International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:6, 2008

Delay Analysis of Sampled-Data Systems in Hard RTOS

A. M. Azad, M. Alam, and C. M. Hussain

Abstract**—** In this paper, we have presented the effect of varying time-delays on performance and stability in the single-channel multi-rate sampled-data system in hard real-time (RT-Linux) environment. The sampling task require response time that might exceed the capacity of RT-Linux. So a straight implementation with RT-Linux is not feasible, because of the latency of the systems and hence, sampling period should be less to handle this task. The best sampling rate is chosen for the sampled-data system, which is the slowest rate meets all performance requirements. RT-Linux is consistent with its specifications and the resolution of the real-time is considered 0.01 seconds to achieve an efficient result. The test results of our laboratory experiment shows that the multi-rate control technique in hard real-time operating system (RTOS) can improve the stability problem caused by the random access delays and asynchronization.

Keywords**—**Multi-rate, PID, RT-Linux, Sampled-data, Servo.

I. Introduction

THE increasing use of digital computers is one of the most important developments in high-end embedded systems in recent years [1, 2, 3, 4, 5, 6, 8, 9]. In the real-time servo computer control loop, the controlled variable is sampled (A/D) at a suitable constant based sampling-rate (10 milliseconds in our case) and PID control algorithm uses this information to compute a new control action, which is reconstructed (D/A) at a faster rate (integer multiple of base sampling rate) and applied to the continuous servo plant. Complexity arises when the random delays are inserted into the control loop. One access delay is inserted between the sampler (A/D) and the controller and the other one is inserted between the controller and the hold (D/A) using real-time (RT-Linux) thread program.

Systems performance can be adversely affected by the presence of random access delays in the closed loop path. In fact, small or large access delays and their associated phase lag provide unwanted oscillations in the closed-loop response, which may even lead to instability. The reasons of such delays can be due to the sensor or actuator characteristics, insufficient processing speed, or communication lags etc. The varying time delay in computing and transmitting the control output and its negative effects on real-time control system are classified into the delay and loss problem. The nonzero varying time delay shorter than the sampling interval yields the delay problem and the loss of control output to the actuator occurs when the varying time delay is equal or longer than the sampling interval.

We implemented a PID controller algorithm in the RT-Linux environment for multi-rate sampling systems and thus synthesize with known access delays that assure stability and minimize system performance degradation. Our experiment shows that the effects of delays, even of considerable length in relation to the sample interval, can be effectively countered under real-time implementation. The best sampling rates selection for sampled-data systems comprises many factors. The basic motivation is to minimize the cost function. Fast sampling rate in comparison to slow sampling rate, improves command signal tracking effectiveness, and decreases sensitivity to random plants disturbances, plant parameter variations, and measurement noise.

II. Multi-Rate Sampled-Data (MRSD) Systems in Discrete-Time Domain

A Nyquist reconstruction theorem states that if the signal is of limited bandwidth then the original signal can be reconstructed exactly from the sampled signal if the sampling rate is higher than twice the bandwidth of the signal. But it is harder to analyze multi-rate sampled-data systems where different samplers and holds work at different sampling rates. The system is periodic provided the sampling rates are related by rational numbers and it can be transformed to fit the standard single rate using lifting technique [7].

The singled channel sampled-data feedback system is considered in this experimental test. The continuous-time plant G_c is described as follows [7]:

$$\begin{cases} \dot{x}(t) = Ax(t) + B_w w(t) + B_{u_c} u_c(t) \\ y_c(t) = C_{y_c} x(t) + D_{y_c w} w(t) \\ z(t) = C_z x(t) + D_{zu_c} u_c(t) \end{cases}$$

For the sampled-data controller problem, the equivalent discrete-time representations of the plant dynamics and averaged measurements are as follows:

A. M. Azad is with the Dept. of Computer Science and Engineering, BRAC University, Mohakhali, Dhaka-1212, Bangladesh (phone: 8802-8824051 Ext. 4024; fax: 8802-8810383; e-mail: a.azad@bracu.ac.bd).

M. Alam, is with the Dept. of CSE, BRAC University, Mohakhali, Dhaka-1212, Bangladesh (e-mail: u05310034@student.bu.ac.bd).

C. M. Hussain is with the Dept. of CSE, BRAC University, Mohakhali, Dhaka-1212, Bangladesh (e-mail: cmih_brac@yahoo.com).

$$\begin{cases} x^d(k+1) = A^d x^d(k) + w^d(k) + B^d \hat{u}(k) \\ \hat{y}_c(k) = C^d x^d(k) \end{cases}$$

Thus, the equivalent discrete-time cost function for the discrete-time controller, K_d with sampling time *h* is given by,

$$J_{k_{d,h}} = \underset{k \to \infty}{Lt} E\begin{bmatrix} (x^d(k))^T Q_d x^d(k) + \\ 2(x^d(k))^T {}_{Q_d} M_{R_d} u^d(k) + \\ (u^d(k))^T R_d u^d(k) \end{bmatrix} + \eta_h$$

where η_h, the constant offset represent the lower bound on the sampled-data performance. Now, we introduce random access delays in the plant input as well as in the measurement. The resulting sampled-data plant is represented by,

$$\begin{cases} \widetilde{x}_\Delta(k+1) = \widetilde{A}_\Delta \widetilde{x}_\Delta(k) + \widetilde{w}(k) + \widetilde{B}_\Delta u(k) \\ y(k) = \widetilde{C}_\Delta \widetilde{x}_\Delta(k) \end{cases}$$

with the equivalent discrete-time cost function given by,

$$J_{k_{d_\Delta}} = \eta_h + \underset{k \to \infty}{Lt} E\begin{bmatrix} \widetilde{x}_\Delta^T(k) \widetilde{Q}_d \widetilde{x}_\Delta(k) + \\ 2(\widetilde{x}_\Delta(k))^T {}_{\widetilde{Q}_d} M_{R_d} u^d(k) + \\ (u^d(k))^T R_d u^d(k) \end{bmatrix}.$$

For small access delay $(\Delta \le h)$,

$$\widetilde{A}_\Delta = \begin{bmatrix} A^d & B^d \\ C^d & 0 \end{bmatrix}, \quad \widetilde{B}_\Delta = \begin{bmatrix} B^d \\ 1 \end{bmatrix}, \quad \widetilde{w}(k) = \begin{bmatrix} w^d(k) \\ 0 \end{bmatrix},$$

$$\widetilde{C}_\Delta = \begin{bmatrix} 0 & 1 \end{bmatrix}, \qquad \widetilde{Q}_d := \begin{bmatrix} Q_d & 0 \\ 0 & 0 \end{bmatrix} \text{ and}$$

$$\widetilde{Q}_d M_{R_d} := \begin{bmatrix} {}_{Q_d} M_{R_d} \\ 0 \end{bmatrix}.$$

Large access delay $(\Delta > h)$, where plant can be modeled by adding a delay block consisting of k delays, each of *h* seconds, then

$$\widetilde{A}_\Delta = \begin{bmatrix} A^d & B^d & 0 \\ C^d & 0 & I_{k-1} \\ 0 & I_{k-1} & 0 \end{bmatrix}, \widetilde{B}_\Delta = \begin{bmatrix} B^d \\ 0 \\ 1 \end{bmatrix},$$

$$\widetilde{w}(k) = \begin{bmatrix} w^d(k) \\ 0 \\ 0 \end{bmatrix}, \quad \widetilde{C}_\Delta = \begin{bmatrix} C^d & 0 & 1 \end{bmatrix},$$

$$\widetilde{Q}_d := \begin{bmatrix} Q_d & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } {}_{\widetilde{Q}_d} M_{R_d} := \begin{bmatrix} {}_{Q_d} M_{R_d} \\ 0 \\ 0 \end{bmatrix}.$$

Now, we consider the problem of obtaining a strictly proper discrete-time dynamic controller for the sampled-data systems with random access delays. The optimal discrete-time LQG controller (strictly proper) with sampling time *h* is given by [8, 9],

$$\begin{cases} x_{k_d}(k+1) = A_{k_d} x_{k_d}(k) + B_{k_d} \hat{y}_c(k) \\ \hat{u}(k) = C_{k_d} x_{k_d}(k) + D_{k_d} \hat{y}(k) \end{cases}$$

where, $A_{k_d} = \widetilde{A}_\Delta + \widetilde{B}_\Delta C_{k_d} - B_{k_d} \widetilde{C}_\Delta$,

$$B_{k_d} = \widetilde{A}_\Delta S_{d_1} \widetilde{C}_\Delta^T (\widetilde{C}_\Delta S_{d_1} \widetilde{C}_\Delta^T)^{-1}$$

and $C_{k_d} = -(R_d + \widetilde{B}_\Delta^T S_{d_2} \widetilde{B}_\Delta)^{-1} (\widetilde{B}_\Delta^T S_{d_2} \widetilde{A}_\Delta + {}_{\widetilde{Q}_d} M_{R_d}^T)$.

The positive semi-definite matrices S_{d_1} and S_{d_2} are the solutions to the following DREs:

$$S_{d_1} = [\widetilde{A}_\Delta S_{d_1} \widetilde{A}_\Delta^T - \widetilde{A}_\Delta S_{d_1} \widetilde{C}_\Delta^T (\widetilde{C}_\Delta S_{d_1} \widetilde{C}_\Delta^T)^{-1} \widetilde{C}_\Delta S_{d_1} \widetilde{A}_\Delta^T + V_n]$$

$$S_{d_2} = [-(\widetilde{A}_\Delta^T S_{d_2} \widetilde{B}_\Delta + {}_{\widetilde{Q}_d} M_{R_d})(R_d + \widetilde{B}_\Delta^T S_{d_2} \widetilde{B}_\Delta)^{-1} (\widetilde{B}_\Delta^T S_{d_2} \widetilde{A}_\Delta + {}_{\widetilde{Q}_d} M_{R_d}^T) + \widetilde{A}_\Delta^T S_{d_2} \widetilde{A}_\Delta + \widetilde{Q}_d]$$

Where, the nonnegative definite non-homogeneous term V_n is the measurement noise covariance [10]. The optimal cost of the above sampled-data controller is given by,

$$J_{k_d}^{opt} = \eta_h + tr\begin{bmatrix} S_{d_1} \widetilde{Q}_d + \hat{S}_{d_1} (\widetilde{Q}_d + 2C_{k_d}^T {}_{\widetilde{Q}_d} M_{R_d} + \\ C_{k_d}^T R_d C_{k_d}) \end{bmatrix}$$

where $\hat{S}_{d_1}$ is the solution to the following equation,

$$\hat{S}_{d_1} = [(\widetilde{A}_\Delta + \widetilde{B}_\Delta C_{k_d})\hat{S}_{d_1}(\widetilde{A}_\Delta + \widetilde{B}_\Delta C_{k_d})^T + \widetilde{A}_\Delta S_{d_1}\widetilde{C}_\Delta^T(\widetilde{C}_\Delta S_{d_1}\widetilde{C}_\Delta^T)^{-1}\widetilde{C}_\Delta S_{d_1}\widetilde{A}_\Delta^T]$$

This is especially for the more restrictive problem in which the controller is strictly proper (i.e., $D_{k_d} = 0$). Thus, the sampled-data equivalent discrete-time problem provides an optimal LQG-type sampled-data regulator, which explicitly accounts for random access delays.

A. PID Controller Derived from the LQG Regulator for the 2nd Order Servo Systems in Frequency Domain

A Proportional-Integral controller and state estimation provide the elements of a classical Proportional-Integral-Derivative (PID) controller for a second-order servo system with a single command input, a single measurement, and a single control [10].

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -\omega_n^2 & -2\xi\omega_n\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \begin{bmatrix}0\\ \omega_n^2\end{bmatrix}u(t)$$

$$\bar{y}(t) = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\bar{x}_1(t)\\ \bar{x}_2(t)\end{bmatrix} = x_1(t),$$

$$z(t) = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} = x_1(t)$$

where, command input vector, $\bar{y}$, represents a desired value of an output vector *y*. It is assumed that disturbances are unmeasured and that the Kalman-Bucy filter optimally estimates the state's perturbation from the set point $[x(t) - \bar{x}(t)]$ by integrating [10],

$$\begin{bmatrix}\dot{\hat{x}}_1(t)\\ \dot{\hat{x}}_2(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -\omega_n^2 & -2\xi\omega_n\end{bmatrix}\begin{bmatrix}\hat{x}_1(t)\\ \hat{x}_2(t)\end{bmatrix} + \begin{bmatrix}0\\ \omega_n^2\end{bmatrix}u(t) + \begin{bmatrix}K_1\\ K_2\end{bmatrix}[z(t) - \hat{x}_1(t)]$$

where, K_1 and K_2 are optimal Kalman filter gain. The filter generates an estimate of the state, $\hat{x}_1$ as well, and it provides feedback information for PI control law. Ignoring the initial condition of the integrator, the control signal is

$$u(t) = C_F\bar{y}(t) + C_I\int_0^t[\bar{y}(t) - \hat{x}_1(t)]dt - C_1\hat{x}_1(t) - C_2\hat{x}_2(t)$$

where, C_F, C_I are simply algebric functions of the set-point sensitivities and the gain matrix *C*. Thus, the Laplace transformation of the control signal is,

$$u(s) = C_F\bar{y}(s) + \frac{C_I[\bar{y}(s) - \hat{x}_1(s)]}{s} - (C_1 + C_2 s)\hat{x}_1(s)$$

$$= (C_F + \frac{C_I}{s})\bar{y}(s) - (C_1 + \frac{C_I}{s} + C_2 s)\hat{x}_1(s)$$

The state estimator introduces dynamics between the measurement *z* and the estimate of the state $\hat{x}_1$, but the PID characteristic contained in $(C_1 + \frac{C_I}{s} + C_2 s)$ is clear. This characteristic would be retained even if there were no estimator (i.e., if x_1 and x_2 were measured and fed back directly).

III. Servo System in Hard Real-Time

A Real Time System is an information system whose correction does not only depend on the logical output of the algorithms but also on the moment in time when these output occurred. Thus, a real time system is not necessarily fast, but must be accurate in time. The design of a real-time system goes through several phases: First the tasks to be performed and the temporal restrictions that must be satisfied are identified, Secondly the code is written and Finally the run-time of each task is measured and a schedulability test is done to ensure that the task will not miss its deadline while the system is running. Real-time is divided into two areas: hard and soft real-time. We will concentrate our discussion to hard real-time systems.

There is a wide range of real-time operating systems available in the market to support any specific real-time performance required by the control applications. The real-time performance is limited to the low-level interaction with the hardware. lack of synchronization and random delays appears in a natural way in Industrial distributed computing and parallel processing, where shared communication is used with a lot of sensors and actuators. In the servo experiment, random delays are generated by means of real-time thread program instead of using real industrial multi-processor systems. The real-time systems is implemented with a combination of Linux, RT-Linux, data acquisition card, servo systems, availability of source code and a standard PC.

A. Hardware and Software Structure

The servo experiment under RT-Linux environment included a standard Pentium laboratory PC with the feedback modules and ax5411 data acquisition card. We have loaded red hat Linux 7.0 and the basic system came up and used Vxscope for plotting the systems response. RT-Linux comes as a set of different files from the source version 2.2.18. From the software point of view, a real-time thread program is developed. It caught an interrupt from a clock, latched the interrupts, read the clock, waited the right amount of time for

the encoder latch to settle and read the position. With the position known, it called the PID routine to determine the output voltage to drive the motor. The clock interrupt was arbitrarily set to 10*msec*. Among the two important real-time issues, one is the ability to latch the encoders in the interrupt service routine and the other one is the ability to compute the response within a reasonable period of time (10 milliseconds) so that no interrupt is missed. Real-time data acquisition framework of multi-rate sampled-data system is presented in RT-Linux environment. The RT-Linux facilities for task handling are basic. There is rt_task_init(), which creates and starts a task. The stack size and priority can be specified. Linux itself is run as a real-time task with the lowest priority. The task is set up to run at periodic intervals by rt_task_make_periodic(). The rt_task_wait() facility blocks the calling task. The tasks are run using a simple preemptive scheduler.

The primary means of communication between the real-time tasks and the Linux processes is the FIFO. The rtf_create() facility creates a FIFO of a desired size. Data is enqueued onto the FIFO by rtf_put(), returning an error if the FIFO is full. Similarly, rtf_get() dequeues data from the FIFO, returning an error if the FIFO is empty. The most obvious use for this FIFO scheme is data streaming. In a data acquisition application, for example, a real-time task could be set up using rt_task_init() and rt_task_make_periodic() to acquire samples from an I/O board at fixed intervals. This task would send its data to a Linux process using rtf_put() . The Linux process would be in a loop, reading data from the FIFO and perhaps writing the data to disk, sending it over a network, or displaying it in an X-window. The FIFO would serve as a buffer, so the Linux process could operate without real-time constraints.

A simple real-time kernel with a priority scheduling scheme was implemented for the data acquisition card with different sampling and hold rate. Tasks are assigned to each module during the initialization step. The flow of data to and from the data acquisition card is implemented with real-time tasks through real-time FIFO. A preemptive scheme was implemented using interrupt-based techniques to handle three real-time tasks. The inter-task communication among the three real-time tasks is done with shared memory. The RT-kernel receives a fixed set of tasks at the time of initialization and each of the tasks has a priority level assigned in the preemptive scheme. The real-time task communicates with non real-time Linux and the data acquisition card and the RT-FIFOs avoid message losses. The non real-time GUI program using GTK (GIMP Toolkit) in the Linux environment provides the high level interfaces between the user and the experiment. The set of tasks (e.g., data logging, display and GUI etc.) assigned to it is non time-critical. In the data acquisition scheme, the real-time tasks delivering to Linux a low rate of results and final data through shared memory and circular buffer. Thus, Linux is slightly loaded saving CPU resources for non time-critical tasks (display, data logging etc.).

B. Real-Time PID Controller

Asynchronization between controller and plant and loss of information due to the random delays are a potential cause of system instability. One possible solution to this problem involves using multi-rate technique and also RT-Linux operating system is needed to perform control task and time management of this kind of systems. In this regard, a multi-rate sampled-data controller can be implemented with different frequencies in the sampler and hold. In fact, a slow frequency is applied to the sampler (A/D) to provide the controller with the necessary information to take its decisions and employ a fast frequency to the hold (D/A) to apply control actions. Thus, in the real-time tasks, the priority of the fast frequency to the 'dtoa' must be higher than the slow frequency to the sampler. In selecting these frequencies RT-Linux resolution 0.01 seconds must be considered so that the fast frequency to be high enough to achieve required control specifications and slow frequency to be low enough to avoid the loss of information due to the random delays. Multi-rate controller generates higher frequency discrete control signal in its output and the lower frequency feedback signal in its input. In PID controller [9], the transfer function of a differentiator is $T_d s$, but this is undesirable because it magnifies any noise (which may be introduced by measurement for example, quantization effects in the A/D converter). Consequently, a differentiator is approximated by $\frac{T_d s}{\alpha s + 1}$, which is a differentiator and a low-pass filter. This approaches the true differentiator as α is made small ($\alpha = 0.1$). The saturation functions ensure that the state space equations are nonlinear. We have to deal with two first-order transfer functions, which will give two states.

The Proportional part of the controller is: $e_d = K_p e$

For the Integrator Part:

$$x_i = \left(\frac{1}{T_i s + 1} \right) sat(e_d + x_i)$$

And the differentiator part:

$$u_d = \frac{T_d s}{\alpha s + 1} e_d = T_d s x_d, \quad \text{where} \quad x_d = \frac{e_d}{\alpha s + 1}$$

$$\therefore \quad s x_d = \frac{1}{\alpha}(e_d - x_d) \quad \text{and} \quad u_d = \frac{1}{\alpha} T_d (e_d - x_d)$$

The final state-space system comprises two state equations:

$$\frac{d}{dt} x_i = \frac{sat(e_d + x_i) - x_i}{T_i} \quad \text{and} \quad \frac{d}{dt} x_d = \frac{1}{\alpha}(e_d - x_d)$$

and an output equation:

$$u = sat\left(sat(e_d + x_i) + \frac{1}{\alpha} T_d (e_d - x_d) \right)$$

The final saturation block gets added just to ensure that even when the output of the differentiator is large, and may be added to an already saturated integration term, the control output remains bounded and has a range suitable for the D/A

converter. In the computer controlled servo system, the motor shaft position, measured with a sensor, is the controlled variable. The above conventional PID controller algorithm has been designed to reach certain closed-loop specifications. In order to obtain a good continuous behavior, a sampling period of *10msec* is considered to be within RT-Linux resolution capacity.

The PID control, from our point of view, is a function that takes a number of parameters -- the position of the servomotor -- and returns the value of the control signal that has to be applied to the servomotor -- the voltage that must be fed to the servomotor. The theory behind the design of PID algorithms, which by the way is extensive, assumes that the computational time is negligible, that is, the time from reading the position of the servomotor until the time when we act is very small. Under normal circumstances systems allow for a small delay time. Another characteristic of this type of control is that it must be executed periodically, in other words, the PID algorithm must be executed regularly. If the time between two consecutive calls to the PID function is too large, then the servomotor may reach an unstable condition.

IV. Results from the Laboratory Test

Initially, we haven't introduced any delays in the system with single-rate (N=1) controller. The step response of the servo system reflects the perfect stability with reasonable peak overshoot and settling time. Then, we introduced small access delay (*5msec*) into the system using RT-Linux real-time thread program and observed the response and found no significant effect on stability and the step response also remains unchanged. Thus, the small access delay with in the boundary doesn't cause any loss of information in the control loop, which is significantly responsible for the instability.

When, we introduced large access delay (*50msec*) into the control loop, it influences the stability in systems response by introducing more oscillations to settle down into steady state (Fig. 1). The peak overshoot increased to 0.9 as well. If the upper limit of the delay is more than the chosen sampling period, some of the samples/holds will be lost during the process. We have considered the upper limit of large access delay 50*msec* in our hard real-time experiment. Thus two to three samples/holds may be lost in each time.

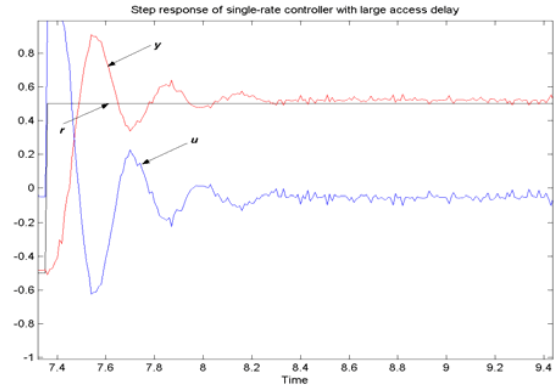

Fig. 1 Response to the step input with large access delay

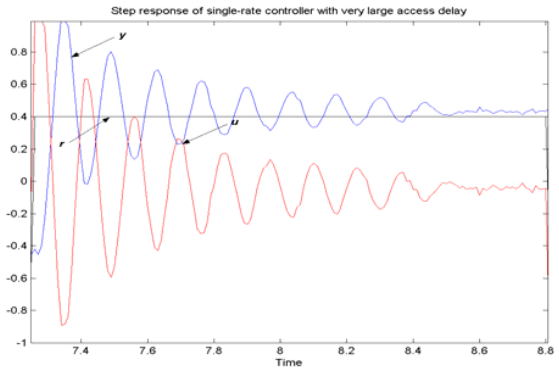

Fig. 2 Response to the step input with very large access delay

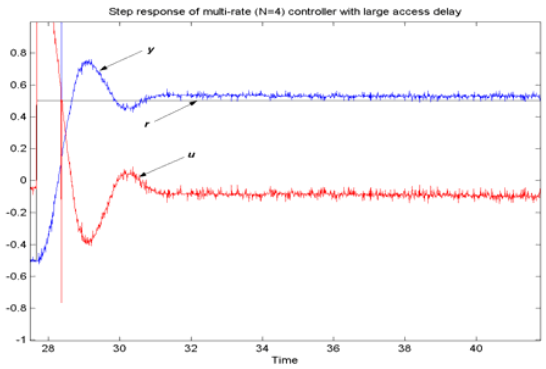

Fig. 3 Response to the step input for multi-rate controller with large access delay

The randomness of very large access delay causes the irregularity in its behavior by losing some samples/holds, which yields potential instability as shown in Fig. 2. The step response (Fig. 2) of the single-rate controller with very large access delay (*1000msec*) in the loop provides very high peak overshoot, more oscillations to settle down and thus threatens stability. Due to the restriction of performance specification, we simply can't increase the sampling period to avoid the loss of information. Paper [1] proposed that with higher priority of hold (D/A) task, the access delay can be reduced till the upper limit of access delay is less than the desired sampling period. Consequently, the priority of sampler task might be reduced and the sampling period of the sampler should be increased to overcome losing samples and so the observed delay is constant which can be compensated. Thus, the frequency of the hold (D/A) is higher than the sampler frequency and so a multi-rate controller is to be employed. Response using multi-rate controller is shown in Fig. 3, where the upper limit of large access delay is considered as 50*msec*. The step response for the multi-rate controller under large access delay with no loss of information and asynchronization is observed. It is clear that the instability influence (Peak overshoot and oscillation) disappeared completely under hard real-time (RT-Linux) operating system.

V. Conclusion

For real-time implementation of multi-rate sampling system, the position control of servo system with real-time PID control algorithm using RT-Linux (hard real-time) thread program is considered. To meet all the required performance for the sampled-data system, the best sampling rate is chosen. The single channel multi-rate controller is composed by a slow sampler (operating at a low frequency) and a fast hold (operating at a high frequency). Each of the slow samples is converted into a group of *N* fast samples (Hold) and thus only one of each group of *N* samples can be used as information feedback to the controller since sensor operates at low frequency. Multi-rate control technique in RT-Linux can solve the stability problems caused by the random access delays and asynchronization. The result using the multi-rate controller is almost similar to the fast single-rate one.

References

[1] V. Casanova & J. Salt, Multirate control for an ICCS environment. Part I: the random access delays, Proceedings of the 2nd IFAC workshop on Linear Time Delay Systems, 2000, Ancona (Italy).
[2] J. Salt & P. Albertos, Multirate controllers design by rate decomposition, Proceedings of the 39th Conference on Decision and Control, 2000, Sydney, Australia.
[3] C. S. Alan, Real-Time Systems and Software, (Wiley, 2001).
[4] B. Wittenmark, B. Bastian & J. Nilsson, Analysis of Time Delays in Synchronous and Asynchronous Control Loops, Proceedings of the 37th Conference on Decision and Control, Tampa, Florida, USA, 1998, pp. 283-288.
[5] K. J. Astrom & B. Wittenmark, Computer-Controlled Systems, (Third edition, Prentice Hall, 1997).
[6] C. K. Chak, G. Feng & T. Hesketh, Multirate adaptive optimal control with application to dc motor, Computers Elect. Engng, Vol. 23, No. 2, 1997, pp. 65-79.
[7] A. M. Azad & T. Hesketh, H -Optimal Control Of Multi-rate Sampled-data Systems, Proceedings of American control Conference, Anchorage, Alaska, USA, 2002, pp.459-464.
[8] D. S. Bernstein, L. D. Davis & S. W. Greeley (1986), The Optimal Projection Equations for Fixed-Order Sampled-Data Dynamic Compensation with Computational Delay, IEEE Transactions on Automatic Control, Vol. AC-31, 1986, pp. 859-862.
[9] A. M. Azad, T. Hesketh & R. Eaton, Real-time Implementation Multi-rate Sampling Systems in RT-Linux Environment, Proceedings of The fourth International Conference on Control and Automation, Montreal, Canada, 2003, pp.605-609.
[10] F. S. Robert, (1994), Optimal Control and Estimation, (Dover Publication, Inc. New York, 1994).