Computing Time Delay and Its Effects on Real-Time Control Systems

Kang G. Shin and Xianzhong Cui

Abstract—**The reliability of a real-time digital control computer depends not only on the reliability of the hardware and software used, but also on the time delay in computing the control output, because of the negative effects of computing time delay on control system performance. For a given fixed sampling interval, the effects of computing time delay are classified into the delay and loss problems. The delay problem occurs when the computing time delay is nonzero but smaller than the sampling interval, while the loss problem occurs when the computing time delay is greater than, or equal to, the sampling interval, i.e., loss of the control output. These two problems are analyzed as a means of evaluating real-time control systems. First, a generic analysis of the effects of computing time delay is presented along with necessary conditions for system stability. Then, we present both qualitative and quantitative analyses of the computing time delay effects on a robot control system, deriving upper bounds of the computing time delay with respect to system stability and system performance.**

I. Introduction

A real-time digital control computer or controller computer can be thought of as a three-stage pipe: data acquisition from sensors, data processing to generate control/display commands, and outputting the results to actuators/display devices. Although each of the three stages will take time to complete, this paper is concerned only with the time taken by the most complicated stage, data processing, since the other two are much simpler and more static. More precisely, the time taken to execute programs that implement control algorithms—called the computing time delay—is the subject of this paper.

A controller computer implements the underlying control algorithms by executing a sequence of instructions. Unlike analog control systems, the reliability of a digital control system depends not only on the MTBF (mean time between failures) of the controller hardware and software, but also on the delay in executing control algorithms on the controller computer. The execution time for a control algorithm is defined as the period from its trigger to generation of a corresponding control command. It is an extra time delay that is introduced to the feedback loop in a controlled system. Because of the existence of conditional branches, resource sharing delays, and processing exceptions, the execution time for a given control algorithm, or the computing time delay, is a piecewise continuous, random variable which is usually smaller than the corresponding sampling interval. Conventional controllers are designed without considering the computing time delay. Thus, it is very important to analyze the effects of computing time delay on control system performance when control algorithms are implemented on digital computers.

The computing time delay is quite different from the usual system time delay and cannot be taken care of prior to putting a system in use due to its randomness caused by, for example, the data-dependent branches and loop counts in a program that implements the control algorithm under consideration.

When the computing time delay is long relative to the sampling interval (but small relative to the mission lifetime), it may seriously affect control system performance. Depending on the magnitude of computing time delay relative to the sampling interval, its effects on the control system are classified into either a delay or loss problem. To be more precise, let ξ and T_s denote the computing time delay and the sampling interval, respectively. A delay problem results when $0 < \xi < T_s$, and the loss problem occurs when $\xi \geq T_s$. The former represents the undesirable effects (for example, in terms of operational cost or energy) caused by a nonzero computing time delay smaller than the deadline (that is, the beginning of the next sampling interval) of a control algorithm or task,[1] while the latter represents the case of no update of control output for one or more sampling intervals.

To implement a controller on a digital computer, the sampling rate must be chosen carefully by not only satisfying the conditions of the Shannon's sampling theorem, but also achieving the desired performance. A good example of this can be found in [4] where a series of robot control experiments were conducted for different sampling rates while keeping the controller gains fixed. A higher sampling rate is shown to imply improved performance and higher stiffness (or disturbance rejection property). Increasing the sampling rate, however, will make the computing time delay effects more pronounced on control system performance. These effects cannot be neglected, especially when the time constant of the plant is short and the order of the plant is high [6]. To remedy this problem, an optimal state feedback control law was proposed in [6]. Instead of using the current-state feedback $\boldsymbol{u}(k) = -\boldsymbol{K}\boldsymbol{x}(k)$, the control input is formed as $\boldsymbol{u}(k) = -\boldsymbol{K}\boldsymbol{A}\boldsymbol{x}(k-1) - \boldsymbol{K}\boldsymbol{B}\boldsymbol{u}(k-1)$; that is, the computing time delay is approximated to be one sampling interval. The sampling interval T_s is chosen to be the same as the computation time of the control algorithm. In [2], the computing time delay is represented as a delayed state measurement, and an averaging A/D device is used for the measurement. Then for an LQG-type sampled-data regulator problem, an equivalent discrete-time problem is shown to have

Manuscript received August 25, 1993; revised October 18, 1994. Recommended by Associate Editor, G. Suski. This work was supported in part by Office of Naval Research Grant N00014–91–J–1115 and NASA Grant NAG–1–1220.

The authors are with the Real-Time Computing Laboratory Division of Computer Science and Engineering, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109-2122 USA.

IEEE Log Number 9410675.

[1]The terms "control task" and "control algorithm" will henceforth be used interchangeably.

an increased system order. A design procedure was proposed there for this equivalent discrete-time problem.

The work in both [6] and [2], however, did not consider the randomness of computing time delay. Approximating the computing time delay with one or more sampling intervals and incorporating it into controller design was the basic idea used there. As mentioned earlier, the computing time delay problem must account for the random effects of data-dependent conditional branches/loops and the unpredictable delays in sharing resources during the execution of control algorithms. So, for a complex control system, estimating the maximum delay or assuming the computing time delay to be constant is in general neither realistic nor possible. Note that the computing time delay problem is not only caused by a complicated control algorithm, but also by those systems which require very short sampling intervals and/or many logical decisions and conditional searches (for decisions).

From the computer system's point of view, one may try to make the computing time delay as predictable as possible. Note, however, that we cannot completely eliminate the unpredictability of computing time delay, due to, for example, contention for using shared resources which is workload dependent. From the viewpoint of control algorithm design, it is important to know i) how the control system's stability and performance are affected by a given computing time delay and ii) the maximum tolerable computing time delay for a specified control algorithm. The magnitude of the computing time delay and the number of output losses that the controlled system can tolerate are important indexes in evaluating the performance of any digital control algorithm. These indexes will inherently change with the type of control algorithms and systems under consideration. To demonstrate the importance of the computing time delay, we will evaluate this index for typical real-time control systems, robot control systems, which are briefly described below.

Numerous robot control algorithms have been proposed and their computing time delays estimated. To our best knowledge, however, none of these has analyzed explicitly the effects of computing time delay on control system performance. For example, an adaptive control algorithm based on the computed torque algorithm is reported to require 17 ms on an MC68000 microprocessor [5], and the STP control needs 6.7 ms on an NS32132 microprocessor [3]. These results not only indicate the need of high-speed microprocessors, but also raise an interesting question: can the system tolerate this extra computing time delay? A robot control system is usually evaluated on the basis of tracking accuracy, repetition error, and motion speed. For a nonlinear, time-varying system like a robot, the effects of computing time delay on its performance become significant enough to warrant a careful investigation of various control algorithms before using them. This is also true for all other time-critical control systems such as aircraft and life-support systems. (See [8] for an example of aircraft landing.)

Another example for the analysis of computing time delay effects is related to the application of artificial intelligence (AI) to real-time control systems. In a knowledge-based control system, control actions are usually determined by either searching its knowledge base or looking up tables. This, especially in the case of a heuristic search process, may require a significant amount of time relative to the sampling interval. Unlike a controller based on numerical computations, the time for symbolic reasoning and heuristic searches may vary with the operational conditions of the system. Therefore, both the delay and loss problems should be analyzed for knowledge-based systems.

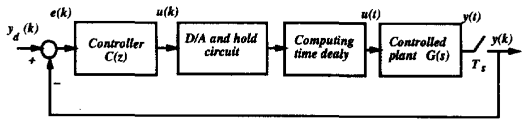

Fig. 1. A digital control system in presence of computing time delay.

We will focus on analyzing the effects of computing time delay on the performance of control systems. Using examples, we will also show how a specific control system is evaluated in terms of computing time delay. In Section II, we first review the basic concepts and definitions related to real-time digital control systems which were introduced in [8]. Then, we address the generic problem of analyzing the effects of computing time delay on control system performance. A generic criterion for the qualitative analysis of computing time delay effects is derived. We present in Section III both qualitative and quantitative analyses of the computing time delay effects on a robot control system. Upper bounds of the computing time delay are derived with respect to system stability and system performance. These upper bounds can be used as an extra constraint on controller design or an index for selection of a microprocessor to implement the control algorithm. The paper concludes with Section IV.

II. Effects of Computing Time Delay on a Control System

The basic concepts and performance measures proposed in [8] are best suited for the analysis of computing time delay effects on control system performance because they are based on task completion times. For completeness, some of the basic concepts in [8] are briefly described first. Then, a generic analysis of the effects of computing time delay is presented along with necessary conditions for system stability.

A. Performance Measures in the Presence of Computing Time Delay

As mentioned earlier, we are interested in analyzing the effects of computing time delay that results from the implementation of a "well-designed" control algorithm on a digital computer. (By "well-designed," we mean that the system is stable and the effects of discretization are accounted for.) The presence of the computing time delay in a control system can be represented by a delay element after the D/A converter and hold circuit, as shown in Fig. 1.

Hence, the analysis of the effects of computing time delay must be done in a continuous-time domain. Note that due to its randomness, the computing time delay is totally different from other usual factors, such as the effects of discretization and system delay, which are not the subject of this paper.

Let $\boldsymbol{X} \subset \boldsymbol{R}^n$ denote the state space and $\boldsymbol{x}(t) \in \boldsymbol{X}$ the state of the controlled system at time t. Evolution of states from time t_0 in the presence of a nonzero computing time delay ξ is represented by

$$\boldsymbol{x}(t) = \Phi(t, t_0, \boldsymbol{x}(t_0), \boldsymbol{u}(t-\xi))$$

where Φ is the state transition map, $\boldsymbol{u}(t) \in \boldsymbol{U}_A \subseteq \boldsymbol{U} \subset \boldsymbol{R}^\ell$ the control input at time t, $\boldsymbol{U}_A$ the admissible input space, and $\boldsymbol{U}$ the input space. The behavior of the system is monitored via $\boldsymbol{y}(t) = \Gamma(t, \boldsymbol{u}(t-\xi), \boldsymbol{x}(t))$, where Γ is the output function, $\boldsymbol{y}(t) \in \boldsymbol{Y} \subset \boldsymbol{R}^m$ the output vector at time t and $\boldsymbol{Y}$ the output space. Let $\boldsymbol{X_A}$ and $\boldsymbol{Y_A}$ be the allowed state space and the allowed output space, respectively. Note that if $\xi = 0$, then $\boldsymbol{u}(t) \in \boldsymbol{U}_A$ implies $\boldsymbol{x}(t) \in \boldsymbol{X}_A$ and $\boldsymbol{y}(t) \in \boldsymbol{Y}_A$. If $0 < \xi < T_s$, the maximum computing time delay that the controlled system can tolerate at time t is defined as the contol system deadline (CSD) at that time

$$d_{\boldsymbol{x}(t)} = \sup_{\boldsymbol{u}(t)\in\boldsymbol{U}_A} \{\xi : \boldsymbol{x}(t) \in \boldsymbol{X}_A\}. \tag{1}$$

This means that if $\xi > d_{\boldsymbol{x}(t)}$, then the system may move out of the allowed space. Note that the CSD at time t is a function of the state of the controlled system at t. For a control task performed during the interval $[t_0, t_1]$, its CSD should be the smallest value of $d_{\boldsymbol{x}(t)}$ for all $t \in [t_0, t_1]$. For all but very simple cases, it is impossible to get a closed-form relationship between CSD's and the allowed space $\boldsymbol{Y_A}$ (see [8] for more on this). So, (1) is usually used as a conceptual definition of the CSD. If the computing time delay associated with a control task is greater than its CSD, a dynamic failure results [8], meaning that the system moved out of the allowed state space (e.g., the system moved into an unstable region).

B. How Does the Delay Problem Affect the System Performance?

As mentioned earlier, system performance and stability are affected by a nonzero computing delay ξ. We want to investigate how the closed-loop system is affected by this nonzero computing delay. Let a closed-loop system be represented by

$$\dot{\boldsymbol{x}}(t,\xi) = \boldsymbol{f}(t, \boldsymbol{x}(t,\xi), \xi), 0 < \xi < T_s. \tag{2}$$

Since ξ can usually be made small relative to the control mission lifetime by using parallel processing or high-speed microprocessors, (2) can be expanded as a Taylor series, and the subsequent first-order approximation gives the equation shown at the bottom of the page. Note that $\boldsymbol{g}(t, \boldsymbol{x})$ is not a function of ξ. From (2), we conclude that the computing time delay affects the system performance through a permanently acting perturbation.

Since the control system is usually designed under the assumption of $\xi = 0$, the closed-loop system is represented by $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t))$. Suppose there exists a Lyapunov function $V(t, \boldsymbol{x})$ which is positive definite, decrescent and

$$\dot{V}(t,\boldsymbol{x}) = \frac{\partial V(t,\boldsymbol{x})}{\partial t} + \frac{\partial V(t,\boldsymbol{x})}{\partial \boldsymbol{x}} \boldsymbol{f}(t,\boldsymbol{x}) \le -r(\| \boldsymbol{x} \|)$$

where $r(\cdot)$ belongs to class K such that the closed-loop system $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t))$ is stable.[2] For $0 < \xi < T_s$, referring to (3), found at the bottom of the page, we have

$$\begin{aligned}\dot{V}_\xi(t,\boldsymbol{x}) &= \left(\frac{\partial V(t,\boldsymbol{x})}{\partial t} + \frac{\partial V(t,\boldsymbol{x})}{\partial \boldsymbol{x}} \boldsymbol{f}(t, \boldsymbol{x}(t,0), 0)\right) \\ &\quad + \xi \frac{\partial V(t,\boldsymbol{x})}{\partial \boldsymbol{x}} \boldsymbol{g}(t,\boldsymbol{x}) \\ &\le -r(\| \boldsymbol{x} \|) + \xi \left| \frac{\partial V(t,\boldsymbol{x})}{\partial \boldsymbol{x}} \boldsymbol{g}(t,\boldsymbol{x}) \right| \\ &\le -r(\| \boldsymbol{x} \|) + \xi \left\| \frac{\partial V(t,\boldsymbol{x})}{\partial \boldsymbol{x}} \right\| \| \boldsymbol{g}(t,\boldsymbol{x}) \|\end{aligned}$$

which results in the following two stability conditions

$$\sup_{t \ge t_0, \|\boldsymbol{x}\| < \rho} \| \boldsymbol{g}(t,\boldsymbol{x}) \| < \infty \tag{4}$$

$$\sup_{t \ge t_0, \|\boldsymbol{x}\| < \rho} \left\| \frac{\partial V(t,\boldsymbol{x})}{\partial \boldsymbol{x}} \right\| < \infty. \tag{5}$$

If conditions (4) and (5) hold, then there exists a $0 < \xi < T_s$ small enough such that $\dot{V}_\xi(t, \boldsymbol{x}) < 0$. Because $V(t, \boldsymbol{x})$ is a positive definite, decrescent function, condition (5) is satisfied. Thus, if condition (4) is satisfied, then there exists a $0 < \xi < T_s$ small enough such that the closed-loop system is uniformly stable. Note that, though the stability analyses in discrete-time and continuous-time systems are not equivalent, a stable system in continuous time is still stable after the discretization, if the sampling rate is chosen properly. This fact is a basis for the analysis presented in this paper.

[2]See, for example, *Nonlinear System Analysis* by M. Vidyasagar, (Prentice-Hall, 1978) for the definition of class K.

$$\begin{aligned}\dot{\boldsymbol{x}}(t,\xi) &\approx \boldsymbol{f}(t, \boldsymbol{x}(t,0), 0) + \xi\left(\frac{\partial \boldsymbol{f}(t, \boldsymbol{x}(t,0), 0)}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}(t,0)}{\partial \xi} + \frac{\partial \boldsymbol{f}(t, \boldsymbol{x}(t,0), 0)}{\partial \xi}\right) \\ &= \boldsymbol{f}(t, \boldsymbol{x}(t,0), 0) + \xi \boldsymbol{g}(t,\boldsymbol{x}) \\ \text{where } \boldsymbol{g}(t,\boldsymbol{x}) &\equiv \frac{\partial \boldsymbol{f}(t, \boldsymbol{x}(t,0), 0)}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}(t,0)}{\partial \xi} + \frac{\partial \boldsymbol{f}(t, \boldsymbol{x}(t,0), 0)}{\partial \xi} \\ \frac{\partial \boldsymbol{f}(t, \boldsymbol{x}(t,0), 0)}{\partial \xi} &\equiv \frac{\partial \boldsymbol{f}(t, \boldsymbol{x}(t,\xi), \xi)}{\partial \xi} \bigg|_{\xi=0}, \text{and} \frac{\partial \boldsymbol{x}(t,0)}{\partial \xi} \equiv \frac{\partial \boldsymbol{x}(t,\xi)}{\partial \xi} \bigg|_{\xi=0}.\end{aligned} \tag{3}$$

C. What about Designing a Controller with an Assumed Maximum Value of ξ?

Generally speaking, there are two scheduling schemes in implementing control algorithms, as shown in Fig. 1. In both schemes, the computing time delay is present. As an alternative, the schedule may also be arranged as 1) sample system outputs, 2) compute the control action, 3) output the control signal to actuators, 4) update the state estimate and/or control parameters, and 5) wait for the next sampling interval and repeat. Clearly, at Step 2, the output is computed based on the control parameters which were updated in the previous sampling interval. That is, the control parameters updated at Step 4 of the current sample interval will affect the control output at Step 3 of the next sampling. Therefore, even the time consumed at Steps 1 and 2 are very short, the computing time at Step 4 will still affect the delay in the loop.

For a stand-alone control system with a simple control algorithm, using a dedicated computer, and without processing exceptions, it may be not difficult to estimate the interval which ξ lies in. Therefore, when designing a controller, one may attempt to handle the computing time delay by using an assumed maximum value of ξ. As was discussed in [9], however, except those systems without using exceptions and having relative simple control algorithms, it is usually impossible to get a precise value of ξ due to the randomness in executing data-dependent branches and loops and sharing resources during the execution of control programs. Moreover, in what follows, we will show that this kind of impreciseness in ξ may fail any attempt of using an assumed maximum value. Suppose the controlled plant is described by $G_0(s) = G_1(s)e^{-ds}$, where $d \geq 0$ is the system time delay and not necessarily an integral multiple of the sampling interval. Suppose the computing time delay is $\xi = \xi_0 T_s, 0 < \xi_0 < 1$, then the equivalent controlled plant is approximated by

$$G(s) = G_1(s)e^{-Ts}, T \equiv d + \xi.$$

Let $d = L_0 T_s - d_0 T_s$ for some $L_0, 0 \leq d_0 < 1$, then

$$T = L_0 T_s + (\xi_0 - d_0)T_s \equiv LT_s - mT_s, 0 < m < 1$$

$$\text{where} \begin{cases} L = L_0 + 1 & \text{and } m = 1 - (\xi_0 - d_0), \quad \text{if } \xi_0 > d_0 \\ L = L_0 & \text{and } m = \xi_0 - d_0, \quad \text{if } \xi_0 \leq d_0. \end{cases}$$

Now, the controlled plant can be represented as

$$G(s) = e^{-LT_s s} G_1(s) e^{mT_s s}.$$

Let $G(z)$ be the controlled plant $G(s)$ in discrete-time domain, then by the hold equivalent (zero-order hold), we get

$$G(z) = (1 - z^{-1})z^{-L}\mathcal{Z}\left\{\frac{G(s)e^{mT_s s}}{s}\right\}$$

where $\mathcal{Z}\left\{\frac{G(s)e^{mT_s s}}{s}\right\}$ represents the z−transform of the time-domain function of $\frac{G(s)e^{mT_s s}}{s}$. Therefore, the computing time delay will affect the open-loop zeros, poles at the origin, and the gain. This can be verified by the following simple example.

Example: Suppose $G(s) = \frac{e^{-1.5s}}{1+s}, T_s = 1$. By the hold equivalent, the controlled plant in discrete-time domain is

$$G_{0.6}(z) = 0.5934\frac{z + 0.0652}{z^3(z - 0.3679)}, \text{if } \xi = 0.6, \text{or} \quad (6)$$

$$G_{0.3}(z) = 0.1813\frac{z + 2.4872}{z^2(z - 0.3679)}, \text{if } \xi = 0.3. \quad (7)$$

Obviously, a controller designed for (6) by assuming a maximum computing time delay may not work well for (7).

D. How Does the Loss Problem Affect System Performance?

The loss problem is quite different from the delay problem. Let k denote a discrete-time index, and suppose at time k the computer controller fails to update the control output. Because of the D/A converter and the hold circuit, the output of the computer controller does not change when a loss problem occurs; that is, $\boldsymbol{u}(k-1)$ is used over two sampling intervals instead of one sampling interval. For example, loss of one controller's output at time k for the aircraft landing problem in [8] keeps the elevator deflection unchanged, but the aircraft does not remain at the same position. At time $k+1$ the controller computer collects a new sample $\boldsymbol{y}(k+1)$ and calculates the corresponding control output $\boldsymbol{u}(k+1)$. Thus, loss of one control output is equivalent to the case when the controller computer fails to update the output during any one sampling interval over the entire mission lifetime. Let $\Delta\boldsymbol{u}(k) \equiv \boldsymbol{u}(k) - \boldsymbol{u}(k-1)$, then $-\Delta\boldsymbol{u}(k)$ can be treated as a disturbance added to the system control input at time k. Because this could occur randomly at any time during the mission, the failure to deliver a control output can be treated as a random disturbance to the system. If the computer again fails to deliver a control output at time $k+1$, then the actual control is still the same as $\boldsymbol{u}(k-1)$. Let $\Delta\boldsymbol{u}(k+1) \equiv \boldsymbol{u}(k+1) - \boldsymbol{u}(k-1)$ and suppose $\boldsymbol{E}[\Delta\boldsymbol{u}(k)] = \text{o}$. Since the correlation matrix $\boldsymbol{E}[\Delta\boldsymbol{u}(k)\Delta\boldsymbol{u}^T(k+1)]$ may not necessarily be zero, this may be a correlated random disturbance.

For all but simple systems, it is difficult to accurately analyze the effects of computing time delay for the following reasons:

1) It is not easy to derive the allowed state space $\boldsymbol{X_A}$, as pointed out in [8].
2) The computing time delay is a random variable, and its effects may change throughout the entire mission lifetime.
3) Different control systems have different structures, thus requiring a separate analysis for each control system. In other words, only case-by-case analyses are possible.

To be more specific on the points discussed thus far, we will analyze the computing time delay effects for a typical real-time control system in the next section.

III. EXAMPLE: A ROBOT CONTROL SYSTEM

The generic analysis of computing time delay effects can only be solidified with real control systems, because the computing time delay is an application-sensitive measure. In this section we will therefore analyze a robot control system in detail.

To analyze a robot control system under a permanently acting perturbation, we need to know the system dynamic equation and the control algorithm to be used. The dynamics of a robot arm can be written as

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{8}$$

where $\boldsymbol{H}(\boldsymbol{q})$ is the $n \times n$ inertia matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$, an vector representing the centrifugal and Coriolis forces, $\boldsymbol{G}(\boldsymbol{q})$, the gravitational loading vector, $\boldsymbol{q}$, the vector of joint position, and $\boldsymbol{\tau}$, the vector of torque/force exerted by joint actuators [1].

We will analyze the robot control system both qualitatively and quantitatively. The qualitative analysis is based on the generic stability criterion (4). For the quantitative analysis, we will demonstrate how the effects of computing time delay can be analyzed and evaluated in practice.

A. Qualitative Analysis

Let the controller be represented by $\boldsymbol{\tau} = \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}},\xi)$. Note that the controller is originally designed by assuming $\xi = 0$. By letting $\boldsymbol{x}_1 \equiv \boldsymbol{q}$ and $\boldsymbol{x}_2 \equiv \dot{\boldsymbol{q}}$, one can describe the closed-loop system by

$$\dot{\boldsymbol{x}}(t,\xi) = \boldsymbol{f}(t,\boldsymbol{x}(t,\xi),\xi) \tag{9}$$

where

$$\boldsymbol{x} \equiv \left[\boldsymbol{x}_1^T, \boldsymbol{x}_2^T\right]^T$$

and

$$\boldsymbol{f}(t,\boldsymbol{x}(t,\xi),\xi) \equiv \begin{bmatrix} \boldsymbol{x}_2 \\ \boldsymbol{H}(\boldsymbol{x}_1)^{-1}(-\boldsymbol{C}(\boldsymbol{x}_1,\boldsymbol{x}_2)\boldsymbol{x}_2 - \boldsymbol{G}(\boldsymbol{x}_1) + \boldsymbol{c}(\boldsymbol{x}_1,\boldsymbol{x}_2,\xi)) \end{bmatrix}.$$

For qualitative analysis, referring to (3), we get the standard form of system with a permanently acting perturbation as

$$\dot{\boldsymbol{x}}(t,\xi) \approx \boldsymbol{f}(t,\boldsymbol{x}(t,0),0) + \xi\boldsymbol{g}(t,\boldsymbol{x}).$$

From (9), we get

$$\boldsymbol{g}(t,\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial \boldsymbol{x}_2(t,0)}{\partial \xi} \\ \boldsymbol{a}_{21}\dfrac{\partial \boldsymbol{x}_1(t,0)}{\partial \xi} + \boldsymbol{a}_{22}\dfrac{\partial \boldsymbol{x}_2(t,0)}{\partial \xi} + \boldsymbol{H}(\boldsymbol{x}_1)^{-1}\dfrac{\partial \boldsymbol{c}(\boldsymbol{x}_1,\boldsymbol{x}_2,0)}{\partial \xi} \end{bmatrix}$$

where

$$\boldsymbol{a}_{21} = \frac{\partial}{\partial \boldsymbol{x}_1}\left[\boldsymbol{H}(\boldsymbol{x}_1)^{-1}(-\boldsymbol{C}(\boldsymbol{x}_1,\boldsymbol{x}_2)\boldsymbol{x}_2 - \boldsymbol{G}(\boldsymbol{x}_1) + \boldsymbol{c}(\boldsymbol{x}_1,\boldsymbol{x}_2,0))\right]$$

$$\boldsymbol{a}_{22} = \boldsymbol{H}(\boldsymbol{x}_1)^{-1}\frac{\partial}{\partial \boldsymbol{x}_2}(-\boldsymbol{C}(\boldsymbol{x}_1,\boldsymbol{x}_2)\boldsymbol{x}_2 + \boldsymbol{c}(\boldsymbol{x}_1,\boldsymbol{x}_2,0)).$$

Because each term of $\boldsymbol{g}(t,\boldsymbol{x})$ is bounded above in t, $\sup_{t\geq t_0, \|\boldsymbol{x}\|<\rho} \| \boldsymbol{g}(t,\boldsymbol{x}) \| < \infty$, that is, stability condition (4) is satisfied. Thus, we conclude that there exists a $0 < \xi < T_s$ small enough such that the closed-loop system is uniformly stable.

B. Quantitative Analysis with Respect to System Stability

For any quantitative analysis of the effects of computing time delay, it is necessary to specify the control algorithm to be used. In the robot control system, the controller is $\boldsymbol{\tau} = \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}},0)$. If the system parameters in (8) are known, then one can choose the control torque/force vector as

$$\boldsymbol{\tau} = \boldsymbol{H}(\boldsymbol{q})\boldsymbol{u}(t) + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) \tag{10}$$

$$\boldsymbol{u}(t) = \ddot{\boldsymbol{q}}_d(t) - \boldsymbol{K}_v(\dot{\boldsymbol{q}}(t) - \dot{\boldsymbol{q}}_d(t)) - \boldsymbol{K}_p(\boldsymbol{q}(t) - \boldsymbol{q}_d(t)) \tag{11}$$

where $\boldsymbol{q}_d$ is the desired joint positions and $\boldsymbol{K}_v, \boldsymbol{K}_p$ the matrices of controller gains. This is the well-known computed torque algorithm, on which several adaptive algorithms (for example, [7]) are proposed in the presence of unknown parameters. Taking Laplace transform of (11), we get

$$\boldsymbol{U}(s) = s^2\boldsymbol{Q}_d(s) - (\boldsymbol{K}_v s + \boldsymbol{K}_p)\Delta(s) \tag{12}$$

where $\boldsymbol{U}(s), \boldsymbol{Q}_d(s)$ and $\Delta(s)$ are the Laplace transforms of $\boldsymbol{u}(t), \boldsymbol{q}_d(t)$ and $\delta(t) = \boldsymbol{q}(t) - \boldsymbol{q}_d(t)$, respectively. Plugging (10) and (11) into (8), taking Laplace transform, and noting that $\boldsymbol{H}(\boldsymbol{q})$ is nonsingular, we get

$$\left(s^2\boldsymbol{I} + (s\boldsymbol{K}_v + \boldsymbol{K}_p)\right)\Delta(s) = \mathbf{o}, \text{that is,} s^2\boldsymbol{I} + s\boldsymbol{K}_v + \boldsymbol{K}_p = \mathbf{o}. \tag{13}$$

If there is a nonzero computing time delay, that is, the control output at time t is computed based on the sample at time $t - \xi$, $\delta(t)$ must be replaced by $\delta(t - \xi)$. When the controller is actually implemented on a digital computer, the reference input $\ddot{\boldsymbol{q}}_d(t)$ does not change during one sampling interval, and thus, (12) and (13) become

$$\boldsymbol{U}(s,\xi) = s^2\boldsymbol{Q}_d(s) - (s\boldsymbol{K}_v + \boldsymbol{K}_p)e^{-s\xi}\Delta(s), \quad \text{and}$$
$$\left(s^2\boldsymbol{I} + (s\boldsymbol{K}_v + \boldsymbol{K}_p)e^{-s\xi}\right)\Delta(s) = \mathbf{o}.$$

Because $\xi < T_s$ and T_s is small enough to recover the continuous-time signals, one can approximate $e^{-s\xi} \approx 1 - \xi s$ to get

$$s^2(\boldsymbol{I} - \boldsymbol{K}_v\xi) + s(\boldsymbol{K}_v - \boldsymbol{K}_p\xi) + \boldsymbol{K}_p = \mathbf{o}. \tag{14}$$

One may choose $\boldsymbol{K}_v = diag[K_{vi}]$,$\boldsymbol{K}_p =$diag[K$_{\text{pi}}$], $K_{vi}, K_{pi} > 0, i = 1, 2, \cdots, n$, such that the closed-loop system becomes uncoupled, linear, and exponentially stable. Then, (15) becomes

$$s^2(1 - K_{vi}\xi) + s(K_{vi} - K_{pi}\xi) + K_{pi} = 0, i = 1, 2, \cdots, n. \tag{15}$$

We can derive from (15) the least upper bound of ξ for system stability

$$\xi < \inf_i \left(\min\left\{\frac{K_{vi}}{K_{pi}}, \frac{1}{K_{vi}}\right\}\right), i = 1, 2, \cdots, n. \tag{16}$$

This upper bound of ξ can be viewed as a CSD with respect to system stability and used as an extra constraint when selecting controller gains.

C. Quantitative Analysis with Respect to Real-Time Performance

In practice, we not only need to know the least upper bound of ξ with respect to system stability, but also the quantitative performance changes caused by it. As stated before, the controller is designed under the assumption of $\xi = 0$. If the controller gains are diagonal matrices, then (13) becomes

$$s^2 + K_{vi}s + K_{pi} = 0, i = 1, 2, \cdots, n. \tag{17}$$

Comparing this with the standard form of a second order system $s^2 + 2\zeta_i\omega_{ni}s + \omega_{ni}^2 = 0$, we choose

$$K_{vi} = 2\zeta_i\omega_{ni}, K_{pi} = \omega_{ni}^2, i = 1, 2, \cdots, n \tag{18}$$

where ζ_i and ω_{ni} are the closed-loop damping ratio and the natural frequency of subsystem i, respectively.

We consider the relative displacement of closed-loop system poles from their originally-designed positions as a result of the nonzero computing time delay. Let s_a be the actual pole position which is moved from its desired position s_d due to the presence of $\xi > 0$. s_d and s_a are given by (17) and (15), respectively, as

$$\begin{aligned} s_d &= -\frac{1}{2}K_{vi} \pm j\frac{1}{2}\sqrt{4K_{pi} - K_{vi}^2}, \text{and} \\ s_a &= \\ &\frac{-(K_{vi} - K_{pi}\xi) \pm j\sqrt{4(1 - K_{vi}\xi)K_{pi} - (K_{vi} - K_{pi}\xi)^2}}{2(1 - K_{vi}\xi)}. \end{aligned}$$

We define the maximum relative displacement of poles as the performance tolerance

$$\alpha = \max_i \frac{| \sigma_{di} - \sigma_{ai} |}{\sigma_{di}} \tag{19}$$

$$\text{where} \quad \sigma_{di} = \| s_d \|_2 = \sqrt{K_{pi}}, \quad \text{and}$$

$$\sigma_{ai} = \| s_a \|_2 = \sqrt{\frac{K_{pi}}{1 - K_{vi}\xi}}, \sigma_{ai} \geq \sigma_{di}.$$

Therefore, from (18) and (19), if the performance tolerance α is specified, we get

$$\xi < \begin{cases} \max_i \left(\dfrac{1}{2\zeta_i\omega_{ni}} \left| 1 - \dfrac{1}{(1-\alpha)^2} \right| \right) & \text{if } \alpha \neq 1 \\ \max_i \left(\dfrac{3}{8\zeta_i\omega_{ni}} \right) & \text{if } \alpha = 1. \end{cases} \tag{20}$$

This is the upper bound of ξ with respect to the performance tolerance for the computed torque algorithm. It can be viewed as a CSD with respect to system performance, and used as an index to select a microprocessor to implement the control algorithm.

The computed torque algorithm (10) is based on the assumption that the parameters of (8) are completely known. They are in fact unknown, however, or not known precisely. Some adaptive schemes may be used to estimate these parameters and then implement the computed torque algorithm or variations thereof [7]. If it is assumed that the estimators give the true values and a perfect tracking of time-varying parameters, then performance changes can be analyzed separately while figuring the time consumed by the estimators in the computing time delay. This implies that (16) and (20) can also be used as an approximate analysis for adaptive control methods which are based on the computed torque algorithm. Since the parameter estimators and the adaptive algorithm require a longer computing time delay than nonadaptive ones, it is important to analyze the effects of computing time delay.

Thus far, we have analyzed a typical robot control system in terms of the computing time delay. Note that in this example, both the system dynamic equation and the control algorithm are provided in closed form. There are many complex control systems, however, for which the system dynamic equations are either unknown or known only qualitatively. For such a system, one cannot describe the control actions in closed-form. This implies that the effects of computing time delay in such complex systems need to be evaluated with simulations and/or experiments.

IV. Conclusion

The increasing use of digital computers to implement real-time controllers has made it essential to carefully study the effects of computing time delay on the stability and performance of controlled systems. This computing time delay is different from the usual system time delay; it is a random delay resulting from the execution of control programs on a digital computer.

Effects of the computing time delay on control system performance are classified into the delay and loss problems, which are then analyzed for both general and special cases. A generic criterion is derived for the qualitative analysis of the delay problem. Since any quantitative analysis requires the detailed knowledge of the controlled system and the control algorithm to be used, we have chosen a prototypical real-time control system with a commonly-used control algorithms—a robot control system with the computed torque algorithm—to give a detailed account of computing time delay effects. For such a system, upper bounds of computing time delay for system stability and performance are derived as an extra constraint on controller design and microprocessor selection for control algorithms. Both the qualitative and quantitative analyses of the robot control system have demonstrated how system performance is affected by the computing time delay and how a given system can be evaluated based on the computing time delay.

References

[1] H. Asada and J.-J. E. Slotine, *Robot Analysis and Control.* New York: Wiley, 1986.

[2] D. S. Bernstein, L. D. Davis, and S. W. Greeley, "The optimal projection equations for fixed-order sampled-data dynamic compensation with computation delay," *IEEE Trans. Automat. Contr.*, vol. AC-31, no. 9, pp. 859–862, Sept. 1986.

[3] X. Cui and K. G. Shin, "Robot Trajectory Tracking with Self-Tuning Predicted Control," in *Proc. 1991 Amer. Contr. Conf.* vol. 1, pp. 529–534.

[4] P. K. Khosla, "Choosing Sampling Rates for Robot Control," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1987, pp. 169–174.

[5] C. S. G. Lee and B. H. Lee "Resolved motion adaptive control for mechanical manipulators," *ASME J. Dynamics, Measurement Contr.*, vol. 106, no. 2, pp. 134–142, June 1984.

[6] T. Mita, "Optimal digital feedback control systems counting computation time of control laws," *IEEE Trans. Automat. Contr.*, vol. AC-30, no. 6, pp. 542–547, June 1985.
[7] R. Ortega and M. W. Spong, "Adaptive Motion Control of Rigid Robots: A Tutorial," in *Proc. 27th Conf. Decision Contr.*, 1988, pp. 1575–1584.
[8] K. G. Shin, C. M. Krishna, and Y.- H. Lee, "A unified method for evaluating real-time computer controller and its application," *IEEE Trans. Automat. Contr.*, vol. AC-30, no. 4, pp. 357–366, Apr. 1985.
[9] M. H. Woodbury, "Analysis of Execution Time of Real-Time Tasks," in *Proc. 1986 IEEE Real-Time Syst. Symp.*, pp. 89–96.