}_1^{(i)} = A\widetilde{v}_0^{(i)} + Bu_0$ ($i \in \{1,\ldots,n\}$), and the other parts are the same as those at time $t = 0$.

Proceeding forward, we can readily obtain that $\widehat{\mathcal{X}}_t$ is parallelepiped-shaped for all $t \in \overline{\mathbb{Z}}_+$ and so obtain $\overline{\mathcal{X}}_t$ for all $t \in \mathbb{Z}_+$. Furthermore, $\forall t \in \mathbb{Z}_+$, we have

$$\bigcup_{\kappa_t \in \mathcal{K}_t} \sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t) = \overline{\mathcal{X}}_t, \tag{21}$$

which together with the congruent coding cells implies condition (iii) in Theorem 1 holds with $\underline{t} = 1$.

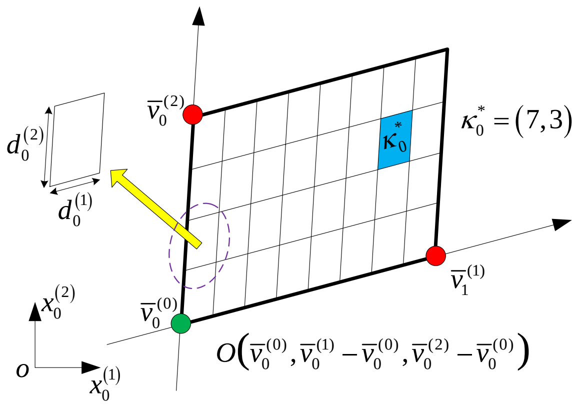


Fig. 4. Parallelepiped coding cells in 2 dimensions. The parallelepiped $\mathcal{P}(\overline{v}_0^{(0)}, \overline{v}_0^{(1)}, \overline{v}_0^{(2)})$ (see the parallelogram enclosed by thick lines) is partitioned into 8×4 parallelepiped-shaped coding cells with lengths $d_0^{(1)}$ and $d_0^{(2)}$ (see the parallelogram pointed to by large yellow arrow). Symbols o and $O(\overline{v}_0^{(0)}, \overline{v}_0^{(1)} - \overline{v}_0^{(0)}, \ldots, \overline{v}_0^{(n)} - \overline{v}_0^{(0)})$ represent the origins of the original coordinate system and the affine coordinate system, respectively. The blue (shaded) area in $\mathcal{P}(\overline{v}_0^{(0)}, \overline{v}_0^{(1)}, \overline{v}_0^{(2)})$ is $\widehat{\mathcal{X}}_0 = \sigma_0^{(\kappa_0^*)}(\overline{\mathcal{X}}_1)$, where the index κ_0^* is calculated by (29) and (30).

Remark 8 (Coding Cell Design)**.** *Based on $\overline{\mathcal{X}}_t = \mathcal{P}(\overline{v}_t^{(0)}, \ldots, \overline{v}_t^{(n)})$, the parallelepiped-shaped coding cell is expressed as*

$$\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t) = \mathcal{P}\big(\widetilde{v}_t^{(\kappa_t,0)}, \ldots, \widetilde{v}_t^{(\kappa_t,n)}\big), \tag{22}$$

for all $t \in \overline{\mathbb{Z}}_+$ and $\kappa_t \in \mathcal{K}_t$ [$\mathcal{K}_t$ is with a similar form to (20) for $\mathcal{K}_0$], where

$$\begin{cases} \widetilde{v}_t^{(\kappa_t,0)} = \overline{v}_t^{(0)} + \sum_{j=1}^{n} \kappa_t^{(j)} d_t^{(j)} \dfrac{\overline{v}_t^{(j)} - \overline{v}_t^{(0)}}{\|\overline{v}_t^{(j)} - \overline{v}_t^{(0)}\|}, & i = 0 \\ \widetilde{v}_t^{(\kappa_t,i)} = \widetilde{v}_t^{(\kappa_t,0)} + d_t^{(i)} \dfrac{\overline{v}_t^{(i)} - \overline{v}_t^{(0)}}{\|\overline{v}_t^{(i)} - \overline{v}_t^{(0)}\|}, & i \in \{1,\ldots,n\}. \end{cases} \tag{23}$$

Fig. 4 provides an illustrative case for $n = 2$ and $t = 1$.

Note that the coding cover $\mathcal{Q}_t(\overline{\mathcal{X}}_t)$ is uniquely determined when the coding cells are designed.

B. Codebook Design

This subsection gives the detailed design of the codebook, which assigns each coding cell (designed in Section IV-A) with a unique cell codeword.

If $\#\mathcal{K}_t > 1$, $\forall \kappa_t = \big(\kappa_t^{(0)}, \ldots, \kappa_t^{(n)}\big) \in \mathcal{K}_t$, the cell codeword of $\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)$ is

$$\bar{c}\big(\mathcal{Q}_t(\overline{\mathcal{X}}_t), \sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)\big) = \underbrace{b_1^{(\kappa_t)} \ldots b_{\iota_1}^{(\kappa_t)}}_{\log_2 \overline{\kappa}_t^{(1)}} \ldots \underbrace{b_{\iota_{n-1}+1}^{(\kappa_t)} \ldots b_{\iota_n}^{(\kappa_t)}}_{\log_2 \overline{\kappa}_t^{(n)}}, \tag{24}$$

where $\iota_i = \sum_{j=1}^{i} \log_2 \overline{\kappa}_t^{(j)} \in \overline{\mathbb{Z}}_+$ for $i \in \{1,\ldots,n\}$, specifically, $\iota_n = L_t$. Note that ι_i gives the last index corresponding to $\kappa_t^{(i)}$. Let $\iota_0 := 0$ and $\forall l \in \{\iota_{i-1}+1, \ldots, \iota_i\}$ ($i \in \{1,\ldots,n\}$) the bit b_l in (24) is

$$b_l = \left\lfloor \frac{(\kappa_t^{(i)} - 1) \bmod 2^{l-\iota_{i-1}}}{2^{l-\iota_{i-1}-1}} \right\rfloor. \tag{25}$$

It can be verified that, in this cell codeword design, every coding cell is endowed with a unique cell codeword. If $\#\mathcal{K}_t = 1$, then we set

$$\bar{c}\left(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t\right) = \text{null}, \tag{26}$$

which means the codeword length is $L_t = 0$.

Note that the codebook $\mathcal{C}_t(\mathcal{Q}_t(\overline{\mathcal{X}}_t))$ is uniquely determined when the cell codewords are designed.

C. Representative Mapping Design

We choose the centroid of a coding cell as its representative, and the representative set has the following form

$$\mathcal{B}_t(\mathcal{Q}_t(\overline{\mathcal{X}}_t)) = \Big\{b_t^{(\kappa_t)} \colon b_t^{(\kappa_t)} = \frac{1}{2}\sum_{i=1}^{n} \big(\widetilde{v}_t^{(\kappa_t,i)} - \widetilde{v}_t^{(\kappa_t,0)}\big) + \widetilde{v}_t^{(\kappa_t,0)}, \quad \kappa_t \in \mathcal{K}_t\Big\}. \tag{27}$$

Thus, the representative mapping is $p_t \colon \mathcal{Q}_t(\overline{\mathcal{X}}_t) \to \mathcal{B}_t(\mathcal{Q}_t(\overline{\mathcal{X}}_t))$ such that

$$p_t\big(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)\big) = \frac{1}{2}\sum_{i=1}^{n} \big(\widetilde{v}_t^{(\kappa_t,i)} - \widetilde{v}_t^{(\kappa_t,0)}\big) + \widetilde{v}_t^{(\kappa_t,0)}. \tag{28}$$

D. Control Law Design

We design the control law $u_t = -K\widehat{x}_t$ for the controller such that $\rho(A-BK) < 1$ holds. Note that the feedback matrix K is known to the encoder and the decoder. The control law design satisfies condition (i) in Theorem 1.

E. Encoding-Decoding Process

In this subsection, we describe the encoding and decoding processes such that the quantized system of (2) is stabilized with zero-delay entropy-bound-achieving code, based on the designs of coding cover (see Section IV-A), codebook (see Section IV-B), representative mapping (see Section IV-C), and control law (see Section IV-D).

Algorithm 1 gives the details about what the encoder does at each time t. Before starting Algorithm 1, two more things should be stressed: Firstly, this algorithm needs a buffer to store $n+1$ generating points $\overline{v}_t^{(0)}, \dots, \overline{v}_t^{(n)}$ for the transited-state set $\overline{\mathcal{X}}_t$ and $n+1$ generating points $\widetilde{v}_t^{(0)}, \dots, \widetilde{v}_t^{(n)}$ for the selected coding cell $\widehat{\mathcal{X}}_t$, which are replaced with the generating points at time $t+1$, respectively. Thus, the buffer size is finite. Secondly, the initial generating points $\overline{v}_0^{(0)}, \dots, \overline{v}_0^{(n)}$ at time $t=0$ are not arbitrarily given if we want to achieve the entropy bound $H(A)$, as we discuss in the following remark.

Remark 9 (Initial Points)**.** *Let $J = W^{-1}AW \in \mathbb{R}^{n\times n}$ be the real Jordan form (see Chapter 3.1.4 in [20]) of A, where the transformation matrix is $W \in \mathbb{R}^{n\times n}$. Note that $\forall \overline{\alpha} \in \mathbb{R}_+$, $\overline{\alpha}W$ is still a transformation matrix, since $(\overline{\alpha}W)^{-1}A(\overline{\alpha}W) = J$. Then, select $\overline{v}_t^{(0)}$ and $\overline{\alpha}$ such that $\overline{\mathcal{X}}_0 \subseteq \mathcal{P}(\overline{v}_0^{(0)}, \overline{v}_0^{(0)} + \overline{\alpha}\mathrm{col}_1(W), \dots, \overline{v}_0^{(0)} + \overline{\alpha}\mathrm{col}_n(W))$, where $\mathrm{col}_i(W)$ is the ith column of W. Now, the other n generating points are determined by $\overline{v}_0^{(i)} = \overline{v}_0^{(0)} + \overline{\alpha}\mathrm{col}_i(W)$ ($i \in \{1, \dots, n\}$). We stress that only the initial points are related to the real Jordan form which will not be used for subsequent times $t \in \mathbb{Z}_+$.*

Algorithm 1 Encoder Design

```
Input: $x_t$.
Output: $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t)$.
 1: if $t == 0$ then
 2:     $\overline{v}_0^{(0)}, \dots, \overline{v}_0^{(n)}$ are given by Remark 9;
 3: else
 4:     $\overline{v}_t^{(i)} = A\widetilde{v}_{t-1}^{(i)} - BK\widehat{x}_{t-1}$, $i \in \{0, \dots, n\}$, where $\rho(A-BK) < 1$;
 5: end if
 6: $\delta_t^{(i)} = \|\overline{v}_0^{(i)} - \overline{v}_0^{(0)}\|/(t+1)$, $i \in \{1, \dots, n\}$;
 7: $d_t^{(i)} = \|\overline{v}_t^{(i)} - \overline{v}_t^{(0)}\|/2^{\lceil (\log_2(\|\overline{v}_t^{(i)} - \overline{v}_t^{(0)}\|/\delta_t^{(i)}))^+ \rceil}$, $i \in \{1, \dots, n\}$;
 8: Use (29) and (30) to derive $\kappa_t^* = (\kappa_t^{*(1)}, \dots, \kappa_t^{*(n)})$;
 9: Calculate $\widetilde{v}_t^{(0)}, \dots, \widetilde{v}_t^{(n)}$ by (32);
10: Send $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t)$, which is calculated from (24) or (26) with $\kappa_t = \kappa_t^*$, to the decoder.
11: $\widehat{x}_t \leftarrow$ (33);
```

The line-by-line explanation of Algorithm 1 is given as follows. Firstly, the input and the output for the encoder is the current state x_t obtained from the sensor and the corresponding codeword (see the encoder block in Fig. 1). Secondly, the three stages in Table I from the encoder side are:

1) **Determining coding range.** From Line 1 to Line 5, generating points $\overline{v}_{t+1}^{(i)}$ ($i \in \{1, \dots, n\}$) are calculated: for $t=0$, Line 2 gives the initial points $\overline{v}_0^{(0)}, \dots, \overline{v}_0^{(n)}$ through Remark 9. for $t \in \mathbb{Z}_+$, Line 4 calculates generating points $\overline{v}_t^{(0)}, \dots, \overline{v}_t^{(n)}$ for determining the transited-state set $\overline{\mathcal{X}}_t$, which follows from Remark 7. Line 6 and Line 7 give a method to choose the edge-length constraint $\delta_t^{(i)}$ and the edge length $d_t^{(i)}$ for $i \in \{1, \dots, n\}$ such that $d_t^{(i)} \leq \delta_t^{(i)}$. Note that with properly designing the edge-length constraint $\delta_t^{(i)}$, the convergence process of the edge length $d_t^{(i)}$ can be accurately designed. In this work, we just provide a special case of $\delta_t^{(i)}$.
2) **Encoding.** Lines 8 and 9 finish the process of the coding cell selection based on the current state x_t obtained from the sensor. Firstly, the encoder calculates the coordinate of x_t in the affine coordinate system $(\overline{v}_t^{(0)}, \overline{v}_t^{(1)} - \overline{v}_t^{(0)}, \dots, \overline{v}_t^{(n)} - \overline{v}_t^{(0)})$ by solving
$$\sum_{i=1}^{n} a_t^{(i)} \frac{(\overline{v}_t^{(i)} - \overline{v}_t^{(0)})}{\|\overline{v}_t^{(i)} - \overline{v}_t^{(0)}\|} = x_t - \overline{v}_t^{(0)}, \tag{29}$$
which is a group of linear equations with a unique solution $a_t^{(1)}, \dots, a_t^{(n)} \in \overline{\mathbb{R}}_+$. Secondly, the encoder derives $\kappa_t^* = (\kappa_t^{*(0)}, \dots, \kappa_t^{*(n)})$ via
$$\kappa_t^{*(i)} = \left\lceil \frac{a_t^{(i)}}{d_t^{(i)}} \right\rceil, \quad i \in \{1, \dots, n\}. \tag{30}$$
Thus, the selected coding cell $\widehat{\mathcal{X}}_t = \sigma_t^{(\kappa_t^*)}(\overline{\mathcal{X}}_t)$ is expressed as (23) with $\kappa_t = \kappa_t^*$ and $\widetilde{v}_t^{(\kappa_t, j)} = \widetilde{v}_t^{(j)}$ ($j \in \{0, \dots, n\}$), i.e.,
$$\widehat{\mathcal{X}}_t = \mathcal{P}(\widetilde{v}_t^{(0)}, \dots, \widetilde{v}_t^{(n)}), \tag{31}$$
where
$$\begin{cases} \widetilde{v}_t^{(0)} = \overline{v}_t^{(0)} + \sum_{j=1}^{n} \kappa_t^{*(j)} d_t^{(j)} \dfrac{\overline{v}_t^{(j)} - \overline{v}_t^{(0)}}{\|\overline{v}_t^{(j)} - \overline{v}_t^{(0)}\|}, & i = 0 \\ \widetilde{v}_t^{(i)} = \widetilde{v}_t^{(0)} + d_t^{(i)} \dfrac{\overline{v}_t^{(i)} - \overline{v}_t^{(0)}}{\|\overline{v}_t^{(i)} - \overline{v}_t^{(0)}\|}, & i \in \{1, \dots, n\}. \end{cases} \tag{32}$$
In Line 10, the codeword $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t) = \overline{c}(\mathcal{Q}_t(\overline{\mathcal{X}}_t), \sigma_t^{(\kappa_t^*)}(\overline{\mathcal{X}}_t))$ is encoded by (24) or (26) with $\kappa_t = \kappa_t^*$, and then is sent to the decoder.
3) **Generating quantized state.** In Line 11, the quantized state $\widehat{x}_t$ is determined by
$$\widehat{x}_t = p_t(\widehat{\mathcal{X}}_t) = \frac{1}{2}\sum_{i=1}^{n} (\widetilde{v}_t^{(i)} - \widetilde{v}_t^{(0)}) + \widetilde{v}_t^{(0)}, \tag{33}$$
which follows from Section IV-C and $\widehat{\mathcal{X}}_t = \sigma_t^{(\kappa_t^*)}(\overline{\mathcal{X}}_t)$.

Algorithm 2 gives the details of the decoder design. Similar to Algorithm 1, this algorithm also needs a finite size buffer to store generating points $\overline{v}_t^{(0)}, \dots, \overline{v}_t^{(n)}$ and $\widetilde{v}_t^{(0)}, \dots, \widetilde{v}_t^{(n)}$, and the initial points $\overline{v}_0^{(0)}, \dots, \overline{v}_0^{(n)}$ are the same as those in Algorithm 1.

The line-by-line explanation of Algorithm 2 is given as follows. Firstly, the input and the output for the decoder

Algorithm 2 Decoder Design

```
Input: $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t)$.
Output: $\widehat{x}_t$.
1: The same as Lines 1-7 in Algorithm 1;
2: if $\#\mathcal{K}_t > 1$ then
3:     Use (34) to obtain $\kappa_t^*$;
4:     Derive $\widetilde{v}_t^{(0)}, \dots, \widetilde{v}_t^{(n)}$ by (32);
5: else
6:     $\widetilde{v}_t^{(i)} = \overline{v}_t^{(i)}$ for $i \in \{0, \dots, n\}$;
7: end if
8: Pass $\widehat{x}_t \leftarrow$ (33) to the controller;
```

is the codeword $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t)$ and the estimated-state $\widehat{x}_t$, respectively, which can be found in the decoder block in Fig. 1. Secondly, the three stages in Table I from the encoder side are:

1) **Determining coding range.** This stage is shown in Line 1, which is the same as the determining coding range stage from the encoder side.
2) **Decoding.** From Line 2 to Line 7, the generating points of selected coding cell $\widehat{\mathcal{X}}_t$ are determined by decoding the codeword $c(\mathcal{Q}_t(\overline{\mathcal{X}}_t), x_t)$ sent from the encoder: When $\#\mathcal{K}_t > 1$, the index $\kappa_t^* = (\kappa_t^{*(1)}, \dots, \kappa_t^{*(n)})$ of corresponding coding cell is uniquely determined by
$$\kappa_t^{*(i)} = \sum_{l=\iota_{i-1}+1}^{\iota_i} b_l 2^{l-\iota_{i-1}-1} + 1, \quad i \in \{1, \dots, n\}. \tag{34}$$
Then, the selected coding cell $\widehat{\mathcal{X}}_t$ is determined by (31). When $\#\mathcal{K}_t = 1$, the decoder does not do any decoding, since the encoder sends nothing. In this case, we have $\widehat{\mathcal{X}}_t = \overline{\mathcal{X}}_t$.
3) **Generating quantized state.** In Line 8, the quantized state is determined by (33), and then is sent to the controller.

Note that both Algorithm 1 and Algorithm 2 have low computational complexity. For Algorithm 1, the complexity is at the level of $\mathcal{O}(n^3)$, which is due to solving the linear equation (29). For Algorithm 2, the complexity is $\mathcal{O}(n)$, i.e., with linear complexity.

We would highlight an important part in our design, i.e., the edge-length constraint $\delta_t^{(i)}$ which non-exponentially converges to 0. On the one hand, it makes sure $d_t^{(i)} \le \delta_t^{(i)}$ always holds such that condition (ii) in Theorem 1 is satisfied. On the other hand, it also guarantees $d_t^{(i)} \ge \delta_t^{(i)}/2$ when t is sufficiently large [see (76) in Appendix E], which ensures non-exponential convergence of $d_t^{(i)}$, so as the measure of each coding cell. We can see that the constraint from $\delta_t^{(i)}$ is bidirectional, i.e., $\delta_t^{(i)}/2 \le d_t^{(i)} \le \delta_t^{(i)}$. Note that the edge-length constraint design is very different from the existing methods. In the literature, the edge (usually in a transformed space) decreases periodically (every T time units, where T is the period) which has to be exponentially convergent; while our design can make the edge decrease in every time unit controlled by $\delta_t^{(i)}$. This difference is similar to that between the time-triggered and event-triggered controls, where the edge-length constraint $\delta_t^{(i)}$ triggers the change of $d_t^{(i)}$ and also determines how much $d_t^{(i)}$ should be reduced in each time unit.

Now, we prove that the encoding-decoding process in Algorithm 1 and Algorithm 2 stabilizes the quantized system of (2) with finite length entropy-bound-achieving code.

Theorem 2 (Achieving Entropy Bound with Uniformly Bounded Codeword Length)**.** *Under the designs in Section IV-A, Section IV-B, Section IV-C, and Section IV-D, the quantized system of* (2) *is stabilized by Algorithm 1 and Algorithm 2, with date rate equaling $H(A)$, and the codeword length is upper bounded by*
$$L_t \le n\lceil \log_2(2\|A\|)\rceil, \quad t \in \overline{\mathbb{Z}}_+. \tag{35}$$

Proof: See Appendix E. ■

Remark 10 (Extension to Handling Interfering Signals)**.** *Consider system* (2) *with interfering signal w_t:*
$$x_{t+1} = Ax_t + Bu_t + B_w w_t, \tag{36}$$
where $B_w \in \mathbb{R}^{n\times q}$ and $w_t \in \mathbb{R}^q$ is a q-dimensional exogenous signal generated by an exosystem of the form
$$w_{t+1} = A_w w_t, \tag{37}$$
in which $A_w \in \mathbb{R}^{n\times n}$ is marginally stable (i.e., the interfering signal w_t is bounded) with its eigenvalues satisfying $|\lambda_w| = 1$, and the initial state w_0 is deterministically unknown to the decoder[8] *in a compact set $\mathcal{W}_0$. The design is given as follows:*

i) *Controller: We use the static feedback control law [21]*
$$u_t = -K\widehat{x}_t - K_w\widehat{w}_t, \tag{38}$$
where $\widehat{x}_t$ and $\widehat{w}_t$ are the quantized x_t and w_t, respectively; $K \in \mathbb{R}^{m\times n}$ and $K_w \in \mathbb{R}^{m\times q}$ are control gains.
ii) *Encoding-decoding process for w_t: The encoder and decoder designs are the same as that in Algorithms 1 and 2, respectively, except for the system equation being* (37) *(i.e., Line 4 in Algorithm 1 becomes $\overline{v}_t^{(i)} = A_w\widetilde{v}_{t-1}^{(i)}$).*
iii) *Encoding-decoding process for x_t: At each t, we encode and decode w_t first so that $\widehat{w}_t$ is derived. Then, we still use Algorithms 1 and 2 to encode and decode x_t, respectively, where Line 4 in Algorithm 1 is replaced by: $\overline{v}_t^{(i)} = \left[1 + \frac{D(\widehat{\mathcal{W}}_{t-1})}{\|\widetilde{v}_{t-1}^{(i)}\|}\right] A\widetilde{v}_{t-1}^{(i)} - BK\widehat{x}_{t-1} - B_wK_w\widehat{w}_{t-1}$ for $i \in \{0, \dots, n\}$, where $\rho(A - BK) < 1$ and $BK_w = B_w$.*

It is not hard to prove $\lim_{t\to\infty} x_t = 0$ *with $R = H(A)$. Here we give an outline of the proof: Firstly, we need to guarantee* $\lim_{t\to\infty} x_t = 0$ *by applying the input-to-state stability technique to the closed-loop system $x_{t+1} = (A - BK)x_t + BK(x_t - \widehat{x}_t) + (B_w - BK_w)\widehat{w}_t + BK_w(w_t - \widehat{w}_t)$ with $x_t - \widehat{x}_t$ and $w_t - \widehat{w}_t$ regarded as the inputs, where $(B_w - BK_w)\widehat{w}_t = 0$ as $BK_w = B_w$. Note that* $\lim_{t\to\infty}(x_t - \widehat{x}_t) = 0$ *and* $\lim_{t\to\infty}(w_t - \widehat{w}_t) = 0$ *hold since ii) and iii) give the asymptotic observations of w_t and x_t, respectively. Secondly, we need to prove $R = H(A)$. For ii), its data rate is* 0*, as $H(A_w) = 0$. For iii), the data rate is $H(A)$, which can*

[8]We assume the pair $\left(\begin{bmatrix} A & B_w \\ 0 & A_w \end{bmatrix}, \begin{bmatrix} I & 0 \end{bmatrix}\right)$ is observable. The decoder can observe w_t within finite time steps. Without loss of generality, we assume w_0 is known to the encoder.

be proved by making contradictions to assuming any data rate greater than $H(A)$, since $\lim_{t\to\infty} D(\widehat{\mathcal{W}}_{t-1}) = 0$ and $|\lambda_w| = 1$.

V. Conclusion and Future Work

In this work, we have designed entropy-achieving codes with finite blocklength 1 (i.e., zero delay) for linear system stabilization, and such a design has no precedent in the literature. A coding-quantizing framework established the relationship between the system state, codeword, and quantization. More significantly, an encoding-decoding framework has been provided such that each system state is encoded and decoded within only one time unit. With this zero-delay framework, we have proposed a set of sufficient conditions as a criterion for obtaining a zero-delay entropy-achieving code. Finally, by our proposed criterion, an entropy-achieving code with zero delay is designed, where the quantization is directed conducted in the original space rather than the transformed space (related to the real Jordan form), and all the codeword lengths are uniformly bounded.

For future work, we will consider a realistic constraint on the maximum number of bits that can be transmitted in every time unit. Note that the constraint on the number of bits can be caused by the limited amount of energy available at the encoder side in each time unit. Even though in Theorem 2 the codeword length is uniformly bounded by $L = n\lceil \log_2(2\|A\|)\rceil$, further consideration is required when less than L bits are permitted for transmission in every time unit. In that case, our encoding-decoding framework cannot achieve zero delay anymore, since there always exist some codewords which cannot be fully transmitted in one time unit. Therefore, it is very interesting and practical to consider such a constrained case in our future work.

Appendix A
Proof of Theorem 1

To start with, we give a lemma for the matrix power norm bound as a preparation, which is an enhanced version of Theorem 3.5 in [22] and can be proved by using a similar idea.

Lemma 3. *Given $F \in \mathbb{R}^{n\times n}$, for each $\gamma > \rho(F)$, there exists $M_\gamma \geq 1$, such that $\forall k \in \overline{\mathbb{Z}}_+$, $\|F^k\| \leq M_\gamma \gamma^k$ holds.*

Now, we can prove Theorem 1. This proof contains two steps: the first step proves that the quantized system is stabilized by conditions (i) and (ii); and the second step proves that the data rate spent by the stabilized system is exactly the entropy bound $H(A)$ under conditions (iii)–(v).

1) Since pair (A, B) is controllable, there exists linear feedback matrix $K \in \mathbb{R}^{m\times n}$ such that $\rho(A - BK) < 1$, which implies condition (i) is satisfiable. For any K satisfying condition (i), we set $\widehat{x}_t = x_t - (x_t - \widehat{x}_t)$, and system (2) can be rewritten as

$$x_{t+1} = (A - BK)x_t + BK(x_t - \widehat{x}_t), \quad t \in \overline{\mathbb{Z}}_+. \tag{39}$$

The solution to (39) is

$$\begin{aligned} x_t =&(A - BK)^t x_0 + \sum_{\tau=0}^{t-1}(A - BK)^{t-1-\tau}BK(x_\tau - \widehat{x}_\tau),\\ =&(A - BK)^t x_0 + \sum_{\tau=0}^{\underline{t}-1}(A - BK)^{t-1-\tau}BK(x_\tau - \widehat{x}_\tau)\\ &+ \sum_{\tau=\underline{t}}^{t-1}(A - BK)^{t-1-\tau}BK(x_\tau - \widehat{x}_\tau), \end{aligned} \tag{40}$$

where the first item goes to 0 as $t \to \infty$, and similarly the second item also goes to 0 as $t \to \infty$ for any $\underline{t} \in \mathbb{Z}_+$. Now, we prove the third item also goes to 0 for a proper $\underline{t} \in \mathbb{Z}_+$. Since $\rho(A - BK) < 1$, it follows from Lemma 3 that for any $\gamma \in (\rho(A - BK),\ 1)$, there exists a $M_\gamma \geq 1$ such that $\|(A - BK)^{t-1-\tau}\| \leq M_\gamma \gamma^{t-1-\tau}$. Then, $\forall \varepsilon > 0$, there exists a $\underline{t} \in \mathbb{Z}_+$ such that $\forall \tau \geq \underline{t}$, $\|BK(x_\tau - \widehat{x}_\tau)\| \leq \varepsilon(1-\gamma)/M_\gamma$. This is because $\lim_{\tau\to\infty}\|x_\tau - \widehat{x}_\tau\| = 0$, which is guaranteed by $\|x_\tau - \widehat{x}_\tau\| \leq \overline{D}_{\tau-1}$ and $\lim_{\tau\to\infty}\overline{D}_{\tau-1} = 0$ [see condition (ii)]. Then, the norm bound of the third item in the right-hand side of (40) is

$$\begin{aligned} &\Big\|\sum_{\tau=\underline{t}}^{t-1}(A - BK)^{t-1-\tau}BK(x_\tau - \widehat{x}_\tau)\Big\|\\ \leq& \sum_{\tau=\underline{t}}^{t-1}\left\|(A - BK)^{t-1-\tau}BK(x_\tau - \widehat{x}_\tau)\right\|,\\ \leq& \Big(\sum_{\tau=\underline{t}}^{t-1} M_\gamma \gamma^{t-1-\tau}\Big)\frac{\varepsilon(1-\gamma)}{M_\gamma} \overset{(a)}{\leq} \varepsilon, \end{aligned} \tag{41}$$

where inequality (a) follows from $\sum_{\tau=\underline{t}}^{t-1}\gamma^{t-1-\tau} \leq \sum_{\tau=-\infty}^{t-1}\gamma^{t-1-\tau} = 1/(1-\gamma)$. Inequality (41) means the third item in the right-hand side of (40) also goes to 0 as $t \to \infty$. Hence, the quantized system is stabilized.

2) We label those coding-cover sequences satisfying conditions (ii)–(v) as $\mathfrak{Q}^a = (\mathcal{Q}_t^a(\overline{\mathcal{X}}_t))_{t\in\overline{\mathbb{Z}}_+}$, where the superscript a refers to the achievability. By Definition 3 and condition (iv), the codeword length at time t is

$$L_t = \log_2\left\lceil \#\mathcal{Q}_t^a(\overline{\mathcal{X}}_t)\right\rceil = \log_2 \#\mathcal{Q}_t^a(\overline{\mathcal{X}}_t). \tag{42}$$

In addition, with condition (iii), we can derive that $\forall t \geq \underline{t}$,

$$\log_2 \#\mathcal{Q}_t^a(\overline{\mathcal{X}}_t) = \log_2 \frac{\mu_n(\overline{\mathcal{X}}_t)}{\mu_n(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t))}, \tag{43}$$

holds for all $\kappa_t \in \mathcal{K}_t$. Noticing that $\mu_n(\overline{\mathcal{X}}_t) = |\det(A)|\mu_n(\widehat{\mathcal{X}}_{t-1})$ [with (10) and observing that $u_{t-1} = \psi_{t-1}(\widetilde{x}_{t-1})$ is a constant w.r.t. $\widetilde{x}_{t-1}$], we have

$$\begin{aligned} \sum_{t=\underline{t}}^{T-1}\log_2 \#\mathcal{Q}_t^a(\overline{\mathcal{X}}_t) =& \log_2 \prod_{t=\underline{t}+1}^{T-1}\frac{|\det(A)|\mu_n(\widehat{\mathcal{X}}_{t-1})}{\mu_n(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t))}\\ =& \log_2 \frac{|\det(A)|^{T-\underline{t}-1}\mu_n(\widehat{\mathcal{X}}_{\underline{t}})}{\mu_n(\sigma_{T-1}^{(\kappa_{T-1})}(\overline{\mathcal{X}}_{T-1}))}, \end{aligned} \tag{44}$$

where $T \geq \underline{t}+2$. Then, the data rate [defined in (12)] under the coding-cover sequence $\mathfrak{Q}^a$ is

$$\begin{aligned} & R(\Sigma, \overline{\mathcal{X}}_0, \mathfrak{Q}^a, \mathfrak{C}, \Pi, \Psi) \\ &= \varliminf_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} L_t, \\ &\overset{(a)}{=} \varliminf_{T\to\infty} \frac{1}{T}\sum_{t=\underline{t}+1}^{T-1} L_t, \\ &\overset{(b)}{=} \varliminf_{T\to\infty} \frac{1}{T}\log_2 \frac{|\det(A)|^{T-\underline{t}-1}\mu_n(\widehat{\mathcal{X}}_{\underline{t}})}{\mu_n(\sigma_{T-1}^{(\kappa_{T-1})}(\overline{\mathcal{X}}_{T-1}))}, \end{aligned} \tag{45}$$

where equality (a) follows from

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{\underline{t}} L_t = 0. \tag{46}$$

Equality (b) in (45) is derived from (42) and (44). Note that

$$\begin{aligned} \lim_{T\to\infty} \frac{\log_2 \mu_n(\widehat{\mathcal{X}}_{\underline{t}})}{T} &\overset{(c)}{=} 0, \\ \varliminf_{T\to\infty} \frac{\log_2 \mu_n(\sigma_{T-1}^{(\kappa_{T-1})}(\overline{\mathcal{X}}_{T-1}))}{T} &\overset{(d)}{=} 0, \end{aligned} \tag{47}$$

where (c) follows from $T\to\infty$ and $-\infty < \log_2 \mu_n(\widehat{\mathcal{X}}_{\underline{t}}) < \infty$ [since $0 < \mu_n(\widehat{\mathcal{X}}_{\underline{t}}) \leq \mu_n(\overline{\mathcal{X}}_0)$]. Equality (d) is established by condition (v). Then, with (47), we can rewrite (45) as

$$\begin{aligned} & R(\Sigma, \overline{\mathcal{X}}_0, \mathfrak{Q}^a, \mathfrak{C}^a, \Pi, \Psi^a) \\ &= \log_2 |\det(A)| = \sum_{\lambda\in\mathrm{spec}(A)} \log_2 |\lambda|, \end{aligned} \tag{48}$$

where Ψ^a refers to the control law sequence satisfying condition (i), and $\mathfrak{C}^a$ stands for the coding cover sequence satisfying condition (ii). Since the quantized system is stabilized under conditions (i) and (ii) [see step 1) in this proof], (15) holds.

According to condition (iv), we have $L_t = \log_2(\#\mathcal{Q}_t^a(\overline{\mathcal{X}}_t)) \leq L$, i.e., the codeword lengths for the entropy-achieving codes are finite. ■

Appendix B
Proof of the Non-Exponential Stability in Theorem 1

Firstly, we give a lower bound on the worst case $\|x_t\|$ (after decoding) in the following lemma.

Lemma 4 (Lower Bound of $\|x_t\|$)**.** *For $t\in\overline{\mathbb{Z}}_+$, we have*

$$\max_{\kappa_t\in\mathcal{K}_t} \max_{x_t\in\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)} \|x_t\| \geq \frac{\overline{D}_t}{2}. \tag{49}$$

Proof: For $\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)$, where $t\in\overline{\mathbb{Z}}_+$, we have

$$D(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)) = \max_{x,y\in\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)} \|x-y\|, \tag{50}$$

i.e., the maximum $\|x-y\|$ gives the diameter of coding cell $\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)$. We select a maximizer (x^*,y^*) such that

$$(x^*,y^*) = \operatorname*{argmax}_{x,y\in\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)} \|x-y\|, \tag{51}$$

which means $\|x^*-y^*\| = D(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t))$. Then, we have

$$\begin{aligned} D(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)) &= \|x^* + (-y^*)\|, \\ &\leq \|x^*\| + \|y^*\|, \\ &\leq 2\max\{\|x^*\|, \|y^*\|\}, \\ &= 2\max_{x_t\in\{x^*,y^*\}} \|x_t\|, \\ &\leq 2\max_{x_t\in\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)} \|x_t\|. \end{aligned} \tag{52}$$

This implies

$$\max_{x_t\in\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)} \|x_t\| \geq \frac{1}{2} D(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)). \tag{53}$$

Noticing that $\max_{\kappa_t\in\mathcal{K}_t} D(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)) = \overline{D}_t$, we get (49). ■

Since x_t can possibly be in any coding cell $\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)$ after decoding, the worst case $\|x_t\|$ is not smaller than $\overline{D}_t/2$ as given in Lemma 4. If we assume x_t converges exponentially fast, then the following would hold

$$0 > \max_{\kappa_t\in\mathcal{K}_t} \max_{x_t\in\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)} \log_2 \|x_t\| \geq \frac{\log_2 \overline{D}_t}{2}, \tag{54}$$

which indicates $\log_2 \overline{D}_t < 0$. From our discussions in Remark 4, we know that it will contradict condition (v) in Theorem 1. Therefore, x_t converges asymptotically but not exponentially. ■

Appendix C
Proof of Proposition 1

We use a similar way as Proposition 3.2 in [7] to give this proof. If x_t converges exponentially fast, then (17) holds. At $t\in\mathbb{Z}_+$, we define the following set

$$\Omega_{u_0,\ldots,u_t} = \left\{x_0 \colon \|x_t\| \leq \alpha\beta^{-t}\right\}, \tag{55}$$

where $x_t = A^t x_0 + \sum_{\tau=0}^{t-1} A^{t-1-\tau}Bu_\tau$. Note that $\Omega_{u_0,\ldots,u_t} \subseteq \overline{\mathcal{X}}_0$ indicates the set of $x_0\in\overline{\mathcal{X}}_0$ satisfying (17) under control sequence $u_0,\ldots,u_t$. To control all the initial values $x_0\in\overline{\mathcal{X}}_0$ to satisfy (17), we need several control sequences such that the union of their corresponding $\Omega_{u_0,\ldots,u_t}$ contains $\overline{\mathcal{X}}_0$, since otherwise there would exist $x_0\in\overline{\mathcal{X}}_0\backslash\bigcup_{u_0,\ldots,u_t}\Omega_{u_0,\ldots,u_t}$ violating (17). Thus, $\prod_{\tau=0}^{t} M_\tau$ (recall that M_τ is the coding alphabet at time τ) must be no less than the number of sets $\Omega_{u_0,\ldots,u_t}$ to cover $\overline{\mathcal{X}}_0$. Observe that the measure of $\Omega_{u_0,\ldots,u_t}$ is

$$\begin{aligned} \mu_n(\Omega_{u_0,\ldots,u_t}) &= |\det(A)|^{-t}\mu_n\left(\{x_t\colon \|x_t\|\leq\alpha\beta^{-t}\}\right) \\ &= |\det(A)|^{-t}\varpi_n(\alpha\beta^{-t})^n. \end{aligned} \tag{56}$$

Then, a lower bound of $R_t := \frac{1}{t}\sum_{\tau=0}^{t-1}\log_2 M_\tau$ can be derived by using the measures:

$$\begin{aligned} R_t &\geq \frac{1}{t}\log_2 \frac{\mu_n(\overline{\mathcal{X}}_0)}{|\det(A)|^{-t}\varpi_n(\alpha\beta^{-t})^n}, \\ &= H(A) + n\log_2\beta + \frac{1}{t}\log_2\frac{\mu_n(\overline{\mathcal{X}}_0)}{\varpi_n\alpha^n}. \end{aligned} \tag{57}$$

It combined with $R = \varliminf_{t\to\infty} R_t$ gives (18). ■

Appendix D
Proof of Lemma 2

As φ is invertible, we have $\varphi(\cdot) = F\cdot + b$, where $F \in \mathbb{R}^{n\times n}$ is invertible, and $b \in \mathbb{R}^n$. This means $\varphi(\sum_{i=1}^n \theta_i (v^{(i)} - v^{(0)}) + v^{(0)}) = F(\sum_{i=1}^n \theta_i (v^{(i)} - v^{(0)}) + v^{(0)}) + b = \sum_{i=1}^n \theta_i [(Fv^{(i)} + b) - (Fv^{(0)} + b)] + (Fv^{(0)} + b) = \sum_{i=1}^n \theta_i [\varphi(v^{(i)}) - \varphi(v^{(0)})] + \varphi(v^{(0)})$. Thus,

$$\begin{aligned}&\varphi(\mathcal{P}(v^{(0)}, \dots, v^{(n)}))\\ &= \Big\{ \sum_{i=1}^n \theta_i [\varphi(v^{(i)}) - \varphi(v^{(0)})] + \varphi(v^{(0)}) \colon \theta_1, \dots, \theta_n \in [0, 1] \Big\}.\end{aligned} \tag{58}$$

Since F is invertible, vectors $\varphi(v^{(i)}) - \varphi(v^{(0)})$ ($i \in \{1, \dots, n\}$) are linearly independent, thus the right-hand side of (58) is $\mathcal{P}(\varphi(v^{(0)}), \dots, \varphi(v^{(n)}))$. ∎

Appendix E
Proof of Theorem 2

Before start, we give the following lemma to calculate the measure of any parallelepiped in Definition 10, which can be easily obtained from Corollary 1 in [19].

Lemma 5 (Measure of Parallelepiped)**.** *Given a parallelepiped $\mathcal{P}(v^{(0)}, \dots, v^{(n)})$, its measure is*

$$\begin{aligned}&\mu_n(\mathcal{P}(v^{(0)}, \dots, v^{(n)})) = \\ &\qquad \left| \det \left(\begin{bmatrix} v^{(1)} - v^{(0)} & \dots & v^{(n)} - v^{(0)} \end{bmatrix} \right) \right|.\end{aligned} \tag{59}$$

In this proof, we prove that by the control law $u_t = -K\widehat{x}_t$ with $\rho(A - BK) < 1$, Algorithm 1, and Algorithm 2, all the five conditions in Theorem 1 are satisfied.

Satisfying condition (i). Since we use the control law $u_t = -K\widehat{x}_t$ such that $\rho(A - BK) < 1$, condition (i) in Theorem 1 holds.

Satisfying condition (ii). According to Line 6 and Line 7 in Algorithm 1, we know that $\forall i \in \{1, \dots, n\}$ and $\forall t \in \overline{\mathbb{Z}}_+$, $d_t^{(i)} \le \delta_t^{(i)}$ holds. As $\lim_{t\to\infty} \delta_t^{(i)} = 0$, we can derive $\lim_{t\to\infty} d_t^{(i)} = 0$ for all $i \in \{1, \dots, n\}$. This means $\lim_{t\to\infty} \overline{D}_t = 0$, because $\overline{D}_t \le \sum_{i=1}^n d_t^{(i)}$ always holds.

Satisfying condition (iii). Since the coding cells are congruent for any given time (see Remark 7), with (21) in Remark 7, we know that condition (iii) in Theorem 1 holds with $\underline{t} = 1$.

Satisfying condition (iv). By Line 7 in Algorithm 1, for any axis $\overline{v}_t^{(i)} - \overline{v}_t^{(0)}$ ($i \in \{1, \dots, n\}$), the maximum number of partitions $\overline{\kappa}_t^{(i)}$ satisfies $\log_2 \overline{\kappa}_t^{(i)} \in \overline{\mathbb{Z}}_+$. More specifically,

$$\overline{\kappa}_t^{(i)} = 2^{\left\lceil \left(\log_2 \frac{\|\overline{v}_t^{(i)} - \overline{v}_t^{(0)}\|}{\delta_t^{(i)}} \right)^+ \right\rceil}. \tag{60}$$

For $t = 0$, we can obtain $\|\overline{v}_1^{(i)} - \overline{v}_1^{(0)}\| \le \|A\| \|\widetilde{v}_0^{(i)} - \widetilde{v}_0^{(0)}\| \le \|A\| \|\overline{v}_0^{(i)} - \overline{v}_0^{(0)}\| = \|A\| \delta_0^{(i)}$, and thus $\log_2 \overline{\kappa}_0^{(i)} = \lceil (\log_2 (\|\overline{v}_1^{(i)} - \overline{v}_1^{(0)}\| / \delta_0^{(i)}))^+ \rceil \le \lceil (\log_2(\|A\|))^+ \rceil = \lceil \log_2(\|A\|) \rceil$ holds. For $t \in \mathbb{Z}_+$, we have $\|\overline{v}_t^{(i)} - \overline{v}_t^{(0)}\| \le \|A\| d_{t-1}^{(i)} \le \|A\| \delta_{t-1}^{(i)}$ and $\delta_{t-1}^{(i)} / \delta_t^{(i)} = (t + 1)/t$, which implies $\log_2 \overline{\kappa}_t^{(i)} \le \lceil \log_2 [\|A\| (t + 1)/t] \rceil \le \lceil \log_2(2\|A\|) \rceil$. Therefore, for $L = n\lceil \log_2(2\|A\|) \rceil$, we can derive $\log_2(\#\mathcal{Q}_t(\overline{\mathcal{X}}_t)) = \log_2(\#\mathcal{K}_t) = \sum_{i=1}^n \log_2 \overline{\kappa}_t^{(i)} \in \overline{\mathbb{Z}}_+ \bigcap [0, \ L]$, which establishes condition (iv) in Theorem 1. Additionally, because $L_t \le L$ holds for all $t \in \overline{\mathbb{Z}}_+$, we can derive (35) in Theorem 2.

Satisfying condition (v). We divide this part of the proof into three steps: In the first step, we derive an important inequality in (73) to describe the relationship among the "total uncertainty", the edge length of a coding cell, and the maximum number of partitions, in each axis. In the second step, based on (73), an upper bound in (78) is given for the number of all potential coding cells. In the third step, we complete the entire proof.

Step 1: Let $\overline{w}_t^{(i)} = \overline{v}_t^{(i)} - \overline{v}_t^{(0)}$ and $\widetilde{w}_t^{(i)} = \widetilde{v}_t^{(i)} - \widetilde{v}_t^{(0)}$ for all $i \in \{1, \dots, n\}$ and $t \in \overline{\mathbb{Z}}_+$. For $t \in \overline{\mathbb{Z}}_+$,

$$\widetilde{w}_{t+1}^{(i)} \overset{(a)}{=} \frac{1}{\overline{\kappa}_{t+1}^{(i)}} \overline{w}_{t+1}^{(i)} \overset{(b)}{=} \frac{1}{\overline{\kappa}_{t+1}^{(i)}} A \widetilde{w}_t^{(i)}, \tag{61}$$

where $\overline{\kappa}_{t+1}^{(i)}$ is the maximum partition number for axis $\overline{w}_{t+1}^{(i)} = \overline{v}_{t+1}^{(i)} - \overline{v}_{t+1}^{(0)}$, and (a) follows from the congruence of coding cells. Equality (b) is established by Line 4 in Algorithm 1. With (61), we get

$$\Big(\prod_{\tau=1}^t \overline{\kappa}_\tau^{(i)} \Big) \widetilde{w}_t^{(i)} = A^t \widetilde{w}_0^{(i)}, \quad t \in \mathbb{Z}_+, \ i \in \{1, \dots, n\}. \tag{62}$$

Then, we define[9]

$$\breve{w}_t^{(i)} := \widetilde{w}_t^{(i)} \prod_{\tau=1}^t \overline{\kappa}_\tau^{(i)}. \tag{63}$$

With (62) and (63), we have

$$\breve{w}_t^{(i)} = A^t \widetilde{w}_0^{(i)}, \quad t \in \mathbb{Z}_+, \ i \in \{1, \dots, n\}. \tag{64}$$

From Remark 9, we know that

$$W := \Big[\widetilde{w}_0^{(1)}, \dots, \widetilde{w}_0^{(n)} \Big] \tag{65}$$

is a transformation matrix such that $J = W^{-1}AW$. By Theorem 3.4.1.5 in [20], the real Jordan form is a real block diagonal matrix

$$J = \bigoplus_{s=1}^S J_s, \tag{66}$$

where the spectrum of each $J_s \in \mathbb{R}^{r_s \times r_s}$ has only one element $\lambda_s \in \mathbb{C}$, i.e., $\text{spec}(J_s) = \{\lambda_s\}$. By (64), (65), and (66), the following is obtained

$$\Big[\breve{w}_t^{(1)}, \dots, \breve{w}_t^{(n)} \Big] = A^t \Big[\widetilde{w}_0^{(1)}, \dots, \widetilde{w}_0^{(n)} \Big] = WJ^t. \tag{67}$$

We group the columns of W as

$$W = \Big[\widetilde{W}_0^{(1)}, \dots, \widetilde{W}_0^{(S)} \Big], \tag{68}$$

[9]We note that $\breve{w}_t^{(i)}$ corresponds to the i^{th} edge of the parallelepiped formed by the coding cells $\sigma_t^{(\kappa_t)}$ indexed by all potential codewords from time steps 1 to t.

where $\widetilde{W}_0^{(s)} := \big[\widetilde{w}_0^{(r_{s-1}+1)}, \dots, \widetilde{w}_0^{(r_s)}\big]$ ($s \in \{1, \dots, S\}$) and $r_0 := 0$. Then, with (66) and (67), we have

$$\begin{aligned}\breve{W}_t^{(s)} &:= \Big[\breve{w}_t^{(r_{s-1}+1)}, \dots, \breve{w}_t^{(r_s)}\Big] = A^t \widetilde{W}_0^{(s)} \\ &= \widetilde{W}_0^{(s)} J_s^t, \quad s \in \{1, \dots, S\}. \end{aligned} \tag{69}$$

For any column $\breve{w}_t^{(i)}$ in $\breve{W}_t^{(s)}$ ($s \in \{1, \dots, S\}$), we have

$$\begin{aligned}\big\|\breve{w}_t^{(i)}\big\| &= \big\|\widetilde{W}_0^{(s)} J_s^t e_{r_s}^{(i-r_{s-1})}\big\| \\ &\le \big\|\widetilde{W}_0^{(s)}\big\|_F \big\|J_s^t\big\| \big\|e_{r_s}^{(i-r_{s-1})}\big\| = \big\|\widetilde{W}_0^{(s)}\big\|_F \big\|J_s^t\big\|, \end{aligned} \tag{70}$$

where $e_{r_s}^{(i-r_{s-1})} \in \mathbb{R}^n$ is a column vector whose $(i - r_{s-1})^{\text{th}}$ entry is 1 and other entries are 0. Note that in (70) $s = s(i)$ is a function of i, and it has the following form

$$s(i) = \mathbf{1}(i) + \mathbf{1}(i - r_1) + \dots + \mathbf{1}(i - r_{S-1}), \tag{71}$$

where $\mathbf{1}(\cdot)$ is the Heaviside step function. In the rest of this proof, we use s instead of $s(i)$, for the sake of simplicity. Since the real Jordan form J_s is similar to one (or two) complex Jordan block(s), for sufficiently large $\underline{t} \in \mathbb{Z}_+$, its power norm has the following bound[10]

$$\big\|J_s^t\big\| \le t^{r_s} |\lambda_s|^t, \quad t \ge \underline{t}. \tag{72}$$

Hence, inequality (70) can be rewritten as

$$\big\|\breve{w}_t^{(i)}\big\| \le t^{r_s} |\lambda_s|^t \big\|\widetilde{W}_0^{(s)}\big\|_F, \quad t \ge \underline{t}. \tag{73}$$

Step 2: Recall that $\breve{w}_t^{(i)} = \widetilde{w}_t^{(i)} \prod_{\tau=1}^t \overline{\kappa}_\tau^{(i)}$ [see (63)], where $\overline{\kappa}_\tau^{(i)}$ is either equal to 1 (no partitioning occurs for axis $\overline{w}_\tau^{(i)}$) or greater than 1 (partitioning occurs). Now, we define the time set of partitioning occurring w.r.t. axis $\overline{w}_t^{(i)}$ as

$$\mathcal{T}_{\text{partition}}^{(i)} = \left\{t \in \overline{\mathbb{Z}}_+ : \overline{\kappa}_t^{(i)} > 1\right\}, \tag{74}$$

which is non-empty[11]. We label the k^{th} smallest element in $\mathcal{T}_{\text{partition}}^{(i)}$ as $t_{k,i}$. Thus, $\forall t \ge \max\{\underline{t}, t_{1,i}\}$, we can find a $k \in \mathbb{Z}_+$ such that $t \in [t_{k,i}, t_{k+1,i})$, and

$$\begin{aligned}\prod_{\tau=1}^{t} \overline{\kappa}_\tau^{(i)} = \prod_{\tau=1}^{t_{k,i}} \overline{\kappa}_\tau^{(i)} &\overset{(c)}{\le} \frac{t_{k,i}^{r_s} |\lambda_s|^{t_{k,i}} \big\|\widetilde{W}_0^{(s)}\big\|_F}{\|\widetilde{w}_{t_{k,i}}^{(i)}\|} \\ &\le \frac{t^{r_s} |\lambda_s|^t \big\|\overline{W}_0^{(s)}\big\|_F}{\|\widetilde{w}_{t_{k,i}}^{(i)}\|}, \end{aligned} \tag{75}$$

where (c) follows from (63) and (73). Now, we want to find a lower bound on $\|\widetilde{w}_{t_{k,i}}^{(i)}\|$ to upper bound the right-hand side of (75), which relies on rewriting Line 7 of Algorithm 1 that

$$\begin{aligned}\|\widetilde{w}_{t_{k,i}}^{(i)}\| &= \|\overline{w}_{t_{k,i}}^{(i)}\| / 2^{\left\lceil \left(\log_2(\|\overline{w}_{t_{k,i}}^{(i)}\| / \delta_{t_{k,i}}^{(i)})\right)^+ \right\rceil} \\ &\overset{(d)}{=} \|\overline{w}_{t_{k,i}}^{(i)}\| / 2^{\left\lceil \log_2(\|\overline{w}_{t_{k,i}}^{(i)}\| / \delta_{t_{k,i}}^{(i)}) \right\rceil} \\ &\overset{(e)}{\ge} \frac{\delta_{t_{k,i}}^{(i)}}{2} \overset{(f)}{=} \frac{\|\overline{w}_0^{(i)}\|}{2(t_{k,i}+1)} \ge \frac{\|\overline{w}_0^{(i)}\|}{2(t+1)}, \end{aligned} \tag{76}$$

where the substitutions $d_{t_{k,i}}^{(i)} = \|\widetilde{w}_{t_{k,i}}^{(i)}\|$ and $\|\overline{w}_{t_{k,i}}^{(i)}\| = \|\overline{v}_{t_{k,i}}^{(i)} - \overline{v}_{t_{k,i}}^{(0)}\|$ are made in the first line; equality (b) holds since $\overline{\kappa}_{t_{k,i}}^{(i)} > 1$ (partitioning occurs) at $t_{k,i}$ so that $\|\overline{w}_{t_{k,i}}^{(i)}\| = \overline{\kappa}_{t_{k,i}}^{(i)} \delta_{t_{k,i}}^{(i)} > \delta_{t_{k,i}}^{(i)}$; inequality (e) follows from $\lceil \cdot \rceil \le \cdot + 1$; equality (f) is established by Line 6 in Algorithm 1. Then, inequality (75) can be rewritten as

$$\prod_{\tau=1}^{t} \overline{\kappa}_\tau^{(i)} \le \frac{2(t+1) t^{r_s} |\lambda_s|^t \big\|\overline{W}_0^{(s)}\big\|_F}{\|\overline{w}_0^{(i)}\|}, \tag{77}$$

which means $\forall t \ge \max\{\underline{t}, t_{1,1}, \dots, t_{1,n}\}$, the following holds

$$\prod_{\tau=1}^{t} \#\mathcal{K}_\tau = \prod_{\tau=1}^{t} \prod_{i=1}^{n} \overline{\kappa}_\tau^{(i)} \le \prod_{i=1}^{n} \frac{2(t+1) t^{r_s} |\lambda_s|^t \big\|\overline{W}_0^{(s)}\big\|_F}{\|\overline{w}_0^{(i)}\|}. \tag{78}$$

Step 3: For the measure of each coding cell at time $t = 0$, we have

$$\mu_n\left(\sigma_0^{(\kappa_0)}(\mathcal{X}_0)\right) = \mu_n(\mathcal{P}(\widetilde{v}_0^{(0)}, \dots, \widetilde{v}_0^{(n)})) = |\det(W)|, \tag{79}$$

which follows from (65) and Lemma 5. $\forall t \in \mathbb{Z}_+$, we have

$$\begin{aligned}\mu_n\left(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)\right) &= \frac{\mu_n(\mathcal{P}(\overline{v}_t^{(0)}, \dots, \overline{v}_t^{(n)}))}{\#\mathcal{K}_t} \\ &= \frac{|\det(A)| \mu_n(\mathcal{P}(\widetilde{v}_t^{(0)}, \dots, \widetilde{v}_t^{(n)}))}{\#\mathcal{K}_t}. \end{aligned} \tag{80}$$

By (79) and (80), the measure of each coding cell at any given time t is

$$\mu_n\left(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)\right) = \frac{|\det(A)|^t |\det(W)|}{\prod_{\tau=1}^t \#\mathcal{K}_\tau}. \tag{81}$$

Combining it with (78), we can derive

$$\begin{aligned}\frac{\log_2 \mu_n\left(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)\right)}{t} &\ge \frac{\log_2(|\det(A)|^t |\det(W)|)}{t} \\ &\quad - \sum_{i=1}^{n} \frac{\log_2\left(\frac{2\|\overline{W}_0^{(s)}\|_F}{\|\widehat{w}_0^{(i)}\|}(t+1) t^{r_s} |\lambda_s|^t\right)}{t}. \end{aligned} \tag{82}$$

We stress that as $t \to \infty$ limit exists in the right-hand side of (82) and equals 0, since when $t \to \infty$ both the first and second items go to $|\det(A)|$. Note that in the second item, s is a function of i [see (71)] such that each s appears exactly r_s times for $i \in \{r_{s-1}+1, \dots, r_s\}$. As $t \to \infty$, the second item goes to $\sum_{s=1}^S r_s \log_2 |\lambda_s| = \log_2 |\det(A)|$. Then, observing that $\log_2 \mu_n\left(\sigma_t^{(\kappa_t)}(\overline{\mathcal{X}}_t)\right)/t \le 0$ holds for sufficiently large t, we can obtain condition (v) in Theorem 1. ■

[10]In [12], a similar but tighter bound can be found, while our bound is simpler and good enough for this proof.

[11]We can show that $\mathcal{T}_{\text{partition}}^{(i)}$ is a non-empty set by contradiction: If we assume $\mathcal{T}_{\text{partition}}^{(i)}$ is an empty set, i.e., no partitioning occurs, then $\overline{w}_t^{(i)}$ (where $\overline{w}_t^{(i)}$ is an eigenvector w.r.t. $|\lambda_s| \ge 1$) is non-decreasing with $t \in \overline{\mathbb{Z}}_+$, i.e., $\|\overline{w}_{t+1}^{(i)}\| = \|A\widetilde{w}_t^{(i)}\| = \|\lambda_s \widetilde{w}_t^{(i)}\| = |\lambda_s| \|\widetilde{w}_t^{(i)}\| \ge \|\widetilde{w}_t^{(i)}\| = \|\overline{w}_t^{(i)}\|$. Noticing that $\lim_{t\to\infty} \delta_t^{(i)} = 0$ (Line 6 in Algorithm 1), we know that partitions must occur. It contradicts the no partitioning assumption. Thus, $\mathcal{T}_{\text{partition}}^{(i)}$ must be non-empty. Furthermore, we can verify that $\mathcal{T}_{\text{partition}}^{(i)}$ is a countably infinite set.

References

[1] J. Baillieul, "Feedback designs for controlling device arrays with communication channel bandwidth constraints," in *ARO Workshop on Smart Structures, Pennsylvania State Univ*, 1999, pp. 16–18.

[2] W. S. Wong and R. W. Brockett, "Systems with finite communication bandwidth constraints. ii. stabilization with limited information feedback," *IEEE Trans. Autom. Control*, vol. 44, no. 5, pp. 1049–1053, May 1999.

[3] G. Nair, F. Fagnani, S. Zampieri, and R. Evans, "Feedback control under data rate constraints: An overview," *Proc. IEEE*, vol. 95, no. 1, pp. 108–137, Jan. 2007.

[4] K. You, N. Xiao, and L. Xie, *Analysis and Design of Networked Control Systems*. Springer, 2015.

[5] D. Liberzon and R. W. Brockett, "Nonlinear feedback systems perturbed by noise: steady-state probability distributions and optimal control," *IEEE Trans. Autom. Control*, vol. 45, no. 6, pp. 1116–1130, Jun. 2000.

[6] T. M. Cover and J. A. Thomas, *Elements of information theory*, 2nd ed. Hoboken, NJ: Wiley-Interscience, 2006.

[7] S. Tatikonda and S. Mitter, "Control under communication constraints," *IEEE Trans. Autom. Control*, vol. 49, no. 7, pp. 1056–1068, Jul. 2004.

[8] G. Nair and R. Evans, "Stabilizability of stochastic linear systems with finite feedback data rates," *SIAM J. Control Optim.*, vol. 43, no. 2, pp. 413–436, 2004.

[9] R. L. Adler, A. G. Konheim, and M. H. McAndrew, "Topological entropy," *Trans. Amer. Math. Soc.*, vol. 114, no. 2, pp. 309–319, 1965.

[10] L.-S. Young, "Entropy in dynamical systems," in *Entropy*. Princeton University Press Princeton, NJ, 2003, pp. 313–327.

[11] K. Butt. (2014) An introduction to topological entropy. University of Chicago. [Online]. Available: https://math.uchicago.edu/ may/REU2014/REUPapers/Butt.pdf

[12] G. Nair, R. Evans, I. Mareels, and W. Moran, "Topological feedback entropy and nonlinear stabilization," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1585–1597, Sept. 2004.

[13] F. Colonius and C. Kawan, "Invariance entropy for control systems," *SIAM J. Control Optim.*, vol. 48, no. 3, pp. 1701–1721, 2009.

[14] C. Kawan, "Invariance entropy for deterministic control systems," *Lecture Notes in Mathematics*, vol. 2089, 2013.

[15] F. Colonius, "Minimal bit rates and entropy for exponential stabilization," *SIAM J. Control Optim.*, vol. 50, no. 5, pp. 2988–3010, 2012.

[16] F. Colonius, C. Kawan, and G. Nair, "A note on topological feedback entropy and invariance entropy," *Systems & Control Letters*, vol. 62, no. 5, pp. 377 – 381, 2013.

[17] M. Fu and L. Xie, "The sector bound approach to quantized feedback control," *IEEE Trans. Autom. Control*, vol. 50, no. 11, pp. 1698–1711, Nov. 2005.

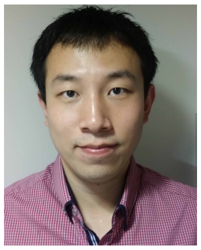
[18] K. You and L. Xie, "Network topology and communication data rate for consensusability of discrete-time multi-agent systems," *IEEE Trans. Autom. Control*, vol. 56, no. 10, pp. 2262–2275, Oct. 2011.

[19] B. Peng, "The determinant: a means to calculate volume," *Recall*, vol. 21, pp. 1–6, 2007.

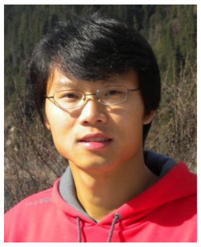
[20] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[21] J. Huang, *Nonlinear output regulation: theory and applications*, ser. Advances in Design and Control. SIAM: Society for Industrial and Applied Mathematics, 2004.

[22] D. A. Dowler, "Bounding the norm of matrix powers," Master's thesis, Brigham Young University, United States, 2013.



Yirui Cong (S'14–M'18) received the B.E. degree (outstanding graduates) in automation from Northeastern University, Shenyang, China, in 2011, the M.Sc. degree (graduated in advance) in control science and engineering from National University of Defense Technology, Changsha, China, 2013, and the Ph.D. degree from Australian National University, Canberra, Australia, in 2018. He is currently a lecturer at National University of Defense Technology, Changsha, China. His research interests are in the fields of communication theory and control theory.



Xiangyun Zhou (M'11–SM'17) is an Associate Professor at the Australian National University (ANU). He received the Ph.D. degree from ANU in 2010. His research interests are in the general fields of communication theory and networks. He currently serves on the editorial board of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE WIRELESS COMMUNICATIONS LETTERS. He was the chair of the ACT Chapter of the IEEE Communications Society and Signal Processing Society from 2013 to 2014.

Rodney A. Kennedy (S'86–M'88–SM'01–F'05) received the B.E. degree (1st class honours and university medal) from the University of New South Wales, Sydney, Australia, the M.E. degree from the University of Newcastle, and the Ph.D. degree from the Australian National University, Canberra. Since 2000 he has been is a Professor (now Emeritus Professor) in engineering at the Australian National University, Canberra, Australia.

He has co-authored over 300 refereed journal or conference papers and a book "Hilbert Space Methods in Signal Processing" (Cambridge Univ. Press, 2013). He has been a Chief Investigator in a number of Australian Research Council Discovery and Linkage Projects. His research interests include digital signal processing, digital and wireless communications, and acoustical signal processing.