On a Control Algorithm for Time-varying Processor Availability

V. Gupta Member, IEEE and F. Luo

Abstract—We propose an anytime algorithm for control when the processor resource availability is limited and time-varying. The basic idea behind the algorithm is to calculate the components of the control input vector sequentially to maximally utilize the processing resources available at every time step. For the LQR case, we present a Markovian jump linear system formulation that provides analytical performance and stability expressions. For more general cases, we present and analyze a receding horizon control based formulation. The improvement in performance is also illustrated through numerical simulations.

I. INTRODUCTION

We consider the problem of control in the presence of limited and time-varying availability of processing resources. This problem may arise, e.g., in embedded control systems, in which the micro-processor may be responsible for multiple functions. Similarly, in networked systems, one remote controller may control multiple processes leading to competition for the shared processor resources. As controllers are designed for systems to operate in more and more complex environments, the problem of control under limited processing resources is likely to remain relevant even if economic considerations permit the use of very powerful processors. It is, thus, imperative to design control algorithms that are able to guarantee stability and performance in spite of constrained and time-varying processing resources.

Owing to its importance, this problem has been considered in a growing body of work (see, e.g., [5], [6], [7]). Particular relevant is the work on event-triggered and self-triggered control (e.g., [9], [10]) in which a control input is calculated periodically leading to better processor utilization for the same performance (however, it is assumed that the control input can be calculated whenever required). In this paper, we consider a slightly different question: How do we guarantee good control performance, when the processor availability is time-varying and a priori unknown? Our approach is to develop anytime control algorithms. Anytime algorithms provide a solution even with limited processing resources, and refine the solution as more resources become available. Anytime algorithms can tolerate fluctuating processor availability, and are, thus, very popular in real-time systems. In control, however, there are few methods available for developing anytime controllers. Bhattacharya and Balas [1] focused on linear processes and controllers, and presented a controller that updated a different number of states depending on the available and a priori known computational time. Greco et al [3] proposed switching among a set of given controllers that may require different execution times but yield increasingly better performance. The analysis was presented by assuming a Markovian jump linear system, and hence, was limited to linear process and controllers.

We present an anytime control algorithm that calculates the control inputs sequentially with a time-varying computation priority for calculating the various components of the control vector. For an LQR formulation, we solve for the optimal control inputs and analyze the resulting stability and performance. We also extend the algorithm to constrained linear systems using a receding horizon control formulation.

The paper is organized as follows. We begin in Section II by formulating the problem. In Section III, we present the proposed algorithm for the LQR problem. We then consider the case of constrained linear systems in Section IV. Numerical results are presented in Section V.

II. PROBLEM FORMULATION

Process Model: Consider a process evolving as

\[ x(k+1) = Ax(k) + Bu(k), \quad x(0) \]

with the state \( x(k) \in \mathbb{R}^n \), the control input \( u(k) \in \mathbb{R}^m \), and the pair \( (A,B) \) assumed to be controllable. We consider two problem setups. In the unconstrained problem set up, the control inputs need to be calculated to minimize a cost function of the form

\[ J_{N_h} = \sum_{k=0}^{N_h} [x^T(k)Qx(k) + u^T(k)Ru(k)] \]

\[ + x^T(N_h + 1)P_{N_h+1}x(N_h + 1)], \]

for given positive definite matrices \( Q, R \) and \( P_{N_h+1} \), and a given horizon \( N_h \). In the constrained problem set up, the control inputs are again computed to minimize the cost (2); however, additionally, the state and control inputs have to satisfy constraints of the form \( h(x(k),u(k)) \in S \) for a given set \( S \). For simplicity, we assume full state feedback at the controller and \( R \) to be a diagonal matrix. Let \( R_{ij} \) denote the \((i,j)\)-th element of \( R \) and \( u_i(k) \) the \( i \)-th element of \( u(k) \). For concreteness, we adopt the following convention for the value of the control input utilized if the controller is unable to calculate a new control input at time \( k \).

Assumption 1: If no control input is available at time \( k \), then the actuator applies the control input \( u(k) = 0 \).

Processing Time Availability: Without loss of generality, we map the availability of any processing resource at time \( k \) to the

Preprint submitted to IEEE Transactions on Automatic Control. Received: July 15, 2012 07:39:57 PST
execution time \( \tau(k) \) that is available for the control calculation at time \( k \). We make the following assumptions.

**Assumption 2:** The execution time \( \tau(k) \) available at time \( k \) is an independent and identically distributed sequence, with a known probability mass function (see, e.g., [3]). The controller does not have a priori knowledge of \( \tau(k) \).

**Assumption 3:** The execution time required to calculate the control input vector is a non-decreasing function \( f(p) \) of the number \( p \) of the variables to be calculated.

Due to the randomness introduced by the execution time availability, the value of the cost function becomes stochastic. We consider the expected value of the cost \( E[J_{N_h}] \) where the expectation for the cost in (2) is taken with respect to the sequence \( \{\tau(k)\} \). We denote this cost by \( \overline{J}_{N_h} \equiv E[J_{N_h}] \). We say that a system is stable if \( \overline{J} \triangleq \lim_{N_h \to \infty} \overline{J}_{N_h} \) is bounded.

**Problem Description:** For the unconstrained problem setup, we wish to design a control algorithm to minimize the cost \( \overline{J}_{N_h} \). In the constrained problem setup, algorithms to optimize \( E[J_{N_h}] \) globally are not available in general, even without additional processing resource limitations. Thus, in such cases, we are interested merely in stabilizing the plant.

### III. UNCONSTRAINED LINEAR SYSTEMS

**Benchmark Algorithm:** Consider a baseline algorithm \( A_1 \) in which the processor calculates the optimal control input if the execution time is sufficient and applies zero input otherwise. However, the control law takes into account the possibility that the input may not be applied at every time step. Thus, the process evolves as a stochastically switched system

\[
x(k + 1) = \begin{cases} 
Ax(k) & \text{with prob. } 1 - q \\
Ax(k) + Bu(k) & \text{with prob. } q,
\end{cases}
\]

where \( q \) is the probability that enough execution time was available to calculate the control input. This system is identical to a system being controlled across a packet dropping channel with drop probability \( 1 - q \) as studied in [8].

**Proposition 3.1 (Benchmark Algorithm):** Consider the problem formulation in Section II with the baseline algorithm \( A_1 \) being used. Then,

- **Optimal Control:** The optimal control is given by
  \[
u(k) = - (R + B^T T(k) B)^{-1} B^T T(k) A x(k),\]
  where \( T(k) = A^T (k+1) A + Q - q A^T T(k+1) A + B (R + B^T T(k+1) B)^{-1} B^T T(k+1) A, \) with the initial condition \( T(N_h + 1) = P_{N_h+1} \).

- **Performance:** With this input, the cost \( E[J_{N_h}] = E[x^T(0) T(0) x(0)] \).

- **Stabilizability Conditions:** A condition that is both necessary and sufficient for stabilizability is that there exists a matrix \( K \) and a positive definite matrix \( P \) such that \( P > Q + (1 - q) (A^T P A) + q ((A + BK)^T P (A + BK) + K^T R K) \).

**Proposed Algorithm:** The proposed algorithm refines the control input through a sequential calculation of the components of the control input vector as more execution time becomes available. While calculating the \( j \)-th component, the values of the first \( j - 1 \) components already calculated are utilized and in keeping with Assumption 1, the rest of the components are assumed to be zero. However, all the components need not be calculated with either the same frequency or the same priority. We focus on priority rules that can be described according to a Markov chain, and hence have memory to counteract situations where due to the processor time constraints, a particular component is not calculated for a long time. For pedagogical ease, we present the algorithm for the case when \( m = 2 \). Denote \( B = [B_1 \ B_2] \). Define the mode \( r(k) \) and the corresponding matrix \( B_r(k) \) for each of the five possible outcomes at time \( k \) as follows:

- If no control input is calculated, \( r(k) = 0 \) and \( B_r(k) = 0 \).
- If only \( u_1(k) \) is calculated, \( r(k) = 1 \), \( B_r(k) = [B_1 \ 0] \).
- If \( u_1(k) \) is calculated first, and then \( u_2(k) \) is calculated, define \( r(k) = 2 \) and \( B_r(k) = B \).
- If only \( u_2(k) \) is calculated, \( r(k) = 3 \), \( B_r(k) = [0 \ B_2] \).
- If \( u_2(k) \) is calculated first, and then \( u_1(k) \) is calculated, define \( r(k) = 4 \) and \( B_r(k) = B \).

The process (1) evolves as a discrete time switched system

\[
x(k + 1) = Ax(k) + B_r(k) u(k),
\]

with continuous state \( x(k) \) and a discrete mode \( r(k) \) which switches probabilistically, due both to the designer specified mode switching sequence and to the processor time constraints. Define the set \( \{q_{11} \} = \text{Prob}(u_1(k) \text{ is calculated before } u_2(k) \text{ and } r(k - 1) = i) \) and the probability \( q_2 = \text{Prob}(u_1(0) \text{ is calculated before } u_2(0)) \) for \( i = 0, 1, \ldots, 4 \). Denote a sequence \( c = (c(0), c(1), \ldots, c(N_h)) \) where each element \( c(j) \in \{1, 2\} \).

For any \( j \), define \( \bar{c}(j) \) as \( \bar{c}(j) = 1 \text{ if } c(j) = 2, \) and otherwise. The proposed algorithm \( A_2 \) operates as follows. We specify how to optimally calculate the control inputs \( u_1(k) \) and \( u_2(k) \) in Theorem 3.2.

1. **Initialization:** (Control inputs calculated using Thm 3.2)
   - 1.1 At time \( k = 0 \), set \( c(0) = 1 \) with probability \( q_1 \) and \( c(0) = 2 \) otherwise.
   - 1.2 If processor available, calculate the control input \( u_{e(0)}(0) \); else set \( r(0) = 0 \).
   - 1.3 If processor still available, calculate the control input \( u_{e(0)}(0) \); else
     - 1.3.1 Set \( u_{e(0)} = 0 \) if \( c(0) = 1 \), else set \( r(0) = 3 \).
     - 1.3.2 Set \( k = 1 \) and go to Step 2.1.
   - 1.4 If \( c(0) = 1 \), set \( r(0) = 2 \); else set \( r(0) = 4 \).
   - 1.5 Set \( k = 1 \) and go to Step 2.1.

2. **Update:** (Control inputs calculated using Thm 3.2)
   - 2.1 Given \( r(k - 1) \) and the probabilities \( \{q_{11}\} \), calculate the probability that \( u_1(k) \) is calculated before \( u_2(k) \) and denote it by \( p \). Set \( c(k) = 1 \) with probability \( p \) and \( c(k) = 2 \) otherwise.
   - 2.2 If processor available, calculate the control input \( u_{e(k)} \); else
     - 2.2.1 Set \( r(k) = 0 \), \( u_1(k) = u_2(k) = 0 \).
     - 2.2.2 If \( k < N_h \), set \( k = k + 1 \) and go to Step 2.1; else break.
   - 2.3 If processor still available, calculate the control input \( u_{e(k)} \); else
     - 2.3.1 Set \( u_{e(k)} = 0 \) if \( c(k) = 1 \), set \( r(k) = 1 \); else set \( r(k) = 3 \).
     - 2.3.2 If \( k < N_h \), set \( k = k + 1 \) and go to Step 2.1; else break.

---

Limited circulation. For review only
2.4 If \( c(k) = 1 \), set \( r(k) = 2 \); else set \( r(k) = 4 \).

2.5 If \( k < N_h \), set \( k = k+1 \) and go to Step 2.1; else break.

Analysis: Denote by \( p_1 \) the probability that the processor provides enough time to calculate only one component of the control vector, and by \( p_2 \) the probability that the processor provides enough time to calculate both the components sequentially. For given probabilities \( \{q_{1i}\} \), the system (4) is a Markovian jump linear system (MJLS), with a \( 5 \times 5 \) mode transition probability matrix \( P \) such that for \( i = 0, \ldots, 4 \), the \( i \)-th row of \( P \) is given by the vector \( [1 - p_1 - p_2 \ q_11p_1 \ q_11p_2 \ q_12p_1 \ q_12p_2] \). Let \( p_{ij} \) denote the \((i,j)\)-th element of \( P \). The initial probability vector at time 0 given by \( \pi(0) = [1 - p_1 - p_2 \ q_11p_1 \ q_11p_2 \ q_12p_1 \ q_12p_2] \). Let \( \pi_k(k) \triangleq \text{Prob}(r(k) = i) \) be the probability of the mode \( i \) being active at time \( k \). Note that the controller does not have a priori knowledge of the processor availability and hence does not know what mode the process is in at any time. Thus, for instance, the controller cannot a priori distinguish between \( r(k) = 1 \) and \( r(k) = 2 \) and the control input \( u_1(k) \) needs to be identical in both cases. Thus, the calculation of optimal control inputs cannot be done directly according to conventional results in MJLS theory. Following Assumptions 1 and 2, we assume the following for the calculation of control inputs.

Assumption 4: The calculation of the input \( u_1(k) \) (resp. \( u_2(k) \)) in modes \( r(k) = 1, 2 \) (resp. \( r(k) = 3, 4 \)) is done by assuming \( u_2(k) = 0 \) (resp. \( u_1(k) = 0 \)) for that time step.

Theorem 3.2 (Optimal Control): Consider the MJLS in (4) with algorithm \( A_2 \). The optimal controller is given by

\[
\begin{bmatrix}
0 \\
-(R_{11} + B_1^T P_{1k+1}^1 B_1) & 0 \\
-(R_{11} + B_1^T P_{1k+1}^1 B_1 & 0 & 0 \\
-(R_{22} + B_2^T P_{2k+1}^3 B_2) & 0 & 0
\end{bmatrix}
\]

where \( \forall j = 0, \ldots, 4 \), \( P_{j+1} = P_{N_h+1} \), and the \((i,j)\)-th element of the transition probability matrix \( P \),

\[
\begin{align*}
&f^0(P^0_{k+1}) \triangleq Q + A^T P_{k+1}^0 A \\
&f^1(P^1_{k+1}) \triangleq Q + A^T P_{k+1}^1 A \\
&f^2(P^2_{k+1}) \triangleq Q + A^T P_{k+1}^2 A \\
&f^3(P^3_{k+1}) \triangleq Q + A^T P_{k+1}^3 A \\
&f^4(P^4_{k+1}) \triangleq Q + A^T P_{k+1}^4 A
\end{align*}
\]

The completion of squares argument now yields that the optimal choice of control is \( u_2(N_h) = \sum_{j=0}^4 P_{j+1} f^j(P_k^j) \), \( p_{ij} = \text{Prob}(r(k) = j | r(k - 1) = i) \) is the \((i,j)\)-th element of the transition probability matrix \( P \),

\[
\begin{align*}
&f^0(P^0_{k+1}) \triangleq Q + A^T P_{k+1}^0 A \\
&f^1(P^1_{k+1}) \triangleq Q + A^T P_{k+1}^1 A \\
&f^2(P^2_{k+1}) \triangleq Q + A^T P_{k+1}^2 A \\
&f^3(P^3_{k+1}) \triangleq Q + A^T P_{k+1}^3 A \\
&f^4(P^4_{k+1}) \triangleq Q + A^T P_{k+1}^4 A
\end{align*}
\]

The completion of squares argument now yields that the optimal choice of control is \( u_2(N_h) = \sum_{j=0}^4 P_{j+1} f^j(P_k^j) \), \( p_{ij} = \text{Prob}(r(k) = j | r(k - 1) = i) \) is the \((i,j)\)-th element of the transition probability matrix \( P \),

\[
\begin{align*}
&f^0(P^0_{k+1}) \triangleq Q + A^T P_{k+1}^0 A \\
&f^1(P^1_{k+1}) \triangleq Q + A^T P_{k+1}^1 A \\
&f^2(P^2_{k+1}) \triangleq Q + A^T P_{k+1}^2 A \\
&f^3(P^3_{k+1}) \triangleq Q + A^T P_{k+1}^3 A \\
&f^4(P^4_{k+1}) \triangleq Q + A^T P_{k+1}^4 A
\end{align*}
\]

\[
\begin{align*}
&f^0(P^0_{k+1}) \triangleq Q + A^T P_{k+1}^0 A \\
&f^1(P^1_{k+1}) \triangleq Q + A^T P_{k+1}^1 A \\
&f^2(P^2_{k+1}) \triangleq Q + A^T P_{k+1}^2 A \\
&f^3(P^3_{k+1}) \triangleq Q + A^T P_{k+1}^3 A \\
&f^4(P^4_{k+1}) \triangleq Q + A^T P_{k+1}^4 A
\end{align*}
\]

Limited circulation. For review only.
\[-(B_{2,2} + B_{2}^T P_{N+1}^2 B_2)^{-1} B_{2}^T P_{N+1}^2 A(N_h) x(N_h), \text{ and} \]

\[T_{2, N_h}^2 = E \left( x^T(N_h) f^2(P_{N+1}^2) x(N_h) \right). \]

- The argument for \(r(N_h) = 3\) is similar to that for \(r(N_h) = 1\), and that for \(r(N_h) = 4\) is similar to that for \(r(N_h) = 2\).

Define the matrices \(P_{N+1}^j, j = 0, \ldots, 4\), as \(P_{N+1}^j \neq 0\) \(\sum_{i=0}^{4} p_{j,i} f^i(P_{N+1}^i)\). Then, with the optimizing choice for \(u(N_h)\) for the various modes, the term \(T_{N_h}\) becomes

\[T_{N_h} = \sum_{i=0}^{4} \sum_{j=0}^{4} \pi_i(N_h) \text{Prob}(r(N_h) - 1 = j | r(N_h) = i) \cdot E \left( x^T(N_h) f^i(P_{N+1}^i) x(N_h) | r(N_h) - 1 = j \right)
\]

\[= \sum_{j=0}^{4} \pi_j(N_h - 1) E \left( x^T(N_h) P_{N+1}^j x(N_h) | r(N_h) - 1 = j \right).
\]

Thus, with the optimizing choice of the control \(u(N_h)\), \(\tilde{J}_{N_h}\) can be rewritten as

\[\tilde{J}_{N_h} = E \left( \sum_{k=0}^{N_h-1} [x^T(k) Q x(k) + u^T(k) R u(k)] \right) + E \left( \sum_{j=0}^{4} \pi_j(N_h - 1) x^T(N_h) P_{N+1}^j x(N_h) | r(N_h) - 1 = j \right),\]

and we need to design \(u(N_h - 1), \ldots, u(0)\). We can again extract the terms in the cost function that depend only on \(x(N_h - 1)\) and \(u(N_h - 1)\) and write them as \(T_{N_h-1} = \sum_{i=0}^{4} \pi_i(N_h - 1) T_{N_h-1}^i\), with \(T_{N_h-1}^i\) defined exactly as in (5) (except for a shift by one time step). Since our argument was independent of the specific value of \(N_h\), the result follows.

**Corollary 3.3 (Performance):** Consider the problem formulation in Section II with the algorithm \(A_2\) and the optimal control inputs from Theorem 3.2. For a given initial state \(x(0)\), the resulting cost \(J_{N_h} = \sum_{i=0}^{4} \pi_i(0) x^T(0) P_{0}^i x(0)\) where the terms \(P_{0}^i\) are defined in Theorem 3.2.

**Theorem 3.4 (Stability Conditions):** Consider the problem formulation in Section II with the algorithm \(A_2\) being used with the control inputs as identified in Theorem 3.2.

- A necessary condition for stability is that the following inequalities are true

\[p_{00} \rho(A_2)^2 < 1, \quad p_{11} \rho(A_2)^2 < 1, \quad p_{33} \rho(A_2)^2 < 1 \quad (6)\]

where \(\rho(A_1)\) (resp. \(\rho(A_2)\)) is the spectral radius of the uncontrollable subspace of the matrix \(A\) when \((A, B_1)\) (resp. \((A, B_2)\)) is put in the controllable canonical form.

- A sufficient condition for stability of the system is that there exist five positive definite matrices \(X_0, \ldots, X_4\) and matrices \(K_{1,0}, \ldots, K_{2,0}, \ldots, K_{2,4}\) such that

\[X_j > (p_{00}(Q + A^T X_0 A) + p_{11}(A + B_1 K_{1,1})^T X_1 (A + B_1 K_{1,1}) + Q + K_{1,1}^T R_{1,1} K_{1,1}) \]

\[+ p_{22}(A + B_2 K_{2,2})^T X_2 (A + B_2 K_{2,2}) + Q + K_{2,2}^T R_{2,2} K_{2,2}) \]

\[+ p_{33}(A + B_2 K_{3,3})^T X_3 (A + B_2 K_{3,3}) + Q + K_{3,3}^T R_{2,2} K_{3,3}) \]

\[+ p_{44}(A + B_2 K_{4,4})^T X_4 (A + B_2 K_{4,4}) + Q + K_{4,4}^T R_{2,2} K_{4,4}). \quad (7)\]

**Proof:** From Theorem 3.2, a necessary condition for stability with algorithm \(A_2\) is that the the terms \(P_{0}^0, P_{0}^1, \) and \(P_{0}^3\) do not diverge as \(N_h \to \infty\). Consider matrix recursions

\[Z_0^k = p_{00}(Q + A^T Z_0^{k-1} A) + (1 - p_{00}) Q \quad (8)\]

\[Z_{1}^k = (1 - p_{11}) Q + p_{11}(Q + A^T Z_{1}^{k-1} A) - A^T Z_{1}^{k-1} B_1 (B_1^T Z_{1}^{k-1} B_1 + R_{1,1})^{-1} B_1^T Z_{1}^{k-1} A \quad (9)\]

\[Z_2^k = (1 - p_{22}) Q + p_{22}(Q + A^T Z_{2}^{k-1} A) - A^T Z_{2}^{k-1} B_1 (B_1^T Z_{2}^{k-1} B_1 + R_{1,1})^{-1} B_1^T Z_{2}^{k-1} A, \quad (10)\]

with the initial conditions \(Z_0^{0, N_h+1} = P_{N_h+1}, Z_{1}^{0, N_h+1} = P_{N_h+1}\) and \(Z_{2}^{0, N_h+1} = P_{N_h+1}\). Since for every horizon \(N_h\), the relations \(Z_0^{0} \leq P_0^{0}, Z_0^{1} \leq P_0^{1}\) and \(Z_0^{2} \leq P_0^{2}\), hold, the necessary condition is that the terms \(Z_{0}^{k}, Z_{1}^{k}\) and \(Z_{2}^{k}\) do not diverge as \(N_h \to \infty\). Conditions (6) can be obtained as the necessary conditions for stability of (8)-(10) using [4, Proposition V.3].

To prove sufficiency, we consider a system that evolves as

\[z(k + 1) = A(z(k) + \tilde{B}_i(k) \tilde{u}(k)), \quad (11)\]

with \(\hat{B}_0 = 0, \hat{B}_1 = \hat{B}_2 = 1, \text{ and } \hat{B}_3 = \hat{B}_4 = 2,\)

the initial condition \(z(0) = x(0)\) of the original system, the same transition probability matrix as the original system and the optimal controller that minimizes cost \(J_{N_h}\). Since the cost achieved for this system with any control is always greater than the cost in the original system with the optimal control, a sufficient condition for stabilizability of the original system is that (11) is stabilizable since (7) is a sufficient condition for stabilizability of the system (11) [2], the result follows.

**IV. Constrained Systems**

The constrained problem set up is significantly more difficult, even without limitations on computational resources.

- For general sets S, there are no systematic methods to design the control laws. We use receding horizon control (RHC) in which an optimization problem \(\mathcal{O}\) of the form

\[
\min_{\{u(j)\}_{j=0}^{N} \in \mathcal{S}} \sum_{i=0}^{N} \left( x^T(k + i) Q x(k + i) + u^T(k + i) R u(k + i) \right) + x^T(k + N + 1) P x(k + N + 1) \quad \text{s.t.} \quad x(k + 1) = A z(k) + B u(k), \quad x(k + N + 1) \in \mathcal{T}, \quad (12) \]

is solved at every time k. Define \(F(x) := x^T P x + L(x, u) \>| x^T Q x + u^T R u \quad \text{and \ the \ terminal \ set} \quad \mathcal{T} \quad \text{is \ assumed \ to \ be \ such \ that} \quad \text{for \ all \ state \ values} \quad x \in \mathcal{T}, \quad \text{there \ exists \ a \ terminal \ control \ law} \quad \kappa(\xi), \quad \text{that \ ensures} \quad \|
\]

\[F(A \xi + B \kappa(\xi)) - F(\xi) + L(\xi, \kappa(\xi)) \leq 0, \quad A \xi + B \kappa(\xi) \in \mathcal{T}, \quad h(\xi, u) \in \mathcal{S} \quad (14) \]

We assume that \(Q \text{ and } R\) are such that for all state and control values such that \(h(x, u) \in \mathcal{S}, x^T Q x + u^T R u \geq \alpha(||x||), \forall x \in \mathcal{X}_N, \text{ where } \alpha \text{ is a class } K_{\infty} \text{ function.}

- For a purely stochastic model of processor availability, the system can run open loop for arbitrarily long times. Thus, not much can be said, in general, about the feasibility of the optimization problem to be solved in the RHC algorithm. To guarantee feasibility, we assume the following.
Assumption 5: The scheduling algorithm is such that the available execution time \( \tau(k) > 2f(N + 1) \) at least once for any block of consecutive \( N \) time steps.

Assumption 5 is required solely to ensure the feasibility of the optimization problem \( \mathcal{O} \). If the feasibility can be guaranteed in some other fashion, the algorithm \( \mathcal{A}_3 \) is unchanged. Instead of Assumption 1, we use the following alternative. Since RHC yields a control input sequence whenever the optimization problem is solved, this trajectory can be stored in a buffer.

During the next \( N \) steps, if the input cannot be calculated, the control input for the corresponding time step is retrieved from the buffer and applied. Whenever optimization problem for a control input is solved, the corresponding buffer is overwritten.

Algorithm Description: Define the modes \( r(k) \) and the probabilities \( q_{i1} \) as in Section III. Define two buffers \( \{b_1(j)\}_{j=0}^N \) and \( \{b_2(j)\}_{j=0}^N \) that are used to store control inputs \( \{u_1(k + j)\}_{j=0}^N \) and \( \{u_2(k + j)\}_{j=0}^N \), respectively at each time \( k \).

The proposed algorithm \( \mathcal{A}_3 \) is given below (we assume for simplicity that \( N_i \to \infty \)):

1. Initialization:

   1.1 At time \( k = 0 \), set \( c(0) = 1 \) with probability \( q_{11} \) and \( c(0) = 2 \) otherwise. Also initialize the two buffers \( b_1(i) = 0 \), \( i = 1, 2 \).

   1.2 If processor available, calculate the control inputs \( \{u_{c(0)}(t)\}_{t=0}^N \) and store in the corresponding buffer \( b_{c(0)}(i) \); else set \( r(0) = 0 \), \( u_1(0) = u_2(0) = 0 \), \( k = 1 \) and go to Step 2.1.

   1.3 If processor still available, calculate the control inputs \( \{u_{c(0)}(t)\}_{t=0}^N \) and store in the corresponding buffer \( b_{c(0)}(i) \); else

   1.3.1 If \( c(0) = 1 \), set \( r(0) = 1 \); else set \( r(0) = 3 \).

   1.3.2 Set \( u_{c(0)}(0) = b(0) \), \( u_{c(0)}(0) = 0 \), \( k = 1 \) and go to Step 2.1.

   1.4 If \( c(1) = 1 \), set \( r(1) = 2 \); else set \( r(1) = 4 \).

   1.5 Set \( u_1(0) = b_1(0) \), \( u_2(0) = b_2(0) \), \( k = 1 \) and go to Step 2.1.

2. Update:

   2.1 Set \( b_1(k) = b_1(k + 1) \) for \( k = 0, \ldots, N - 1 \), and \( b_1(N) = 0 \), \( i = 1, 2 \).

   2.2 Given \( r(k - 1) \), and the probabilities \( \{q_{i1}\} \), calculate \( p \) as the probability that \( u_1(k) \) is calculated before \( u_2(k) \). Set \( c(k) = 1 \) with probability \( p \) and \( c(k) = 2 \) otherwise.

   2.3 If processor available, calculate the control input \( \{u_{c(k)}(k + t)\}_{t=0}^N \) and store in the corresponding buffer \( b_{c(k)}(i) \); else

   2.3.1 Set \( r(k) = 0 \), \( u_1(k) = k_1(0) \), \( u_2(k) = k_2(0) \).

   2.3.2 Set \( k = k + 1 \) and go to Step 2.1.

   2.4 If processor still available, calculate the control inputs \( \{u_{c(k)}(k + t)\}_{t=0}^N \) and store in the corresponding buffer \( b_{c(k)}(i) \); else

   2.4.1 If \( c(k) = 1 \), set \( r(k) = 1 \); else set \( r(k) = 3 \).

   2.4.2 Set \( u_1(k) = b_1(0) \), \( i = 1, 2 \).

   2.4.3 Set \( k = k + 1 \) and go to Step 2.1.

   2.5 If \( c(k) = 1 \), set \( r(k) = 2 \); else set \( r(k) = 4 \).

   2.6 Set \( u_1(k) = b_1(0) \), \( i = 1, 2 \).

   2.7 Set \( k = k + 1 \) and go to Step 2.1.

To calculate the control inputs, we use the optimization problem \( \mathcal{O} \). However, if \( c(k) = 1 \), the inputs \( \{u_1(k + j)\}_{j=0}^N \) are calculated by solving the optimization problem while assuming the inputs \( \{u_2(k + j)\}_{j=0}^N \) to be equal to the elements \( \{b_2(j)\}_{j=0}^N \) (and hence not to be optimized over). In turn, the inputs \( \{u_2(k + j)\}_{j=0}^N \) are calculated by solving the optimization problem \( \mathcal{O} \) while assuming the inputs \( \{u_1(k + j)\}_{j=0}^N \) to be at the values \( \{b_1(j)\}_{j=0}^N \) that have just been calculated. A similar remark holds for \( c(k) = 2 \). The algorithm can be generalized to further refine the input if even more time is available. Thus, if \( c(k) = 1 \), and the processor has more time available, then at the end of Step 2.4 the inputs \( \{u_1(k + j)\}_{j=0}^N \) can be refined by resolving the problem \( \mathcal{O} \) with the inputs \( \{u_2(k + j)\}_{j=0}^N \) assumed to be the values calculated in the previous step. This process can be iterated till convergence.

Proposition 4.1 (Feasibility): Consider the problem posed in Section II with algorithm \( \mathcal{A}_3 \). Denote by \( k_0 \geq 0 \) the first time at which \( \tau(k) > 2f(N + 1) \). If the optimization problems to be solved in the algorithm \( \mathcal{A}_3 \) are feasible at time \( k_0 \), then they are feasible at any time step \( k \geq k_0 \).

Proof: At time \( \tau \), denote the last time at which the control input \( u_1(\cdot) \) (resp. \( u_2(\cdot) \)) could be calculated by \( t_1(\tau) \) (resp. \( t_2(\tau) \)). By assumption, the problem is feasible at time \( k_0 \). Let the problem be feasible at time \( m \geq k_0 \). At time \( m + 1 \), due to Assumption 5, \((m + 1) - t_1(m + 1) \leq (N - 1) \) for \( i = 1, 2 \). Set \( t = \max(t_1(m + 1), t_2(m + 1)) \), and let \( c(t) = j \). Thus, at time \( t \), the control inputs \( \{u_i(t + i)\}_{i=0}^N \) were calculated and stored in the buffer \( b_j \), such that if the control inputs \( u_i(t + i) = b_j(i) \) are followed for \( k = 1, 2 \) and \( i = 0, \ldots, N \), then the process state \( x(t + N) \in \mathcal{T} \). Thus, at time \( m + 1 \), one set of feasible inputs is provided by the buffer entries \( u_k(m + 1 + i) = b_k(i)_{i=0}^N \), \( k = 1, 2 \), such that the state \( x(t + N) \in \mathcal{T} \). Since a feasible control law exists in the set \( \mathcal{T} \) and the set is invariant, this implies that at time \( m + 1 \), the problem is feasible. Thus, by mathematical induction, the problem is feasible for all \( m \geq k_0 \).

For general sets \( \mathcal{S} \), it is difficult to calculate the moments of the state even in the absence of processor constraints. Instead, we provide the following characterization.

Theorem 4.2 (Stability): Consider the problem posed in Section II with the algorithm \( \mathcal{A}_3 \). Then the system is globally stable with the state \( x(k) \) converging to the origin as \( k \to \infty \).

Proof: Consider the sequence of time steps \( \{k_i\} \in \mathbb{N}_0 \) such that at each instant \( k_i \), \( \tau(k_i) > 2f(N + 1) \). Denote the control input corresponding to time \( m \) as computed by solving the optimization problem \( \mathcal{O} \) at time \( k_i \) by \( u_m(k_i) \) and the entire control sequence by \( U_i \). Denote the value of the cost (12) when control input sequence \( U \) is used by \( V(U) \). Finally define the control sequence \( \hat{U} = (u(k_1 + N + 1), \ldots, u(k_i + N + 1), \ldots, u(k_1 + N + 1), a(k_i + 1, + N) \) where the terms \( a(\cdot) \) are calculated using the terminal control law \( \kappa(\cdot) \). Consider

\[
V(\hat{U}) - V(U_i) = \sum_{l=k_i+N+1}^{k_i+N+1} \left[ F(\hat{x}(l+1)) - F(x(l)) \right] + L(\hat{x}(l), \hat{u}(l)) - \sum_{l=k_i}^{k_i'} L(x(l), u(l; k_i)) \leq - \sum_{l=k_i}^{k_i+1} \alpha(||x(l)||) \leq 0,
\]
where \( \bar{x}(.) \) is obtained by using control inputs from \( \bar{U} \), \( \hat{x} \) is obtained by using control inputs from \( U_i \), and the first inequality follows from (14). Since \( U_{i+1} \) is the optimal control input sequence and \( \bar{U} \) is a particular control input sequence over the same time steps, \( V(U_{i+1}) \leq V(U) \). Thus, \( V(U_{i+1}) - V(U_i) \leq V(U) - V(U_i) \leq 0 \). Since \( V(U_i) \geq 0 \) and this argument holds for any \( i \in \mathbb{N}_0 \), \( \{V(U_i), i \in \mathbb{N}_0\} \) forms a non-increasing convergent sequence with limit value 0. Thus, \( \lim_{i \to \infty} V(U_{i+1}) - V(U_i) = \lim_{i \to \infty} \sum_{k=1}^{\infty} \alpha(||x(l)||) = 0 \). This yields \( \lim_{i \to \infty} \alpha(||x(l)||) = 0 \) or \( \lim_{i \to \infty} ||x(l)|| = 0 \). Thus, origin is a globally attractive point for \( x(k) \). Since times \( k_i \)'s are at most \( N \) steps apart and all controls \( u(m, k_i) \) are bounded, the state is bounded. Thus, the system is stable.

V. NUMERICAL EXAMPLES

Consider a process of the form (1) with matrices

\[
A = \gamma \begin{bmatrix} 1.1 & 0.1 \\ 0.1 & 1.1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}, \quad (15)
\]

and \( Q = R = I \). We assume that the execution time available is uniformly distributed in the interval \([0, 1]\). The time required for calculation of 1 control input is assumed to be 0.3 units, and for 2 control inputs to be 0.6 units. For the algorithm \( A_2 \), we assume that mode selection is done in a memoryless fashion, but after optimizing over the probabilities of selection of the two components to be the first component to be calculated. Figure 1 depicts the costs achieved with the baseline algorithm \( A_1 \) and the proposed algorithm \( A_2 \) with varying margins of instability for the process as \( \gamma \) is varied. We see a significant improvement in cost, particularly as the spectral radius increases. The stability region also increases with the proposed algorithm. For comparison, Theorem 3.4 predicts \( \gamma < 1.23 \) as a sufficient condition and \( \gamma < 1.52 \) as a necessary condition for stability. For \( \gamma = 0.9 \), we then impose the constraint that each control value needs to be less than 0.01 in magnitude. We implemented algorithm \( A_3 \) using Matlab on a Windows XP machine. The optimization problem was solved using the quadprog function in Matlab, and the execution time noted using the functions tic and toc. The execution time available was assumed to be uniformly distributed in the interval \([0, 0.03]\) seconds. We did not impose any terminal set constraint. The simulation length was 50 time steps, and the percentage improvement in cost achieved using the proposed algorithm is plotted as a function of the horizon length in Figure 2. As the horizon length is increased, the performance improvement using the proposed algorithm increases. For very small horizon lengths, there is sufficient time available for both algorithms to calculate both control inputs at every time step. Thus, the proposed algorithm may perform worse since it calculates the control inputs one at a time, rather than jointly.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

We proposed an anytime control algorithm based on computing the control inputs sequentially. For unconstrained linear systems, we presented a Markovian jump linear system framework and analyzed the stability and performance gain. For constrained systems, we proposed a receding horizon control-based extension. Several possible avenues for future work remain, including analytic expressions for performance as a function of parameters like horizon length and joint design of control algorithms and processor schedulers.

REFERENCES