On LQR Control with Asynchronous Clocks

Rahul Singh and Vijay Gupta

Abstract—We consider LQR control for a scalar system when the sensor, controller, and actuator all have their own clocks that may drift apart from each other. We consider both an affine and a quadratic clock model. For a quadratic cost function, we analyze the loss of performance incurred as a function of how asynchronous the clocks are. This also allows us to obtain guidelines on how often to utilize the communication resources to synchronize the clocks.

I. INTRODUCTION

Synchronicity is a basic assumption in most control literature. As networked and embedded control systems become more popular, this assumption may no longer hold. Clocks on different microprocessors seldom agree with each other. If sensors, controllers and actuators are time driven, this difference may significantly impact performance, or even lead to instability. In both distributed systems studied in computer science, and in communication networks, asynchronous clocks have long been identified as a major research issue.

Much of the work in distributed processing literature on asynchronous systems has focused on partial ordering of events (see, e.g., [15]). In control, such solutions may not be enough since precise timing is usually required [8]. Event driven paradigms in control such as petri nets (e.g., [21] and the references therein) explicitly include asynchrony. However, for time-driven systems, such modeling may not always be possible. For communication networks, protocols such as the network time protocol (NTP) have been developed to maintain clock synchronization. Similar protocols have also been developed using public transmitted signals, such as GPS. However, such solutions are expensive, incur communication overheads especially in distributed systems, and can only maintain limited accuracy (e.g., up to 10 ms mean accuracy for GPS based clocks as reported in [17]), that may not be enough for complex dynamical systems.

In works such as [4], [20], it was experimentally demonstrated that a higher precision in time-synchrony of GPS receiver and satellite clocks leads to better vertical estimates at the receiver. Control performance can be similarly expected to improve with better accuracy in clock synchronisation. Complementarily, the work in [18] demonstrated that asynchrony can lead to instability of an otherwise stable system. That work considered an affine model of the clock in which the time displayed by the clock reads as

\[ \tau(t) = at + b, \]  

where \( t \) can be the time read from any other clock, and need not be assumed to be some ‘true’ time. The parameter \( a \) is due to the frequency mismatch of the two clocks, and \( b \) is due to the initial phase offset. The work in [18] also showed that for such affine clocks and with rational \( a \), the system evolves as a periodic system, and in general, there is no explicit criterion for checking the asymptotic stability other than computing the spectral radius over one period. Similarly, the work in [16] showed that some basic control results fall apart when time cannot be perfectly measured. Stability of asynchronous systems is not trivial and interesting problems remain [1], [14], [26] proves a stronger version of the necessity part of the classical Chazan-Miranker theorem. The work in [11] introduces a Lyapunov-based theory for asynchronous dynamical systems and LMI and BMI formulations to construct Lyapunov functions and controllers for asynchronous systems. The work in [7] studies passivity properties of asynchronously non-uniformly sampled systems, and uses the concept of Maximum Sampling time preserving Passivity (MSP) to design controllers for feedback systems that are interconnected via time-varying and asynchronous sampling. Recent advances in the field include the works [8], [9], which studied the fundamental limits on synchronization of networked affine clocks and showed that even for the most favorable case of noiseless sampling, it is impossible to synchronize affine clocks precisely. Many clock synchronization algorithms have been proposed in the robotics literature, e.g., [10], [25], [5], [27]. However, an analysis of how much synchronization is required to guarantee specified control performance is still missing. We should also mention the works on multi-rate sampling, which considers the related problem when different components of the control loop are sampled at different (although usually both constant and known) rates, possibly not at the same time [24], [3], [12] (see also related works in [13], [2]).

Inspite of these significant works, a full understanding of the impact of asynchrony on performance achievable from a control loop is lacking. In particular, since synchronization consumes system resources that may be otherwise utilized for control, the frequency and accuracy of clock synchronization needs to be adjusted to optimize the overall system performance. In this paper, we analytically characterize the performance loss that asynchrony may induce for a basic LQR problem. We also consider the question of how accurately do the clock parameters need to be known to limit the performance degradation to a desired level. For a given synchronization algorithm, such relations may provide guidelines on how much resources to devote to synchronization, as opposed to controlling the system with asynchronous clocks.

The paper is organized as follows. We begin by formu-
lating the problem and stating our assumptions. In Section
III-A, we characterize the performance degradation due to asynchrony for the affine clock model. Section III-
B introduces the notion of “asynchronous sequence”, and
shows that infinite horizon performance for certain class of
systems continuously degrades with increasing asynchrony.
In section IV, we characterize the performance degradation
for the infinite horizon LQR cost for various clock models.

II. PROBLEM FORMULATION

Process Model and Cost: Consider a discrete time pro-
cess evolving as

\[ x_{k+1} = Ax_k + Bu_k, \quad k \geq 0, \]  

(2)

where the state \( x_k \in \mathbb{R} \), the control input \( u_k \in \mathbb{R} \), and \( B \neq 0 \).
The asynchronous case, the sensor measures and transmits
\( x_k \) at times \( kT_s \), where \( k \geq 0 \) and \( T_s \) is a known sampling
time. The controller calculates the control input at times
\( kT_s \) using the latest value of the state. Finally, the actuator
applies this control value at times \( kT_s \) to enable the
process to evolve according to (2). In the asynchronous case,
the control \( u_k \) is generated by a remote controller whose
clock \( C_c \) may differ from the clock \( C_s \) that is shared by the
sensor and the actuator. The sensor and the actuator act at
times given by \( kT_s \) according to \( C_p \), while the controller
updates its inputs at times \( kT_s \) according to \( C_c \). We use
the term “cycle” to denote the intervals \( (kT_s, (k+1)T_s) \) for
either clock. Both the controller and actuator are modeled as
maintaining a buffer of unit length that is used to store the
latest measurement that was received from the sensor, or the
latest control input received from the controller, respectively.
Thus, e.g., if the sensor generates two measurements in
one cycle of the controller’s clock, then only the second
measurement is stored and used by the controller, and the first
measurement is deleted. Once a buffer input is used by the
controller or the actuator, it is deleted from the corresponding
buffer. An empty buffer when queried returns the value 0,
which is indistinguishable from an actual measurement or
control with value 0. We assume that the controller is not
able to change the clock of process, or its own clock. The
control input is calculated to minimize the cost

\[ J_{\infty}(x_0) = \sum_{k=0}^{\infty} (x_k^TQ + u_k^TR). \]  

(3)

Clock Model: We consider two clock models in this work.
- Affine Clocks: The simplest model of clocks (e.g., [9],
[18]) is an affine clock, in which each clock is described by (1). We will write the sensor (or actuator) clock in
terms of the controller clock, so that the relation between
the two clocks is given by

\[ t_{sensor} = at_{controller} + b. \]  

(4)

The parameter \( a \) is time-invariant and is called the skew
or relative frequency. The skew \( a \) may be lesser than or
greater than 1 (corresponding to a slower or a faster sensor
clock), but is positive. In this work, we assume that \( a \)
is rational. The parameter \( b \) is called the phase offset or
the initial phase. For \( a < \frac{1}{2} \), the controller calculates two
control inputs per system transition. Thus, in keeping with
the assumptions stated earlier, for the second control input,
since no new sensor measurement has been received, the
control input calculated is 0. Thus, the process evolves
open-loop forever. Accordingly, we assume \( a > 0.5 \).
- Quadratic Clocks: In this model, the relation between the
two clocks is given by

\[ t_{sensor} = at_{controller}^2 + bt_{controller} + c, \]  

where \( c \) is the initial phase of the sensor clock with respect
to \( C_c \), while \( a \) and \( b \) are sensor clock parameters.

Although we focus on a quadratic clock model in this work,
similar arguments can be carried out for higher order models.
In general, the controller may not be aware of the exact order
of the clock. A reasonable policy in that case is to assume
that \( C_p \) is affine and try to estimate its parameters periodically
via exchange of time stamps.

Notation: We define \( \{l, r \} = \{x | x \in \mathbb{Z}^+, r \geq x \geq l \} \).
The set of positive integers is denoted by \( \mathbb{Z}^+ \) and the set of
reals by \( \mathbb{R} \).

III. PRELIMINARY RESULTS

A. Performance with affine clocks

We begin by characterizing the performance degradation
that is suffered when the clocks are affine and the con-
troller has been designed assuming synchronous operation.
The skew and the phase offset degrade the performance in
different ways. If the phase offset \( b \neq 0 \), then the process
and the controller begin at different times. In particular, if
\( b > 0 \), then the process begins before the controller. In this
case, if we express \( b = (l + b^*)T_s \), where \( l \in \mathbb{Z}^+ \) and
\( 0 \leq b^* < 1 \), the process evolves without any control input for
\( l + 1 \) steps. Similarly, if \( b < 0 \), then the process begins after
the controller. In this case, if we express \( b = a(l^* + b^*)T_s \),
where \( -l^* \in \mathbb{N} \) and \( 0 < b^* < 1 \), then the controller does not
apply any control in its first \( l^* \) time steps. On the other hand,
if the skew parameter \( a \neq 1 \), then the number of control
inputs generated is different from the number of control
inputs applied to the process. In particular, if \( a > 1 \), then
there will be transitions of the process where the actuator
finds an empty buffer and applies the control input zero.
On the other hand, if \( a < 1 \), then the controller will send
more than one control input during some cycles according
to \( C_p \). For pedagogical ease, we will assume that if \( a > 1 \), it
may be written as \( a = \frac{N_a}{N_a + 1} \) for an appropriate \( N_a \in \mathbb{N} \).
Similarly if \( a < 1 \), we will assume that \( a \) can be written as
\( a = \frac{N_a}{N_a + 1} \) for an appropriate \( N_a \in \mathbb{N} \). Thus, the sensor clock
will either gain (\( a > 1 \)) or lose (\( a < 1 \)) a time equal to \( T_s \)
after every \( N_a \) steps according to \( C_p \). The general case of \( a \)
being any other rational number is conceptually similar, but
notationally more difficult. Combining the above arguments,
we obtain the following result.

Theorem 3.1: Consider the process (2) with the associated
cost function (3), the sensor clock modeled as (4), and where
the controller is designed assuming perfect synchronization. The resulting cost can be expressed as \( x^2(0)\tilde{P}_0 \), where \( \tilde{P}_0 \) can be computed as a solution of a Lyapunov equation.

**B. Asynchronous Sequences**

Theorem 3.1 is not very convenient to answer questions such as does the performance loss increase with the asynchrony level. Since increasing delay can, e.g., improve transient performance, the effect of increasing levels of asynchrony are not trivial. We now develop an alternate technique to this end. We assume that the control law \( F \) is given and make the following additional assumption:

**Assumption** \( A_1 \): \( A_c^2 \leq A^2 \), where \( A_c \equiv A + BF \).

Note that, unless otherwise stated, we make no assumption about the clock model.

Asynchrony degrades performance since it leads to the process evolving in open loop at some instances. If we consider the clocks to evolve from the time when \( C_p \) reads 0, then all instances when \( C_p \) completes its one cycle without any occurrence of \( C_c \) completing its cycle, or vice versa, are instances at which the process evolves open-loop. We collect the times at which such transitions occur, as displayed by \( C_p \), in a sequence that we term an asynchronous sequence.

**Definition 3.1:** Consider a sequence \( S = \{S_k\}_{k=1}^N \) where \( S_k \) denotes the \( k \)-th element of the sequence, and \( S_k < S_{k+1} \). For the process in (2) and a given control law \( F \), let the system evolve as

\[
x_{k+1} = \begin{cases} (A + BF)x_k & k \in \mathbb{N} \setminus S \\ Ax_k & k \in S. \end{cases}
\]

If the number of steps in which process evolves without a control input is finite (say \( N - 1 \)), then let \( S_N = \infty \). Then, \( S \) is an asynchronous sequence corresponding to the process (2).

To compare two clocks, we will compare asynchronous sequences that arise due to these clocks for the same process. To this end, use the controller clock as the reference and write the sensor clocks in terms of the controller clock. For the \( i \)-th sensor clock, denote \( S^i = \{S^i_k\} \) to be the corresponding asynchronous sequence.

**Definition 3.2:** We define two asynchronous sequences \( S^1 \) and \( S^2 \). We denote \( S^1 \leq S^2 \) if \( S^1_k \leq S^2_k \), \( \forall k \in \mathbb{N} \). Moreover, we say that \( S^1 < S^2 \) if \( S^1 \leq S^2 \) and \( S^1_k < S^2_k \), for at least one \( k \in \mathbb{N} \).

For two asynchronous sequences \( S^1 \) and \( S^2 \), \( S^1 \leq S^2 \) implies that the process evolves open loop for at least the same number of time steps with sequence \( S^2 \) as with \( S^1 \).

**Definition 3.3:** Given two asynchronous sequences \( S^1 \) and \( S^2 \), we say that \( S^1 \) and \( S^2 \) are alternating sequences with \( S^1 \leq S^2 \) if \( \forall k \in \mathbb{N} \), \( S^1_k \leq S^2_k \leq S^1_{k+1} \). If, in addition, \( S^1_k < S^2_k < S^1_{k+1} \) for at least one \( k \), then we say that \( S^1 \) and \( S^2 \) alternate with \( S^1 < S^2 \).

Consider two sequences \( S^1 < S^2 \) such that \( S^1 \) and \( S^2 \) alternate. Let \( N \) be the cardinality of \( S^1 \) (possibly \( N \to \infty \)). Then, for any \( k < N \), one and only one of the following must be true:

- \( S^1_{k+1} > S^2_k \)
- \( S^1_{k+1} = S^2_k \)
- \( S^1_{k+1} < S^2_k \)

\( S^2_p = S^1_{p+1} \), \( k \leq p < k + l \), \( l \leq L \) for some \( L \in \mathbb{N} \)

\( S^1_{k+1} - 1 \geq S^2_k + 1 \)

**Definition 3.4:** Consider two given sequences \( S^1 < S^2 \) such that \( S^1 \) and \( S^2 \) alternate. We construct three types of sets in the following manner.

1) Initialize with \( k = 1 \) and \( i_\gamma = i_\lambda = i_\psi = 0 \).

2) If \( S^1_{k+1} > S^2_k \), then

- Define \( \Gamma^\gamma_{i+1} = \{S^1_k, S^2_k\} \).
- Increment \( i_\gamma \) and \( k \) by 1, i.e. \( i_\gamma \longrightarrow i_\gamma + 1 \), and \( k \longrightarrow k + 1 \).

else if \( S^2_p = S^1_{p+1} \), \( k \leq p < k + l \), \( l \leq L \) for some \( L \in \mathbb{N} \), then

- Set \( \Lambda^\lambda_{i+1} = \{S^1_k, S^2_{k+l}\} \).
- Increment \( i_\lambda \) and \( k \) by 1, i.e. \( i_\lambda \longrightarrow i_\lambda + 1 \), and \( k \longrightarrow k + 1 \).

else if \( S^1_{k+1} - 1 \geq S^2_k + 1 \), then

- Define \( \Psi^\psi_{i+1} = \{S^2_k, S^1_{k+1}\} \).
- Increment \( i_\psi \) and \( k \) by 1, i.e. \( i_\psi \longrightarrow i_\psi + 1 \), and \( k \longrightarrow k + 1 \).

3) If \( k = N \) then terminate, else repeat step 2.

These indexed sets may be empty, have finite cardinality, or infinite cardinality. Let there be \( \lambda \) number of indexed sets \( \Lambda^\lambda \), \( \gamma \) of \( \Gamma^\gamma \), and \( \psi \) number of \( \Psi^\psi \) sets for a given pair of alternating sequences \( S^1 < S^2 \). Then, we define the following ‘index-sets’:

- \( C^\lambda = \{1, 2, \ldots, \lambda\} \).
- \( C^\gamma = \{1, 2, \ldots, \gamma\} \).
- \( C^\psi = \{1, 2, \ldots, \psi\} \).

Note that if the process evolves open-loop for finite number of times in the systems, then the cardinality of sets \( C^\lambda \), \( C^\gamma \), and \( C^\psi \) would be finite and the cardinality of one of the sets \( \Lambda^\lambda \), \( \Gamma^\gamma \), or \( \Psi^\psi \) would be infinity.

For the process (2) that evolves with the control sequences that are determined by the control law \( F \) and the sensor clock \( i \), denote the control input applied at time \( k \) by \( u^i_k \) and the state value at time \( k \) by \( x^i_k \). With a slight abuse of terminology, we will denote the system as it evolves with the \( i \)-th sensor clock by system \( i \). Define the following terms for the system \( i \):

- Denote by \( p^i_{k} \) the state cost at time \( k \), \( p^i_{k} = (x^i_k)^2 Q \).
- Denote by \( p^i_{k} \) the control cost at time \( k \), \( p^i_{k} = (u^i_k)^2 R \).
- Denote by \( P^i_{k} \) the control cost upto time \( k \), \( P^i_{k} = \sum_{\kappa=0}^{k} p^i_{\kappa} \).
- Denote by \( P^i_{k} \) the cost control cost upto time \( k \), \( P^i_{k} = \sum_{\kappa=0}^{k} p^i_{\kappa} \).
- Denote by \( P^i_{k} \) the total cost up to time \( k \), \( P^i_{k} = P^i_{k} + P^i_{k} \).
- Denote by \( P^i_{k} \) the infinite horizon cost \( P^i_{k} = \lim_{N \to \infty} P^i_{k} \).

Finally, denote the total control cost incurred in steps from \( N_1 \) to \( N_2 \) as \( P^i_{k}(N_2, N_1) = \sum_{k=N_1}^{N_2} P^i_{k} \).

**Lemma 3.2:** Consider the process (2) with the associated cost function (3), the sensor clock modeled as (4), and where the controller is designed assuming perfect synchronization. Let the process evolve with two different clocks from the
same initial condition. Let the asynchronous sequences for the two clocks be denoted by \( S^1 \) and \( S^2 \) respectively, and let the control and state values at time \( k \) for the two cases be denoted by \( (u_{1}^{k}, x_{1}^{k}) \) and \( (u_{2}^{k}, x_{2}^{k}) \) respectively. Denote by \( \Phi_{i}(k) \) the transition matrix for system \( i \) till time \( k \), so that \( x_{i}^{k} = \Phi_{i}(k)x_{0}, i = 1, 2. \) Let \( S^1 \) and \( S^2 \) alternate with \( S^1 < S^2 \). Finally, define for all \( L \geq 1, \Lambda^* = [S^1_L, S^2_{L+1}], \) and \( S^1_{k+p} - S^2_{k+p} = l_{p+1}, \forall 1 \leq p \leq L. \) Then the following statements are true:

- **Claim B1:** \( \Phi_{1}^2(i) \geq \Phi_{2}^2(i), \forall i \in Z^+ \).
- **Claim B2:** \( P_{1}(\Gamma^i) \geq P_{2}(\Gamma^i), \forall i \in C_{\gamma} \).
- **Claim B3:** \( \Lambda^*(k) = A^2 \Phi_{1}^2(S_{k+1}^1) \geq A^2 \Phi_{2}^2(S_{k+1}^2), \forall L \geq 1 \).
- **Claim B4:** \( A^{2p+1} \Phi_{1}^2(S_{k+1}^1) \geq A^{2p} \Phi_{2}^2(S_{k+1}^2), \forall p \) and \( q \) such that \( t_p - 1 \geq q \geq 0, L \geq p \geq 1 \).
- **Claim B5:** \( P_{1}(\Lambda^i) > P_{2}(\Lambda^i), \forall i \in C_{\Lambda} \).
- **Claim B6:** \( P_{1}(\Psi^i) \geq P_{2}(\Psi^i), \forall i \in C_P \).

**Proof:** Omitted for space constraints.

We begin with the case when the sequences are alternating.

**Lemma 3.3:** Consider the process (2) with the associated cost function (3), the sensor clock modeled as (4), and the controller is designed assuming perfect synchronization. Let the process evolve with two different clocks from the same initial condition. Let the asynchronous sequences for the two clocks be denoted by \( S^1 \) and \( S^2 \) respectively, and let the control and state values at time \( k \) for the two cases be denoted by \( (u_{1}^{k}, x_{1}^{k}) \) and \( (u_{2}^{k}, x_{2}^{k}) \) respectively. If \( S^1 \) and \( S^2 \) alternate with \( S^1 < S^2 \), then the following are true:

1. \( p^{1}_{1}s \geq p^{2}_{1}s, \forall k > 0. \)
2. \( p^{1}_{\infty} \geq p^{2}_{\infty}, \forall N > 0. \) In particular, \( P^{\infty}_1 \geq P^{\infty}_2. \)
3. \( P^{\infty}_1 > P^{\infty}_2. \)
4. Assume that the process is stable under case 1. Then \( P^{\infty}_1 \geq P^{\infty}_2. \)

**Proof:**

1. Using B1, we obtain
\[
p^{k}_{1}s = (x_{1}^{k})^2 = x_{1}^{k} \Phi_{1}(k)Q \geq x_{1}^{k} \Phi_{2}(k)Q = (x_{2}^{k})^2 = p^{k}_{2}s. \quad (7)
\]
2. From the first part, \( P^{\infty}_1 = \sum_{k=0}^{\infty}p^{k}_{1}s \geq \sum_{k=0}^{\infty}p^{k}_{2}s = \sum_{k=0}^{\infty}p^{k}_{2}s. \) Since the inequality holds for every \( N \), taking limits as \( N \to \infty \), we obtain \( P^{\infty}_1 \geq P^{\infty}_2. \)
3. Using B2, B3, and B6, we have
\[
P^{\infty}_1 = \sum_{k \in C_{\Lambda}} P_{1}(\Lambda^k) + \sum_{k \in C_{\gamma}} P_{1}(\Gamma^k) + \sum_{k \in C_{\Psi}} P_{1}(\Psi^k) \\
\leq \sum_{k \in C_{\Lambda}} P_{2}(\Lambda^k) + \sum_{k \in C_{\gamma}} P_{2}(\Gamma^k) + \sum_{k \in C_{\Psi}} P_{2}(\Psi^k) = P^{\infty}_2.
\]
4. Using the second and third part of this lemma, we obtain \( P^{\infty}_1 = P^{\infty}_1 + P^{\infty}_1 \geq P^{\infty}_2 + P^{\infty}_2 = P^{\infty}_2. \)

**Remark 1:** Although we have concentrated on the infinite horizon cost, similar arguments can be made for finite horizon costs, as long as the horizon is long enough. In fact, given the sequences \( S^1 \) and \( S^2 \), one can determine the value \( N^* \) such that \( \forall N > N^*, N \in N, P^{1}_{N} > P^{2}_{N} \) holds.

We can now compare the performance of the process with any two asynchronous sequences that can be ordered, even if they are not alternating. If \( S^1 < S^2 \), then the sequences must belong to exactly one of the following cases:

- Type T1: \( S^1, S^2 \) alternate with \( S^1 < S^2 \).
- Type T2: The sequences are not alternating, but there is at least one element of \( S^1 \) between any two consecutive elements of \( S^2 \), i.e., \( \exists \) at least one \( j \geq k \) such that \( S^1_k \leq S^1_k \leq S^2_{k+1}, \forall k \in N. \)
- Type T3: There is at least one \( k \in N \) such that \( S^1_k \leq S^2_k \leq S^2_{k+1} < S^1_{k+1} \) with \( j \geq k + 1. \)

For sequences of type T2, consider the following algorithm to generate an additional sequence \( V \).

**Algorithm 3.1:**

1. **Initialize** with \( k = 1. \)
2. **Let** \( H = \{S^1_1, S^1_2, S^1_3, \ldots, S^1_{j+p}\} \), where \( S^1_k \leq S^1_j < S^1_{j+p} \leq S^1_{k+1}. \)
3. **Set** \( V_k = S^1_j \) and \( k = j + 1. \)

For sequences of type T3, consider the following algorithm to generate an additional sequence \( V \).

**Algorithm 3.2:**

1. **Initialize** \( V = S^1. \)
2. **Pick** the smallest \( k \) for which \( V_j < S^2_k, j > k \) holds.
3. **Set** \( W_i = V_i, \forall i < J. \)
4. **Identify** the set \( B = \{S^1_l, S^2_{l+1}\}, B \subset N, \) where \( l \) is the smallest natural satisfying \( V_i < S^2_l \leq S^2_{l+1} < V_{i+1} \), for some \( i \in N. \) Set \( i = i \). **Pick** any \( b \in B. \)
5. **Set** \( W_i = V_{i+1}, \forall J \leq i < I. \) **Set** \( W_{I} = b. \)
6. **Set** \( W_i = V_i, \forall i > I. \)
7. **Set** \( V = W, W = \phi \) (the empty set).
8. **Repeat** above steps till \( V \) and \( S^2 \) are related with each other in type \( T_2. \)

**Lemma 3.5:** Algorithm 3.2 guarantees that \( V \) and \( S_2 \) are of Type \( T_2 \) when \( V < S^2, \) and \( S_1 \) and \( V \) alternate with \( S^1 < V. \)

**Proof:** The proof follows simply by construction since after each iteration \( V \) is alternating with respect to the output of the previous iteration.

**Remark 2:** Algorithm 3.2 is illustrated with an example. Let \( S^1 = \{2, 3, \infty\} \) and \( S^2 = \{4, 5, \infty\} \), so that they belong to \( T_3. \) In the first iteration of the algorithm, we set \( V = S^1, K = 1, J = 2. \) Thus, \( W_1 = V_1 = 2 \) leading to \( B = \{4, 5\}. \) This in turn implies that \( I = 2. \) We choose \( b = 4. \) Thus, \( W_2 = 4 \) (case of \( I = J \)) and \( W_3 = \infty. \) At the end of the iteration, we set \( V = W. \)

Hence we obtain \( V = \{2, 4\} \) which clearly is of type \( T_2 \) with respect to \( S^2. \) Also note that in each iteration, the algorithm generates a sequence which is alternating with respect to the previous output of iteration. Hence \( P^{\infty}_1 \geq P_{V}. \)

**Theorem 3.6:** Consider the process (2) with the associated cost function (3), the sensor clock modeled as (4), and where the controller is designed assuming perfect synchronization.
Let the process evolve with two different clocks from the same initial condition. Let the asynchronous sequences for the two clocks be denoted by $S^1$ and $S^2$ respectively. If $S^1 < S^2$, then $P^1_{\infty} \geq P^2_{\infty}$.

*Proof:* If $S^1 < S^2$, then the sequences must belong to exactly one of the following cases:

- Type $T_1$: In this case, application of Lemma 3.3 directly yields $P^1_{\infty} \geq P^2_{\infty}$.
- Type $T_2$: In this case, we can generate a sequence $\tilde{V}$ using Algorithm 3.1 such that $P^1_{\infty} \geq P^2_{\infty}$.
- Type $T_3$: In this case, we generate a sequence $V$ using Algorithm 3.2. Since $V$ and $S^2$ are of Type $T_2$, $P^2_{\infty} \geq P^2_{\infty}$. Since $V$ and $S^1$ alternate with $S^1 < V$, $P^1_{\infty} \geq P^2_{\infty}$. Combining the two, we obtain $P^1_{\infty} \geq P^2_{\infty}$.

In the next section, we apply the concept of asynchronous sequences to compare the performance of systems with different asynchronous clocks.

**IV. Performance Comparison**

**A. Affine Clocks**

In this section, we focus on affine clocks of the form (4). With a rational skew $a$, the process evolves as a periodic system. Thus, e.g., if $a = \frac{N_a}{N_a - 1}$ (and $b = 0$), the system evolves as

$$x_{k+1} = \begin{cases} Ax_k & k = lN_a - 1, \ l \in \mathbb{N} \\ A_c x_k & \text{otherwise.} \end{cases}$$

Stability conditions of periodic systems cannot usually be written in terms of individual matrices separately. However, in our case, we have the following result.

*Theorem 4.1:* Consider a process of the form (2) with an LQR control law designed assuming synchrony and where the sensor clock is affine modeled as in (4) with rational $a$. Then the stability of the system depends only on the value of the skew $a$, and is independent of any finite phase parameter $b$. Moreover, if the process is stable with skew $a_1$, then it remains stable with skew $a_2$ if either $a_1 \geq a_2 \geq 1$ or $a_1 \leq a_2 \leq 1$.

*Proof:* Omitted for space constraints.

While the above result implies that the stability of the process does not depend on the value of $b$, the performance does depend on the parameter. To compare the performance with two different clocks, we proceed as follows. We introduce the following notation

$$b = \begin{cases} -a T_s (\gamma - \varphi), \ \gamma \in \mathbb{N}, \ 0 < \varphi < 1, & \text{if } b < 0, \\ T_s (l + b^*), \ l \in \mathbb{Z}^+, \ 0 < b^* < 1, & \text{if } b \geq 0. \end{cases}$$

When $b \geq 0$, for all $l \in \mathbb{Z}^+$, define the sets $b^l = \{lT_s, (l+1)T_s\}$. Finally, denote by $P^\infty(a, b)$, the infinite horizon LQR cost achieved with a system in which the sensor clock has skew $a$ and initial phase $b$.

*Theorem 4.2:* Consider two systems with affine clocks of the form (4), with rational skews $a_1$ and $a_2$, and initial phases $b_1$ and $b_2$, respectively. Let $b_l = -a_i T_s (\gamma_l - \varphi_i)$ if $b_l < 0$ and $b_l = T_s (l_i + b^*_l)$ if $b_l \geq 0$. Then, the following hold true:

1. Result $R_1$: $P^\infty(a_1, b) > P^\infty(a_2, b)$, $\forall b$, if either $a_1 > a_2 \geq 1$, or $a_1 < a_2 \leq 1$.
2. Result $R_2$: $P^\infty(a, b_1) \geq P^\infty(a, b_2)$, if
   a. $(a > 1)$ and $((b_1 > b_2 \geq 0) \text{ or } (b_1, b_2 < 0 \text{ and } \varphi_1 > \varphi_2))$. Also, the relation holds with equality if $\varphi_1 = \varphi_2$.
   b. $(a < 1)$ and $((b_1, b_2 > 0 \text{ and } l_1 > l_2) \text{ or } (b_1, b_2 > 0 \text{ and } b_1^2 < b_2^2 \text{ and } b_1^2 < b_2^2 \text{ for some } l \in \mathbb{N})$ or (if $(b_1, b_2 < 0, \varphi_1 < \varphi_2)$).
3. Result $R_3$: When $\varphi_1 = \varphi_2 = 0$, then $P^\infty(a, b_1) = P^\infty(a, b_2) = P^\infty(a, 0)$.

*Proof:* Omitted for space constraints.

**B. Performance Bounds Under Uncertainty**

Various synchronization algorithms may yield not the exact values of the parameters $a$ and $b$, but rather a range of values for them. It may even be the case that the ranges progressively decrease as more communication resources are spent on synchronization. It is thus of interest to find conditions under which knowing the parameter within a given range can yield acceptable control performance, and thus, further synchronization need not be performed.

We begin when the clocks are affine in reality and the controller knows that the true values of the parameters $a$ and $b$ satisfy $a_1^* \leq a \leq a_2^*$ and $b_1^* \leq b \leq b_2^*$ respectively. The controller can correct for a specific skew $\hat{a}$ and phase $\hat{b}$ in this range. The performance of the system is then identical to the case when the controller assumes the skew to be unity and the phase parameter to be zero, while the true skew and phase lie in the sets $[a_1^*, a_2^*]$ and $[b_1^*, b_2^*]$ respectively. Thus, the characterization of how much accuracy is needed in knowing the parameters of the clocks, and the decision of which parameter values the controller should correct for, can be answered by characterizing the worst performance realized by the system when the control law is designed assuming synchrony, and the true values of the clock parameters $a$ and $b$ lie in the uncertainty set $U$, i.e., $U = \{a, b\} \in [a_1, a_2]$, $b \in [b_1, b_2]$). Accordingly, denote by $P^\infty(a, b)$ the infinite horizon cost realized when the clock skew is $a$ and the initial phase offset is $b$, which can be computed using Lemma 3.1. We denote the upper-bound on performance associated with an uncertainty set $U$ by $P^\infty(U)$, i.e.,

$$P^\infty(U) = \max_{a \in [a_1, a_2], b \in [b_1, b_2]} P^\infty(a, b).$$

*Theorem 4.3:* Consider a process of the form (2) with an affine clock with skew $a$ and offset $b$ that satisfy $a_1 < a < a_2$ and $b_1 < b < b_2$, respectively.

- If $a_1 > 1$, then $P^\infty(U) = P^\infty(a_2, b_2)$.
- If $a_2 < 1$ then

$$P^\infty(U) = \begin{cases} P^\infty(a_1, b_1) & b_1, b_2 \in b^l, l \in \mathbb{N} \\ P^\infty(a_1, l_2 T_s) & b_1, b_2 > 0 \text{ and } l_1 \neq l_2 \\ P^\infty(a_1, -a_2 T_s) & -a_1 T_s \geq b_1, b_2 < 0 \\ P^\infty(a_1, b_1) & b_1 \geq -a_1 T_s, b_1, b_2 < 0 \end{cases}$$
If \( a_2 \geq 1 \geq a_1 \), then
\[
P^\infty (U) = \begin{cases} 
\max \{ P^\infty(a_2, b_2), P^\infty(a_1, b_1) \} & \text{if } (b_1, b_2) \in b^l, l \in \mathbb{N} \text{ or } (b_1 \geq -a_1 T_s, b_1, b_2 < 0) \\
\max \{ P^\infty(a_2, b_2), P^\infty(a_1, l_2 T_s) \} & \text{if } b_1, b_2 > 0 \text{ and } l_1 \neq l_2 \\
\max \{ P^\infty(a_2, b_2), P^\infty(a_1, -a_2 T_s) \} & \text{if } -a_1 T_s \geq b_1, \text{ and } b_1, b_2 < 0 
\end{cases}
\]

\textbf{Proof:} The proof follows directly from results \( R_1 \) and \( R_2 \) as applied for the three cases mentioned in the statement of the result.

Our next result considers the case when \( C_p \) is quadratic according to (5), while the controller assumes \( C_p \) to be affine with a time-varying skew. Thus, the controller estimates the parameters of an affine model of the sensor clock periodically every \( N \) steps (as measured according to \( C_p \)). Over the next \( N \) steps, the assumed affine model gradually diverges more and more from the actual clock model. Denote the estimate of sensor clock by \( \hat{C}(t) \). Denote the instantaneous skew of sensor clock, as measured by controller clock via exchange of time-stamp by \( \hat{a} \). We assume that the skew is estimated accurately. Denote the infinite horizon performance cost thus achieved by \( P^\infty \). As discussed in the case of affine clocks, when the controller corrects for a given model of an affine clock, the situation is the same as if the controller is designed assuming synchrony and the clocks drift apart. For simplicity we assume that for the clock model in (5), \( 1 < a < a_u \) and \( |c| < c_u \).

\textbf{Definition 4.1:} Consider a system in which the process
- evolves open-loop for \( c \) steps in the beginning
- and thereafter evolves as periodic process with a transition matrix \( \Phi(n, N) = A_n^{N-n} A^n \).

The performance of such a system is denoted as \( B(c, n, N) \).

\textbf{Theorem 4.4:} For the system described above, the performance cost is bounded as \( P^\infty < P(N) \), where
\[
P(N) = B \left( \frac{c u}{T_s}, a_u N^2 T_s, N \right)
\]
is increasing in \( N \).

\textbf{Proof:} Omitted for space constraints.

\textbf{Remark 3:} The above result states that synchronizing more often, i.e. smaller \( N \) would yield a lower bound on performance cost. Hence more synchronization would ensure maintaining some predefined performance criteria, denote by \( C \). An appropriate \( N^* \) can be found out so that \( P(N) < C \), \( \forall N < N^* \), and hence \( P^\infty < C \).

\section*{V. CONCLUSIONS}

We consider LQR control of a scalar system when the sensor, controller, and actuator all have their own clocks that may drift apart from each other. We consider both an affine and a quadratic clock model. For a quadratic cost function, we analyze the loss of performance incurred as a function of how asynchronous the clocks are. This also allows us to obtain guidelines on how often to utilize the communication resources to synchronize the clocks.

\begin{thebibliography}{99}


\bibitem{17} B. Li, C. Rizos, and H. K. Lee, “A GPS-slaved time synchronization system for hybrid navigation,” GPS Solutions 10: 207-217, June 2006.


\end{thebibliography}