Notes on Control with Delay

Christopher Brown and David J. Coombs

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 387
August 1991

Abstract

We gather together some introductory and tutorial material on control systems with delay. Delay is especially pernicious in feedback systems, and some form of modeling and prediction is essential to overcome its effects. After a short introduction to the issues, we present four basic techniques for overcoming delay and compare their performance through simulation. The four techniques are the following.

- Predictive techniques to cancel the delay within the loop.
- Cancel negative feedback to obtain open-loop characteristics.
- Smith prediction, which uses a model of the plant.
- System inversion techniques, which use a model of the controller and the plant.

In each of these techniques, predictive filters can be used to overcome latency by providing approximate dynamic predictions of waveforms within the system, such as input and control signals.

This material is based on work supported by the National Science Foundation under Grants numbered IRI-8920771 and CDA-8822724. The government has certain rights in this material.
1 Feedback Control and Delay

We are interested in control systems with delay. Feedback control has several familiar advantages — one important one is the decreased sensitivity of a closed-loop negative feedback system to variations in its parameters. Since the open- and closed-loop systems are not directly comparable, instead we repeat the familiar argument (cf., e.g., [9]) about the decreased sensitivity of feedback systems to parametric variation. Clearly in an open loop system with plant transfer function \(G(s)\) and input and output \(X(s)\) and \(Y(s)\), the change in the transform of the output due to a parameter variation \(\Delta G(s)\) is clearly

\[
\Delta Y(s) = \Delta G(s)X(s).
\]  

(1)

For a closed loop system, find \(Y(s) + \Delta Y(s)\) by substituting \(G(s) + \Delta G(s)\) into the familiar formula \(Y = XG/(1 + GH)\), assume \(GH(s) >> \Delta GH(s)\), and obtain

\[
\Delta Y(s) = \frac{\Delta G(s)}{(1 + GH(s))}X(s).
\]  

(2)

Defining the system sensitivity to be the ratio of the percentage change in the system transfer function to that of the process transfer function, we see that open loop systems have sensitivity unity, but closed loop feedback systems have sensitivity \(1/(1 + GH(s))\), the denominator of which is usually much greater than one.

We bring up this point here only because one of the schemes we shall examine later chooses to sacrifice closed-loop advantages in order to deal with delay. Given that ideally we desire feedback, then why is delay a problem in feedback control systems?

We illustrate with the constant-gain feedback system illustrated in Fig. 1(a). Here the transfer function is simply \(K/(K + 1)\), and the response to a step function is a scaled step function. Thus this system tracks the input perfectly. Fig. 1(b) shows the same system with a delay of \(T\) seconds added.

The responses of these systems to a unit step are shown in the next two figures. Not shown is the perfect step function output of height 0.4117 that exactly tracks the input in the continuous controller, zero delay case. Fig. 2 shows the response of the undelayed discrete feedback loop to a step input. The discrete controller has an implicit sample and hold circuit, and this characteristic gives the discrete realization of the controller some of the aspects of delayed control. These qualitative differences motivate the use of the \(Z\) transform in discrete system analysis. Thus, discrete systems are often written in the manner of Fig. 1(b) with the Laplace transform delay box relabeled as a \(Z\)-transform delay of \(z^{-1}\), indicating the inherent delay implied by stepwise operation. Fig. 3 shows the effect of delay in the continuous version of the controller for \(K = 0.7\), and Fig. 4 shows what happens when the gain exceeds unity.

The decidedly discontinuous performance of the continuous system in the presence of delay requires some explanation. The following paragraphs are stolen from [10]. We have the following situation, where we let the delay be \(\tau\) seconds and we let the time \(t\) vary continuously upward from 0. A step input occurs at time \(t = 0\).

\[
\tau < t < 2\tau \quad \quad y = K
\]
So that

\[
2\tau < t < 3\tau \quad y = 1 - K \\
3\tau < t < 4\tau \quad y = 1 - (1 - K)K = 1 - K + K^2 \\
4\tau < t < 5\tau \quad y = 1 - (1 - (1 - K)K)K = 1 - K + K^2 - K^3
\]

So that

\[
y(n) = \frac{1 - (-K)^n}{1 + K}
\]

at the nth step.

The steady state gain for the delayed system (if it stabilizes) is the same as for the continuous or discrete delayed system steady-state gain: the exponential term in (3) vanishes if the magnitude of K is less than unity.

One can approach the behavior of the delayed system using Laplace transforms. This method puts the problems caused by delay in terms of the delay of the output signal and the poles introduced into the system by the delay. (For an introduction to basic concepts such as "poles", see [9] or any other introduction to control theory.) In the following sections of this report we shall investigate different methods for eliminating the unpleasant effects of the delay and the poles.

The Laplace transform of the output \(y(t)\) given input \(x(t)\) with Laplace transform \(X(s)\) is

\[
\mathcal{L}(y(t)) = Y(s) = \frac{Ke^{-sr}}{1 + K e^{-sr}} X(s) = \frac{K}{K + e^{sr}} X(s).
\]

In the latter form it can be seen that the characteristic equation is not algebraic, and that it has an infinite number of closed-loop poles. In fact, the poles are the roots of
Figure 2: Output of constant gain feedback system for step input and discrete control with $K = 0.7$. Continuous control yields a perfect step.

Figure 3: Output of delayed constant gain feedback system with step input, $K = 0.7$. 
Figure 4: Output of delayed constant gain feedback system with step input, $K = 1.1$.

$$e^{st} = -K = Ke^{i(\pi \pm 2\pi q)}, q = 0, 1, 2...$$

and so

$$s\tau = \ln K + j(\pi \pm 2\pi q), q = 0, 1, 2...$$

Taking $K = 1$ gives poles falling along the imaginary axis, spaced by $2\pi/\tau$, the two principal ones (closest the origin) at $\pm j(\pi/\tau)$ representing the fundamental oscillatory frequency of period $2\pi$ visible in the output. This unstable system then diverges for $K > 1$, since poles move over into the right halfplane. The extension of eq. (4) to a system with a controller (transfer function $C$), a plant ($G$) and delay ($e^{-s\tau}$) is

$$\frac{CGe^{-s\tau}}{1 + CGe^{-s\tau}}X(s).$$

which we shall see again.

In an actual situation, delay again manifests its presence with oscillations. In a straightforward tracking application with the Rochester robot head, one camera on the head is to track a moving object. With the camera moving, the spatial positional error of the camera’s axis is calculated using information about the camera’s position (obtained from reading back angles from the camera’s motors) and the image-coordinate error calculated as the distance of the target’s image from the origin of the image coordinate system. If the time between reading the two necessary data (camera position and image position) is an unmodelled delay, the system performs as shown in Fig. 5.
Figure 5: Tracking a sinusoidally moving target. Delay in calculating camera's true angular position results in an oscillating error that is superimposed on a generally correct sinusoidal head motion. Here the head position graph is from direct readout from the motors, and the error is the target's retinal position measured with vision.
<table>
<thead>
<tr>
<th></th>
<th>Open Loop</th>
<th>Closed Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Delay</td>
<td>basis for comparison</td>
<td>more robust</td>
</tr>
<tr>
<td>Delay</td>
<td>output delayed</td>
<td>output delayed, poles affect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>performance</td>
</tr>
</tbody>
</table>

Table 1: Characteristics of undelayed and delayed systems.

Table 1 sums up these generalities. Compared with open-loop control, a closed loop system can be more resistant to variations in behavior induced by plant parameter variations. Introducing delay into an open loop system simply delays the output, while in a closed loop system it can also introduce instability. In the sections that follow we present five different ways to cope with the effects of delay and try to make some comparisons between them. We shall investigate the following techniques. In every case the starting point is a closed loop system with delay, having the output delay and multiple-pole characteristic equation discussed above.

- Cancel negative feedback – achieve open loop performance, delayed.
- Smith prediction – attain closed loop performance, delayed.
- Signal synthesis adaptive control – attain closed loop performance, undelayed.
- Smith and input prediction – attain closed loop performance, undelayed.
- Predictive techniques to estimate and then predict delayed signals within the loop – with perfect prediction, attain closed loop performance, undelayed.

2 Opening the Loop

In studying primate gaze control, Young [17] wanted to explain how smooth pursuit avoided instability if tracking is modeled as a pure negative feedback system. There are two problems with this model. First, the error, and thus control, signal is zero when accurate tracking is achieved; this should send eye velocity transiently to zero. Second, tracking performance is better than it should be given the delays in the control loop and the time constants of the processes. His proposal is that the system tracks not the retinal image, but a neural signal that corresponds to target motion (in the world).

Robinson [12] describes a mechanism that implements Young's idea: as Robinson says, "if negative feedback bothers you, get rid of it". In the negative feedback system the eye velocity is fed back and subtracted from the target velocity (with some delay). If the eye is in the process of tracking, then the target velocity is the sum of the eye velocity (with respect to the head) and the target's retinal velocity (its velocity with respect to the eye). But the latter is just the error signal resulting from negative feedback. Thus an estimated target velocity signal can be constructed by positively feeding back the commanded eye motion into the control loop, delayed to arrive at the proper time to combine with the error term produced by negative feedback. This mechanism not only provides a signal based
on the target's true motion, but it cancels the negative feedback and thus removes the possibility of oscillations.

In all our examples we shall have a controller in the loop with the plant. Thus the block diagram of the feedback-cancellation idea is shown in Fig. 6. Applying the positive feedback idea in this context involves constructing a model plant $\hat{G}$. If the designer's model of the plant and delay is correct, the system is changed to open-loop, losing all the well-known advantages of closed-loop control. Further the open-loop response is in fact delayed. Given accurate plant and delay models, the designer can do better, as we shall see.

3 Smith Prediction

Smith prediction [13; 14] is a by-now classical technique, is the basic idea behind most modern methods. The treatment in [10] is especially readable. Smith prediction was one of the main tools for managing cooperating delayed controls in the simulation studies of the Rochester Robot [5; 6; 7].

Smith's Principle is that the desired output from a controlled system with delay $\tau$ is the same as that desired from the delay-free system, only delayed by the delay $\tau$. Let the delay be $\tau$, the delay-free series controller be $C(s)$, the desired delay controller be $\hat{C}(s)$ and the plant be $G(s)$. The delay-free system transfer function will be

$$\frac{CG}{1 + CG}.$$

The delay system with its desired controller has transfer function

$$\frac{\hat{C}Ge^{-\tau}}{1 + \hat{C}Ge^{-\tau}}.$$

But Smith's Principle is

$$\frac{\hat{C}Ge^{-\tau}}{1 + \hat{C}Ge^{-\tau}} = \frac{CGe^{-\tau}}{1 + CG}.$$
Figure 7: Smith prediction control.

This quickly leads to the specification for the controller \( \hat{C} \) in terms of \( C \), \( G \), and \( e^{-\sigma T} \):

\[
\hat{C} = \frac{C}{1 + CG(1 - e^{-\sigma T})}.
\]

This simple principle has spawned a number of related controllers, often arising from each other by simple block-diagram manipulation. It is worth noting that Smith did not take the next step and demand that the output should not be delayed. To take this step requires either a non-physical or a prescient component in the system, as we shall see.

Fig. 7 shows the Smith predictor as applied to the standard situation we use in this report, i.e. with continuous control.

This diagram can be rewritten to show the relation to the feedback cancellation technique (Fig. 8).

Assuming exact estimates of \( G \) and \( \tau \), i.e. that \( \hat{G} = G \) and \( \hat{\tau} = \tau \), then the positive feedback coming to summation point \( S1 \) and the negative feedback coming to summation point \( S1 \) in Fig. 8 cancel, and removing them from the diagram yields a simplified diagram in which Smith’s principle clearly holds – the transfer function is

\[
\frac{Y}{I} = \frac{CGe^{-\sigma T}}{1 + CG}.
\]

simply a delayed version of the undelayed closed-loop transfer function.

To sum up, Smith prediction uses the model of the plant in a negative feedback scheme — the controller controls this model. If the model is good, one gets a delayed version of the closed-loop control. That is, the transfer function of the delayed system is changed from

\[
\frac{CGe^{-\sigma T}}{1 + CGe^{-\sigma T}}
\]

to

\[
\frac{CGe^{-\sigma T}}{1 + CG}.
\]
Figure 8: Smith prediction and feedback cancellation.

Figure 9: Cancellation yields Smith's principle.
The delayed closed-loop negative feedback is there also, and the hope is that it would tune up the infelicities in the performance due to inaccurate temporal or parametric (plant) modelling. A result of the delay institutionalized in Smith control is non-zero latency, inducing a steady-state error that can vary depending on the input signal. For position-error tracking of a constant velocity target the latency would cause a steady-state constant positional error with the tracker trailing behind the target.

4 Signal Synthesis Adaptive Control

4.1 Background

Primate systems seem to track with zero latency. Bahill and his coworkers [1; 11] propose a scheme called signal synthesis adaptive control to achieve the advantages of undelayed feedback control with zero latency. As we shall see, this requires predicting the future, something that control theorists are not fond of doing. The scheme is called "signal synthesis" control because the controller is predicting the input signal. It is called "adaptive" because in the published formulation there was a scheme to switch between predicted signals to keep the prediction in line with reality (if a trajectory velocity curve changed from square wave to sine wave, for example).

The derivation of this type of control presents us with a powerful technique. In fact, on paper it is possible to achieve many different sorts of control, including cancelling the latency but leaving the poles, cancelling the poles but leaving the latency, cancelling both, and cancelling both and substituting the effect of an entirely new controller for the existing controller \( C \). As described, the technique simply uses an inverse plant. These are hard to engineer, since for a plant consisting of integrators one obtains an inverse consisting of differentiators. Still, we shall see that we can rewrite the resulting inverse plant to be realizable and in fact we shall see it has a close relation to the techniques we have seen so far.

4.2 Inverting a Delayed Feedback System

As a warm up, let us get some intuition by computing the inverse of our standard delayed feedback system (Fig. 10(a)). The system is

\[
Y(s) = \frac{C(s)G(s)e^{-sr}}{1 + C(s)G(s)e^{-sr}I(s)} \tag{8}
\]

The inverse of the system, remembering that Laplace transforms compose by multiplication, is just the reciprocal of this expression:

\[
Y^{-1}(s) = \frac{1 + C(s)G(s)e^{-sr}}{C(s)G(s)e^{-sr}} I(s), \tag{9}
\]

which implies that

\[
Y^{-1}(s) = I(s)(\frac{1}{C(s)G(s)e^{-sr}} + 1) = I(s)(\frac{e^{sr}}{C(s)G(s)} + 1). \tag{10}
\]
Translating the rightmost expression into a block diagram yields Fig. 10(b), in which certain characteristic features appear including a predictive (non-physical, non-causal) time-advance component, inverses of the individual components of the system, and a positive feed-forward of the input signal.

4.3 The System-inverting Controller

The basic block diagram of the system-inverting controller (SIC) is given in Fig. 11. The block labeled SIG. SYNTH. CTRL. does the work. We need to derive what this block should do (i.e. how to characterize \( B(s) \) in terms of the rest of the system and its desired function). To see how the method goes, let us start by imposing the requirement that we want to synthesize a zero-latency version of the undelayed closed loop original system. This takes us a step beyond Smith prediction since we will now reduce the latency to zero. We thus want

\[
Y(s) = I(s).
\]

For this equation to hold, we have (selectively dropping the Laplace transform variable \( s \)):

\[
Y = \frac{CGe^{-\sigma_T}}{1 + CGe^{-\sigma_T}}(I + B).
\]  

(11)

Since

\[
Y = I = \frac{CGe^{-\sigma_T}}{1 + CGe^{-\sigma_T}}(I + B),
\]
Finally, (12)

\[ B = I(1 - \frac{CGe^{-sr}}{1 + CGe^{-sr}})(\frac{1 + CGe^{-sr}}{CGe^{-sr}}). \]

Finally,

\[ B = I(\frac{1 + CGe^{-sr}}{CGe^{-sr}} - 1). \quad (12) \]

In eq. (12), the first term inside the parentheses is the inverse of the system, that is of the delayed plant and feedback controller. The second term (−1) simply cancels the input! In this form the SIC block diagram appears as in Fig. 12.

The transfer function of the SIC producing \( B(s) \) can be rewritten (multiply the first term in parentheses by \( e^{sr} \), divide out, simplify)

\[ \frac{e^{sr}}{CG} + 1, \]

which we saw in Section 4.2. Written like this, the SIC block diagram is as shown in Fig. 13.

A last rewriting of this particular system shows that it can be realized without any explicit inversions of the components. Thus it can be implemented with the same components that were used to make the Smith controller, as long as the signal is predictable (Fig. 14).
Figure 13: Another realization of the signal synthesis controller for zero latency realization of closed-loop control.

Figure 14: Block diagram manipulation yields this realization of the system of Fig. 12 or 13.
This basic technique of writing down the desired transfer function of the system and then solving for $B$ is clearly quite general. For instance, to synthesize a new delay-free system with controller $D$ and plant $H$, simply invert the existing plant, controller, and delay and substitute the desired ones as follows.

We want the system response to be

$$\frac{Y}{I} = \frac{DH}{1 + DH}$$

Using the block diagram defining the SIC (Fig. 11) we equate what the system will do to what we want it to do:

$$(B + I)\left(\frac{CGe^{-st}}{1 + CGe^{-st}}\right) = I\left(\frac{DH}{1 + DH}\right)$$

Solving for $B$ we get

$$B = I \left[ \frac{DH}{1 + DH} \right] \left[ \frac{1 + CGe^{-st}}{CGe^{-st}} - 1 \right].$$

Here again we see the input being subtracted and the inverse of the unwanted system composed with the desired system (Fig. 15).

Finally, we leave it to the reader to apply this technique to the following problem. Given the delayed system

$$\frac{CGe^{-st}}{1 + CGe^{-st}},$$

the goal is to remove the latency of the system response but not affect the closed loop poles. That is, the goal is to remove the $e^{-st}$ in the numerator. Derive that

$$B = I(e^{st} - 1),$$

I.e. the controller cancels the input and sends out an advanced version of it to the existing (delayed) controller, all of which makes sense intuitively and is rather elegant.

5 Enhancing Smith Prediction with Input Prediction

As we have seen, pure Smith prediction removes the delay-induced poles from the controller but leaves the latency. To remove the latency in the Smith controller, it only remains to predict the input, as in the SIC.

A step in this direction is taken in [4; 5], in which an explicit kinematic simulation used to predict the system state, and an optimal (i.e. variance-minimizing) filter is used to smooth the position and velocity estimation of the world object. Extended Kalman filters, (linear) Kalman filters, and time-invariant filters were investigated as input estimators. However, in this work the filters were not used in their predictive capacity (Fig. 16).
Figure 15: System inversion allows any controller to be substituted for the original.
Our goal in the current work is to predict the signal for the purpose of compensating for delays. The predictor will be placed early in the system, before the Smith predictor. The pure predictor of the SIC is an “oracle” usually dismissed as unphysical or noncausal. In this section we introduce the idea of a predictive filter to supply the necessary offset to remove the Smith predictor’s latency. It is standard practice with optimal filtering to use statistical techniques to see if the current dynamic model fits the data, and if not to substitute another model [3; 8]. This “variable dimension” approach is the predictive filtering equivalent of the signal synthesis adaptive control scheme [2].

One example of a system incorporating Smith prediction and signal estimation is shown in Fig. 16.

5.1 The $\alpha - \beta$ filter

Linear dynamical systems with time-invariant coefficients in their state transition and measurement equations lead to simpler optimal estimation techniques than are needed for the time-varying case. The state estimation covariance and filter gain matrices achieve steady-state values that can often be computed in advance. Two common time-invariant systems are constant-velocity and constant-acceleration systems.

Let us assume a constant velocity model: starting with some initial value, the object’s velocity in LAB evolves through time by process noise of random accelerations, constant during each sampling interval but independent. With no process noise the velocity is constant; process noise can be used to model unknown maneuverings of a non-constant velocity target. The cumulative result of the accelerations can in fact change the object’s velocity arbitrarily much, so we model a maneuvering object as one with high process noise. For this work we assume position measurements only are available, subject to measurement noise of constant covariance. Clearly the more that is known a priori about the motion the better the predictions will be. Some sensors or techniques can provide retinal or world velocity measurements as well.

Assume the object state (its position and velocity) evolves independently in each of the $(X, Y, Z)$ dimensions. For instance, in the $Y$ dimension, it evolves according to

$$y(k + 1) = F_Y y(k) + v(k),$$

(14)

where

$$F_Y = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

(15)

for sampling interval $\Delta t$, error vector $v(k)$, and $y = [Y, Y']^T$. The equations for the other two spatial dimensions are similar, and in fact have identical $F$ matrices. Thus for the complete object state $x = [X, X, Y, Y', Z, Z']^T$, $F$ is a $(6 \times 6)$ block-diagonal matrix whose blocks are identical to $F_Y$. The error vector $v(k)$ can be described with a simple covariance structure: $E(v(k)v^T(j)) = Q_\delta_{kj}$.

The $\alpha - \beta$ filter for state prediction has the form

$$\mathbf{x}(k + 1|k + 1) = \mathbf{x}(k + 1|k) + \begin{bmatrix} \alpha \\ \beta/\Delta t \end{bmatrix} [z(k + 1) - \mathbf{x}(k + 1|k)],$$

(16)
Figure 16: Smith prediction with kinematic model, signal smoothing and estimation with $\alpha - \beta$ filter. This system exhibits latency.
where $\hat{x}(k+1|k+1)$ is an updated estimate of $x$ given $z(k+1)$, the measurement at time $k+1$. Here we assume that $z(k+1)$ consists of the three state components $(X, Y, Z)$ (but not $(\dot{X}, \dot{Y}, \dot{Z})$). The state estimate is a weighted sum of a state $\hat{x}(k+1|k)$ predicted from the last estimate to be $F\hat{x}(k|k)$ and the innovation, or difference between a predicted measurement and the actual measurement. The predicted measurement $\hat{z}(k+1|k)$ is produced by applying (here a trivial) measurement function to the predicted state.

The $\alpha - \beta$ filter is a special case of the Kalman filter. For our assumptions, the optimal values of $\alpha$ and $\beta$ can be derived (see [3], for example) and depend only on the ratio of the process noise standard deviation and the measurement noise standard deviation. This ratio is called the object's maneuvering index $\lambda$, and with the piecewise constant process noise we assume,

$$\alpha = -\frac{\lambda^2 + 8\lambda - (\lambda + 4)\sqrt{\lambda^2 + 8\lambda}}{8}$$  \hspace{1cm} (17)

and

$$\beta = \frac{\lambda^2 + 4\lambda - \lambda\sqrt{\lambda^2 + 8\lambda}}{4}.$$  \hspace{1cm} (18)

The state estimation covariances can be found in closed form as well, and are simple functions of $\alpha$, $\beta$, and the measurement noise standard deviation.

### 5.2 The $\alpha - \beta - \gamma$ Filter

The $\alpha - \beta - \gamma$ filter is like the $\alpha - \beta$ filter only based on a uniform acceleration assumption. Thus it makes a quadratic prediction instead of a linear one. Broadly, it tends to be more sensitive to noise but better able to predict smoothly varying velocities. Its equation is the following.

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + \begin{bmatrix} \alpha \\ \beta/\Delta t \\ \gamma/\Delta t^2 \end{bmatrix} [z(k+1) - \hat{z}(k+1|k)],$$  \hspace{1cm} (19)

With the maneuvering index $\lambda$ defined as before, the optimal $\alpha$ and $\beta$ for the case that the target experiences random small changes in acceleration (random jerks) are the same as before and the optimal $\gamma = \beta^2/\alpha$.

Both the $\alpha - \beta$ and $\alpha - \beta - \gamma$ filters have been implemented as C++ classes, in versions with uniform and nonuniform timesteps. The nonuniform timestep versions are easy extensions in which the timestep is calculated at each iteration as the difference of the last two timestamps. Appendix A gives the complete code for the uniform timestep $\alpha - \beta - \gamma$ filter.

### 5.3 Input Prediction

The smoothing effect of the estimation process improved the performance of the system in noise [4; 5]. The predictive version of the filter was not used to try to remove latency, however. The classical Smith predictor can be enhanced with some form of input prediction in order to ameliorate the latency built into Smith's principle.
Figure 17: Smith prediction control with input prediction to overcome latency.

The resulting block diagram is shown in Fig. 17.

The predictive element $e^{\tau}$ may be realized nonphysically by a "prescience filter" or oracle, by actual prior knowledge of the signal, or (approximately) by a predictive filter. By the same token, the predictive element in the signal synthesis controller shown in Fig. 14 may be realized by any of these mechanisms. In later sections we shall do some experimental work to assess the relative efficacy of the resulting controllers.

6 Controller Sensitivity

One of our goals is to determine quantitative performance characteristics of each approach to predictive control. Parametric sensitivity analyses quantify the degradation in performance of controllers to variation in parameters, and temporal sensitivity analyses similarly quantify the effects of time delays on the system.

Marshall's book [10] has a readable though terse treatment of one analytic approach to sensitivity that is particularly useful in the design and construction of adaptive controllers. The result is due to Tomovic ([15; 16]). If the parameter of interest is $a$, then write the system transfer function as $G(s, a)$. Then the Laplace transform of the output $y(t)$ is $Y(s, a) = X(s)G(s, a)$. We assume that the input $x(t)$ is independent of the parameter $a$. Tomovic's result follows immediately from the definition of the Laplace transform and is that

$$\mathcal{L}\left(\frac{\partial y(t, a)}{\partial a}\right) = \frac{\partial}{\partial a}G(s, a)$$

(20)

That is, the Laplace transform of the sensitivity function $\partial y/\partial a$ is the product of the Laplace transform of the input and the partial derivative of the system transfer function with respect to $a$. The utility of this result is that usually the partial derivative of the system transfer function with respect to $a$ is very close to the transfer function itself, and can be easily built physically since it is close to the original system. Cascading this derivative system with the original system then yields an output that is the sensitivity function. As it happens, the original transfer function is a factor of the derivative function, and it is often
possible to find nodes within the original system whose value is the remaining factors. In turn, this means that a duplicate of the original system can be cascaded with output from these sensitivity take-out points to yield the derivative (Fig. 18).

The method works for delay-free and delayed systems. For example, for the second order system

$$\frac{d^2 y}{dt^2} + a \frac{dy}{dt} + by = x(t),$$

the transfer function is

$$\frac{e^{-\sigma \tau}}{(s^2 + as + b)}$$

and the sensitivity coefficient with respect to $a$ is

$$\frac{\partial Y}{\partial a} = \frac{e^{-\sigma \tau}}{-sX(s^2 + as + b)^2}.$$

For a system $G(s)$ cascaded with a delay $e^{-\sigma \tau}$, the output $z(t)$ is a delayed version of $y(t)$, namely $y(t - \tau)$. Then the temporal sensitivity function

$$\mathcal{L} \left( \frac{\partial z}{\partial \tau} \right) = -sZ(s)$$

So the temporal sensitivity function is just the time derivative of the system output. Clearly this makes sense: if a system is changing rapidly, delay changes can result in large differences in values, whereas if the system is not changing at all, delays have no effect whatever.

Applying these techniques to the system shown in Fig. 7 is easy. The transfer function is

$$Y(s) = \frac{C Ge^{-\sigma \tau}}{1 + C \hat{G} + C (Ge^{-\sigma \tau} - \hat{G}e^{-\sigma \tau})} X(s) = \Gamma(s) X(s).$$

(21)
If $G$ is a function of some parameter $a$, then

$$\frac{\partial}{\partial a} Y(s) = \frac{Y(s)}{G} \left[ \frac{1 + CG - CGe^{-st}}{1 + CG + C(Ge^{-st} - Ge^{-st})} \right] \frac{\partial G}{\partial a}. \tag{22}$$

The term in square brackets may be rewritten as $(1 - \Gamma(s))$. Note that all parameter sensitivity functions will contain the common term $\Gamma(s)(1 - \Gamma(s))$. In particular the temporal sensitivity of Smith's method is

$$L \left( \frac{\partial y}{\partial \tau} \right) = X(s)\Gamma(s)(1 - \Gamma(s)) \frac{\partial e^{-st}}{\partial \tau} \frac{1}{e^{-st}}. \tag{23}$$

Further, it is easy to see that $(\partial e^{-st} / \partial \tau)/e^{-st} = -s$.

These calculations can be extended to the case of sensitivity in the case of mismatch between the modeled plant and the real one, and between the modeled delay and the real one [10]. As of now we have not been able to duplicate this derivation using Mathematica. There is also an approach that uses perturbation analysis and allows the effects of mismatch to be displayed as a change in the block diagram of the system, which is useful for intuition.

At this point more thought could be given to the analytic treatment of sensitivity. In particular there are the following obvious problems.

1. Go to the literature in a more responsible way.
2. Derive the sensitivity functions for signal synthesis or system inverting control (SIC).
3. Compare the parametric and temporal sensitivity of Smith and SIC. This involves thinking about the problem in the Laplace domain or inverse transforming the sensitivity functions.
4. Understand and carry through the mismatch sensitivity calculations for Both Smith and SIC and compare the two approaches.

Leaving these interesting issues aside for now, we proceed to simulation experiments.

7 Experiments

7.1 Simulator and Example System

To explore the effects of noise and systematic error on the control schemes described here we wrote a control system simulator. The user specifies the characteristics of blocks in a system diagram, as well as sample rates, input function, and length of run. The system can be continuous, discrete, or hybrid. A continuous system (or block) is specified by its differential equation (a variable step Runga-Kutta method is used for solutions). Discrete components include integrators (with various options of anti-windup, integration time, saturating or not, and leak), sample and hold, differentiators, gains, inputs (impulse,
step, ramp, cosine, and squarewave), additive noise (uniform or gaussian), summers, and output blocks. Each block can induce a delay or achieve a time advance. Continuous blocks in use are an integrator, a leaky integrator, a cascaded integrator and leaky integrator, and a spring, mass, damper system. There are also predictors of several varieties. Currently, there are nonphysical “prescient” oracles that look into the future of their input as well as $\alpha - \beta$ and $\alpha - \beta - \gamma$ filters – the predictors have a provision to produce noisy predictions in which the time to look ahead is noisy. New blocks are easy to add and the functionality of blocks is easy to change.

Using this tool, the systems illustrated in Figs. 6, 14, and 17 were simulated. Several examples were tried, to establish for instance that in the ideal case of perfect modelling both the Smith and SIC controllers achieved the same effect as the undelayed controller. For this report we ran experiments in which noisy predictions (rather, accurate predictions but for noisy values of the future time) and noisy plant models (simulated simply by adding noise into the system). Since this experimental approach is really no substitute for an analytical understanding of what is happening, we limited our experiment to a typical case that has enough complexity to produce interesting behavior. We chose a PID discrete controller applied to a continuous spring, mass, damper system.

The action of this controller acting alone on the system is illustrated, first for a step input and then for a “sinusoidal” input (actually $u(t)(1 - \cos(\omega 2\pi t))$, with $u(t)$ the unit step). Figs. 19 and 20 show the response of the system and the necessary control signal to a step input, using a good setting of $P, I,$ and $D$ (0.4, 0.4, and 0.8). Fig. 21 shows the step response of the system for a bad setting of $P, I,$ and $D$ (0.9, 0.4, 0.1). Fig. 22 shows the sinusoidal input. Figs. 23 and 24 show the sinusoidal response of the system for the good and bad settings, respectively.

7.2 The PID-MSD System in Noise

The goal of these experiments is to compare the performance of three approaches overcoming delay. Each approach depends on a system model, and two approaches use models of the input. In the experiments to be described, we concentrate on stochastic errors rather than systematic errors. In particular, we consider additive error in the output of the controller that we intend to model the effects of parametric mis-estimation of the system. For temporal sensitivity, we consider stochastic mis-estimation of the correct delay.

To set a baseline for these experiments, we first consider the performance of the pure PID controller for the MSD (mass, spring, dashpot) system described in the previous section. The system diagram is that of Fig. 25.

Clearly there are many parameters that can be interestingly manipulated in experiments on control systems, and many measurements that can characterize the effects. First, to get some idea of the effects of delay in the feedback loop, consider the system response in the case of a delay of 0.4 compared with no delay (besides the inherent one-step discrete controller delay) (Fig. 26).

For this set of experiments, the sampling rate of the discrete controller is a constant 0.1 second. The noise added to the controller output is normally distributed, with a mean of 0.0 and a standard deviation of 0.2. The stochastic time perturbation is calculated as
Figure 19: Output of mass, spring, and dashpot system to step input using \((P,I,D) = (0.4, 0.4, 0.8)\).

Figure 20: Control signal producing the output of Fig. 19.
Figure 21: Output of mass, spring, and dashpot system to step input using \((P, I, D) = (0.9, 0.4, 0.1)\).

Figure 22: Sinusoidal reference input \(u(t)(1 - \cos(t \omega 2\pi))\), with \(u(t)\) the unit step, \(\omega = 0.5\). This input is used throughout rest of work.
Figure 23: Output of mass, spring, and dashpot system to $u(t)(1 - \cos(2\pi \omega t))$ input for $\omega = 0.5$ using $(P,I,D) = (0.4, 0.4, 0.8)$.

Figure 24: Output of mass, spring, and dashpot system to sinusoidal input using $(P,I,D) = (0.9, 0.4, 0.1)$.
Figure 25: Perturbations in the PID controller. The controller output has additive noise and there is mis-estimation of the "zero" delay, implemented by stochastic non-zero delay or advance. In fact the delay is one time step, since this is actually a discrete controller – we have indicated that inherent delay with the $z^{-1}$ box.

Figure 26: Zero delay case compared to a constant delay of 0.4 in the feedback loop of the system. The delay causes overshoot and a phase lag.
the integral number of ticks that results from rounding a normally distributed variate with mean 0.0 and standard deviation of 0.1. The control input is $u(t)(1 - \cos(2\pi \omega))$. The performance of the perturbed systems can be compared to that of the basic PID controller and to noisy versions of the PID controller by the metrics of phase lag and gain (Bode plots) if the frequency of the input sinusoid is varied. Within a frequency, such metrics as overshoot and the sum of absolute differences of the relevant system state (position) suggest themselves.

Fig. 27 shows the system state (position of the mass) under the noisy conditions described above compared with the noiseless case.

7.3 Delay and Simple Prediction

The PID-MSD system is relatively resistant to delay, but its effects are significant. One idea is to insert an estimator-predictor in the loop to cancel the effects of delay. Fig. 28 shows the system.

Here the delay is compensated by an $\alpha - \beta$ filter that estimates and predicts the output of the controller. Using an $\alpha - \beta - \gamma$ filter has advantages and disadvantages: the advantages
Figure 29: The undelayed response (solid line) is perturbed by a delay of 0.3s in the control loop (dashed line). Using the estimator to predict the control signal reduces the phase lag but overshoot remains.

are less overshoot, the disadvantage is a longer initialization time (three readings instead of two) that can make a significant difference in performance when control is active. Fig. 29 shows the zero delay case, the case where delay is 0.3 sec. and no prediction is used, and the case of 0.3s delay and the prediction is for 0.3 sec. ahead. In this system the prediction eliminates the phase lag of the system induced by the delay, but the overshoot of the output in the delayed and prediction-compensated systems is similar.

7.4 Open Loop Control

It is difficult to compare the performance of the open-loop control approach to the other methods because this approach calls for an entirely new system design: the open loop controller in general looks nothing like the closed loop controller. What we show here is simply the degradation of the performance of the controller whose loop has been opened by positive feedback under the two noise conditions of this section: the standard sinusoidal input, additive noise of 0.2 in the positive feedback path, and normally distributed stochastic variation of the delay from its mean of 0.0 with a standard deviation of 0.1. For comparison we simply give the noiseless performance, which should be (and in fact is, in our simulation) the same as the output of the PID controlled MSD system with negative feedback disabled. This output is not related to the closed-loop PID-MSD system performance, of course. Fig. 30 gives the system configuration and Fig. 31 the three responses.

As mentioned in Section 2, the process of opening the control loop by positive feedback does not remove the delay, or latency, from the output. As with all our schemes, the solution is simply to cascade an input predictor with the system.
Figure 30: The positive feedback loop-opening system, showing where noise and stochastic delay were injected.

Figure 31: The open loop PID-MSD system output for $u(t)(1 - \cos(2\pi wt))$ input. Ideal output is shown along with output with $G(0.0, 0.2)$ noise in the positive feedback loop and $G(0.0, 0.1)$ stochastic delay in the positive feedback loop.
Figure 32: The Smith predictor with noise added to simulate plant and delay mis-modeling. The STOCH. DELAY box represents a stochastic variation about the modelled delay, which itself is ideally the same as the true plant delay.

Since the open- and closed-loop systems are not directly comparable, we simply invoke the argument of Section 1 concerning the decreased sensitivity of feedback systems to parametric variation. If we can afford to give up the advantages of feedback systems, then an open-loop controller can of course be used, and then delay will only cause a latency in output that can be more or less compensated by predictive filtering somewhere in the system.

7.5 Smith and Signal Prediction

The block diagram of the system used in this section is given in Fig. 32. As before, the controller is the PID controller of earlier sections and the system is the same mass, spring, damper system.

This system diagram is interesting because it brings up the following question. Suppose that mis-modelling the plant actually is modelled as additive noise emanating from the plant model. In the case of zero modelled delay, this noise is both added to and subtracted from the error signal, and thus cancels itself completely. This argument extends to any behavior of the plant model at all, of course: it is all subtracted out in the zero-delay case.

Consider two cases, the zero delay case with noises as we have had previously, and a case in which the plant delay is 0.2, which is accurately reflected in the model delay except when that modelled delay is perturbed by our standard stochastic perturbations. In this case the effect of the noise is doubled, since a shifted version is subtracted from itself at the input. We obtain the graphs of Figs. 33 and 34. As expected, the Smith predictor avoids the deleterious effects of systematic delay (Fig. 26) on the system output, but retains a delay or latency in its output equal to that of the system delay.
Figure 33: The Smith controller with zero modelled delay. Ideal output is shown (solid line) along with output with $G(0.0, 0.2)$ noise in the feedback loop (long dashes) and $G(0.0, 0.1)$ stochastic delay in the feedback loop (short dashes). The controller cancels the noise in this case (see text), so the long dashed line is the same as the solid one.

Figure 34: As in the previous figure, only with 0.2 system delay. Compared with the last figure, note the similarly-shaped, but delayed, ideal output, which is shown along with output with $G(0.0, 0.2)$ noise in the feedback loop and $G(0.0, 0.1)$ stochastic delay in the feedback loop.
7.6 Signal-Inverting Control

Although this controller is inspired by, and in its most general form (Fig. 15) demands the inversion of the following system, here we use the rewriting that merely calls for a duplication of the controller and a model of the plant (along with a nonphysical signal advancer). Fig. 35 shows the system used in the simulations. Fig. 36 shows the results with the standard noise parameters of this section.

8 Input Prediction

In the experiments in Section 7, the Smith and SIC controllers needed to be able to predict the input in order to overcome all the effects of delay. So far we have assumed an oracle that actually knows the future. This is an impractical general solution (although in some cases one can make an argument that very precise expectations are downloaded into the controller – this is a central idea in the signal synthesis adaptive controller [1; 11]).

In this section we examine the effectiveness of the $\alpha - \beta$ and $\alpha - \beta - \gamma$ filters for input prediction. Recall that the former makes linear predictions based on past input and its
inherent beliefs about the reliability of its sensors versus the predictability of the target. The latter makes quadratic predictions.

As an example, Fig. 37 shows our standard sinusoidal input \( u(t)(1 - 2\pi \omega \cos(t)) \) and the predicted value for 0.2 seconds into the future yielded by the \( \alpha - \beta \) and \( \alpha - \beta - \gamma \) filters with \( \lambda = 1 \) (equal confidence in data and prediction). Neither filter's assumptions about constant velocity or acceleration is met. The \( \alpha - \beta - \gamma \) filter delivers reasonable predictions with less overshoot. The \( \alpha - \beta \) filter is rather less satisfactory.

The \( \alpha - \beta - \gamma \) filter may be substituted for the input predictor in the Smith and SIC schemes. For the Smith predictor and a delay of 0.3, the results are shown in Fig. 38. The output is close to the ideal output, but shows the characteristic overshoot engendered by the predictive filter.

When the \( \alpha - \beta - \gamma \) filter is substituted for the input predictor in the SIC scheme, the results are not quite as good. Fig. 39 shows the ideal and SIC outputs and the corresponding one from the Smith predictor (identical to the predictive filter output of Fig. 38).

A closer approximation to the desired output is actually possible by setting the prediction to 0.4 instead of the correct 0.3. Not only does this change result in less overshoot but in a much closer following of the desired curve between the peaks. This sort of tradeoff is also noticed in the laboratory, where the delay and \( \lambda \) (maneuvering index) parameters may be traded off to improve performance.

For noiseless data, the maneuvering index (reflecting our faith in our sensors over that of our predictions) may be increased. Substantially improved performance results with
Figure 37: $\alpha - \beta$ and $\alpha - \beta - \gamma$ filters predicting sinusoidal input. The quadratic predictor performs better in noiseless data.

Figure 38: The $\alpha - \beta - \gamma$ filter predicts the sinusoidal input to the Smith controller (noiseless conditions). Shown are the ideal performance, the output of the Smith predictor (identical to the ideal with a latency equal to the system delay of 0.3), and the output with the predictive filter. The predicted case quickly moves to approximate the ideal except for overshoot.
Figure 39: The $\alpha - \beta - \gamma$ filter predicts the sinusoidal input to the SIC controller (noiseless conditions). Shown are the ideal performance, the output of the SIC controller for the delay of 0.3, and the comparable output of the Smith predictor shown previously.

$\lambda = 10.0$ (Fig. 40).

9 Discussion

Our goals here were two: understand classical work on control with delay, and apply the most promising techniques in our laboratory to cope with the real delays we experience.

We suspect that there has been more written on delay control that we should read. We feel that mathematical analysis of the systems we have discussed could well be pushed farther. Our simulator has proven a useful and flexible tool and could support countless more experiments by varying parameters of noise, input frequency, delay accuracy, and mismatching of modelled controllers and plants with the real ones.

A summary of our evaluation of several approaches to control of time-delay systems is as follows.

1. Ignore the delay or compensate by lowering controller gains: Infeasible in general, this solution might work for some applications because of the relatively short delays that arise with some of our equipment.

2. Opening the loop through positive feedback: This solution does not appeal, since it loses all the advantages of feedback control and seems sensitive to delay.

3. SIC control: Cancelling various parts of the downstream system by inversion is a general and powerful, if practically difficult, idea. This approach requires modelling
Figure 40: Ideal performance and the Smith predictor with its $\alpha - \beta - \gamma$ predictive filter predicting the sinusoidal input. The performance of the filter with noiseless input data is much better with $\lambda = 10$ than with $\lambda = 1$.

of the controller as well as of the controlled plant. For a software controller that is not infeasible, but complicates matters somewhat. There are useful cases where its realization does not call for computing the inversion of the controlled plant (generally a bad idea because it would call for differentiators). We have not seen this rewriting of the SIC controller presented elsewhere. The SIC seems slightly more sensitive to noise and delay mismatches than does the next scheme, but probably not enough to worry about.

4. Smith with input prediction control: This classical scheme is elegant and has some practical advantages (not modelling the controller, just the plant).

It is worth mentioning that predictive filters have more uses than predicting the signal: they also estimate the signal. The latter facility is useful when the signal is noisy or drops out. Our $\alpha - \beta$ and $\alpha - \beta - \gamma$ filters continue predicting and estimating until a certain amount of time (a filter parameter) elapses during which the filter sees no input data. We expect the estimation aspect of the filters to be as useful as their predictive aspect in practical situations, since the delays confronting us with the MaxVideo equipment and robot head camera controllers is relatively small (on the order of 100 ms or less) compared say to the delays we see with the Puma robot and its VAL controller (on the order of 500 ms).

The filter software described here is used in several laboratory projects and the simulator. We hope that this document and possibly the simulation software we have developed will prove useful for further practical work in the laboratory or at least for helping to understand delay systems.
References


A The $\alpha$ - $\beta$ - $\gamma$ Filter in C++

/*************** Header File filt.h ****************/

#ifndef _filt_h
#define _filt_h

/* did we see a target or not on this "tick"? */
enum data_type {data, nodata};

/* what is the filter doing? */
enum filter_activity{invalid, initializing, tracking, predicting, timed_out};

/* this struct is used to set filter characteristics at creation time. */

typedef struct filter_params{
  double time_step;
  double timeout_secs;
  int init_ticks;
  double alpha, beta, gamma;
} *filter_params_t;
#endif

/******* Header File ABC.h *******/
#ifndef _ABC_h
#define _ABC_h

#include <stream.h>
#include "filt.h"

class abcfilter{

protected:
  filter_params f_params;
  int timeout_ticks;
  double xlast, xcurr, vlast, vcurr, vave; // for the a-b filter calc.
  double xave, alast, acurr, aave; // for a-b-c calc.
  double x_pred, v_pred, a_pred, innov; // more intermediate variables
  double N;

public:
  filter_activity activity;

38
double x_est, v_est, a_est;    // estimated variable
double x_future, v_future, a_future;
// predicted variable at time advance in the future;
int age_ticks;  // total age;
double age_secs;  // total age;
int data_ticks;  // how long we've seen target
double data_secs;  // how long we've seen target
int nodata_ticks;  // how long we've not seen target
double nodata_secs;  // how long we've not seen target

// all the above are public so user can just inquire...

abcfilter(filter_params parms);  // create and init
void run(data_type signal, double x, double advance);  // assumes runs every tick.
void run(data_type signal, double x, double advance, double timestamp);
// for irregular data arrival
void dump(void);
);
#endif

#include "ABC.h"

/* Create and Initialize Filter */

abcfilter::abcfilter(filter_params parms)
{
    int i;

    f_params = parms;

    activity = initializing;
    age_ticks = 0;
    age_secs = 0.0;
    data_ticks = 0;
    data_secs = 0.0;
    nodata_ticks = 0;
    nodata_secs = 0.0;
    timeout_ticks = irint(f_params.timeout_secs/f_params.time_step);

    xcurr = vcurr = vave = xave = aave = acurr = 0.0;
void abcfilter::run(data_type signal, double X, double advance) {

    activity = (age_ticks < f_params.init_ticks) ? initializing : tracking;
    //tracking or predicting actually, but we'll find that out later

    double adv = advance;

    if (activity == initializing) {
        if (signal == nodata) {activity = timed_out; return;}
        //policy decision -- don't initialize through dropout

        N = double(age_ticks) + 1.0; //for running average in initialization

        if (age_ticks >= 0) {
            xlast = xcurr;
            xcurr = X;
        }

        if (age_ticks >= 1) {
            vlast = vcurr;
            vcurr = (xcurr - xlast)/f_params.time_step;
        }

        if (age_ticks >= 2) {
            alast = acurr;
            acurr = (vcurr - vlast)/f_params.time_step;
            aave = aave + (acurr - aave)/(N-2.0);
        }

        if (age_ticks < 2) activity = invalid;

    } /* now compute the estimates and future predictions during initialization */
\[
x._{est} = x._{curr}; \\
v._{est} = v._{curr}; \\
a._{est} = aave; \\
x._{future} = x._{est} + v._{est}*(adv) + \\
+ 0.5 * a._{est} * (adv*adv); \\
v._{future} = v._{est} + adv*a._{est}; \\
a._{future} = a._{est}; \\
\]

//end initialization phase

else // in this case, we are actually filtering
{

/* start timer if no data...*/

if (signal == nodata) // no data this time .. start predicting blindly
{
if (nodata_ticks++ >= timeout_ticks)
{activity= timed_out; return;}
activity = predicting;
data_ticks = 0;

x._{est} = x._{est} + v._{est}*f.params.time_step \\
+ 0.5*a._{est}*f.params.time_step*f.params.time_step;

//predict based on last known vel, acc.. \\
v._{est} = v._{est} + a._{est}*f.params.time_step ; //last known acc...
}

/* Following are the alpha-beta-gamma equations! */

if (signal == data) //got data this time, can do the full calculation
{
activity = tracking;
nodata_ticks = 0;
data_TICKS++;
\[
x._{pred} = x._{est} + f.params.time_step*v._{est} \\
+ 0.5*a._{est}*f.params.time_step*f.params.time_step; \\
v._{pred} = v._{est} + f.params.time_step*a._{est};
\]
innov = X - x._{pred};
//compute the current estimate \\
x._{est} = x._{pred} + f.params.alpha*innov; \\
v._{est} = v._{pred} + (f.params.beta/f.params.time_step)*innov;

a_est = a_est +
(f_params.gamma / (f_params.time_step*f_params.time_step))*innov;
}

// compute the predicted future value
x_future = x_est + v_est*adv + 0.5*a_est*adv*adv;
  v_future = v_est + adv*a_est;
a_future = a_est;
}

age_ticks++;  // go to next time step
age_secs += f_params.time_step;  // in real time...
data_secs = data_ticks*f_params.time_step;
nodata_secs = nodata_ticks*f_params.time_step;
}
  // end run