Improving PID Control with Unreliable Communications

Jianping Song, Aloysius K. Mok
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712

Deji Chen, Mark Nixon,
Terry Blevins, Willy Wojsznis
Emerson Process Management
12301 Research Blvd. Building III
Austin, TX 78759

KEYWORDS

PID algorithm, wireless communications, Distributed Control System - DCS, unreliable communication

ABSTRACT

For standard industrial process control systems, blocks in a control loop are executed periodically. That is, the sampling data is fed to control blocks by sensors at a fixed rate. At the same rate, control blocks do some calculations on the inputs and send out commands to actuators. This paradigm works well with reliable wireline networks, such as Fieldbus and ProfiBus. However, it is a justified assumption that part of the wired networks will be replaced by wireless networks. With wireless networks, we shall expect intermittent communication losses. In this paper, we first identify the poor dynamic response of the standard PID algorithms in the case of lost communications. An enhanced PID algorithm is proposed to improve the dynamic response under these conditions. When there is no communication loss, the enhanced PID block acts exactly the same as a standard PID block. Lost data is compensated by the integral component in the enhanced PID block. When communications are reestablished, the derivative component in the enhanced PID block eliminates possible spikes in the output. We evaluate the enhanced PID algorithm under several wireless scenarios. The results demonstrate the advantages of the enhanced PID algorithm.

1. INTRODUCTION

A modern process control system is often structured as shown in Figure 1 [1]. As illustrated in this figure, sensors and actuators are connected to controllers via control networks, such as Fieldbus [2] and Profibus [4]. A sensor provides measurements and status of a physical property, e.g. the flow in a pipe associated with the process. Based on the measurement from sensors, the controller determines any adjustment in the actuators that is needed to maintain the process at a target value, i.e., the

Presented at ISA EXPO 2006, 17-19 October 2006, Reliant Center Houston, Houston, Texas

setpoint. The control loop is executed periodically at a rate fast enough to correct any unwanted deviations in the process.

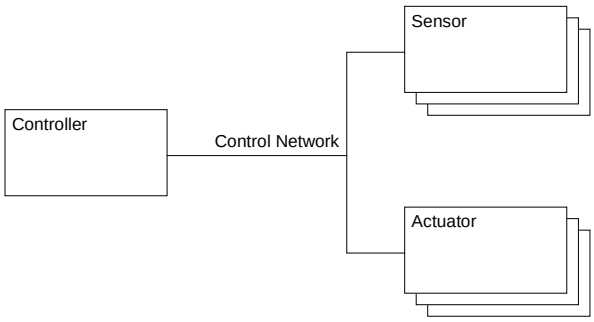

Figure 1. A process control system

In some applications, it is desirable to replace the wireline control networks with wireless networks [5, 6]. Some justifiable reasons include cost reduction and enhanced control. As a consequence, many standard efforts for wireless control are being actively pursued. For example, ZigBee is a wireless standard mainly targeted for industrial manufacturing and office automation. ZigBee Specification V1.0 was ratified in late 2004 and ZigBee compliant products are readily available on the market [7]. ISA [3] also has a very active wireless standard committee SP100. Suffice to say that wireless process control is becoming a reality.

In contrast to non-wireless control systems, communications in a wireless control system are inherently unreliable. This unreliability may be caused by interferences, power failures and environmental factors such as lightening storms. Unfortunately, current control strategies are based on the assumption of reliable communications. In standard process control systems, a missed IO communication is considered an error. Usually, a control loop is configured with a maximum number of lost IO communications, after which the loop declares failure and the values are set according to the fail-safe configuration.

Therefore, unreliable communications presents a very challenging problem for standard control paradigms. Consider a PID block with inputs from a wireless channel. Suppose the inputs are lost at time t_1 and reestablished at time t_2. The derivative component of the PID would cause a spike in the output at t_2. Also, from t_1 to t_2, the reset component may windup based on the error that existed at time t_1.

In this paper, we focus on the most widely used control blocks – PID blocks. We revise the calculation of the integral and derivative components of PID algorithms by detecting missed communications and compensate for it proactively. Simulation results on several wireless scenarios prove the superiority of the enhanced PID algorithm.

The rest of this paper is organized as follows. Section 2 analyzes the shortcomings of the standard PID block in dealing with intermittent communication losses. Section 3 describes the enhanced PID algorithm. Experiments and results are presented in Section 4. Section 5 concludes this paper.

2. The standard PID algorithm

PID is the most widely used control algorithm in industrial process control [9]. As shown in Figure 2, the controller compares the process variable (PV) with a reference setpoint (SP). The error is then processed to calculate a new output to bring the PV back to its desired SP [10].

PID stands for "proportional, integral and derivative" components of the algorithm. Each of the three components performs a different task and has a different effect on the functioning of a system. Their outputs are summed up to produce the system output.

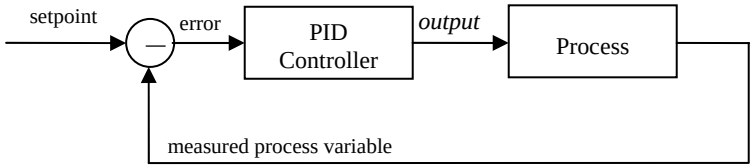

Figure 2. PID block

Though there are many variations of PID algorithms, in its non-interacting form without rate limiting and all actions based on error, the equation for standard PID algorithm is

$$Output = K_P\left[e(t) + K_I\int e(t)d(t) + K_D\frac{de(t)}{dt}\right]$$

K_P, K_I and K_D are the proportional, integral and derivative gains, respectively.

In its digital form, the software implementation of the PID algorithm is based on the sampled data for its PV being provided on a periodic basis.

When there is no communication loss, PID, once configured for the process it controls, keeps the process in a steady state. Figure 3(a) shows the PID reaction to a process disturbance.

Before time t_0 the PID output (*out*) is kept at a constant value to maintain *pv* at the *sp* value. At t_0 a drop of PV is observed due to process disturbances. To correct the drop, the PID increases its output. At time t_2, *pv* comes back at *sp*, and *out* is stabilized at some value that is slightly bigger than its original value to compensate the disturbance. As is shown in Figure 3(a), *out* is the sum of three parts: P, I, and D.

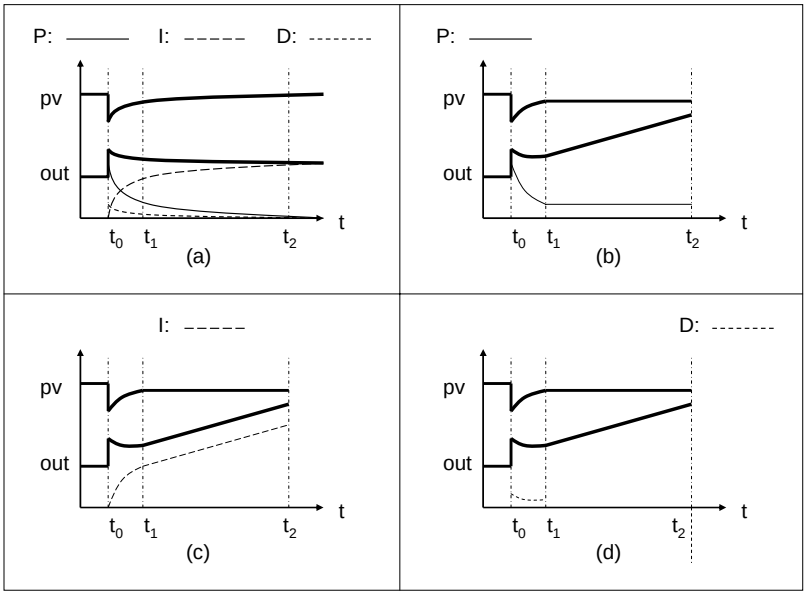

Figure 3. Standard PID block with lost input

2.1 Input communication lost

Now consider what would happen to each component of *out* if the communication from the sensor input is lost between times t_1 and t_2. For the PID block, the measured *pv* value remains the same as at time t_1 during this period, shown in Figure 3(b,c,d).

Figure 3(b) shows the proportional gain P. Since the measured *pv* and *sp* remains the same, the proportional gain is constant from t_1 to t_2. Figure 3(c) shows the integral part I. Since *pv* and *sp* remains the same, the error remains the same, so the integral part is a linear increasing line from t_1 to t_2. Figure 3(d) shows the derivative part D. Since *pv* and *sp* remains the same, the error remains the same, so the derivative stays 0 from t_1 to t_2. As a result, *out* of the PID block is a linear increasing line from t_1 to t_2, shown in Figure 3(b,c,d). This destabilizes the process. The longer the communication is lost, the bigger deviation *pv* is from *sp*.

Once the communication is reestablished at t_2, PID is back to normal. However, since the derivative part calculated at time t_2 is based on the difference of measured *pv*'s between t_2 and one period before t_2, we shall expect a spike for the derivative part. This is because the *pv* value at t_1 is used as the *pv* value at one period before t_2. The value of *pv* at t_2 could be significantly different from the *pv* value at t_1. Since *pv* changes between time t_1 and t_2, we expect the derivative spike even bigger. Due to sudden changes of the proportional and derivative parts, the value of *out* will have a big impulse before and after t_2.

2.2 Output communication lost

We continue to analyze how standard PID behaves if output communication is lost between t_1 and t_2. Here we assume there are no other disturbances and input communication is OK.

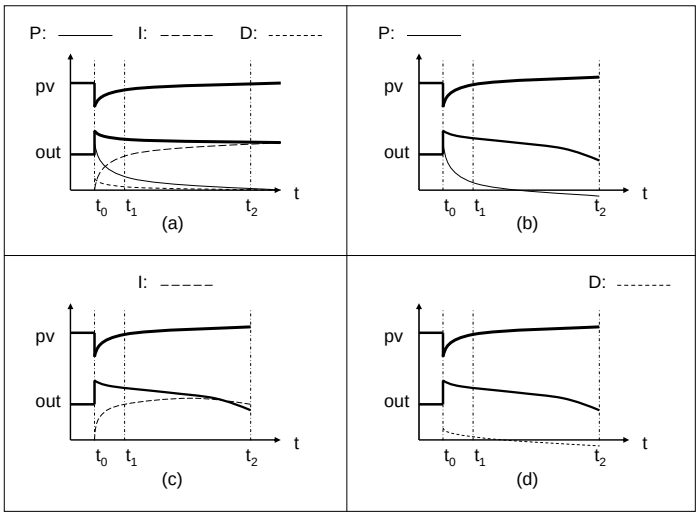

Figure 4. Standard PID block with lost output

Since t_1 the actuator will stay with the *out* value of t_1 until t_2 when a new *out* value is received from the PID. This will cause the measured *pv* to eventually reach *sp* and then overshoot a little bit, shown in Figure 4(b,c,d). The P, I, and D components are all calculated based on the current pv, shown in Figure 4(b,c,d). This is as good as we could expect from PID. The only drawback is that the actuator gets a bump in the *out* value, from the one calculated at t_1 to the one calculated at t_2.

2.3 Both input and output communication lost

When both input and output communications are lost, the PID behaves the same as when only input communication is lost in Figure 3. The only difference is that when communication is reestablished at t_2, the actual *pv* is different. In this case *pv* strays less as the actuator output stays constant. Similarly, the actuator receives a bump in the *out* value.

3. The enhanced PID algorithm

The underlying assumption in the digital implementation of the PID algorithm above is that the algorithm is executed on a periodic basis. When the input containing the measurement is lost, the calculated reset action may not be appropriate [1]. When later on a new measurement gets through, the calculated derivative action may produce a spike in the output. If a PID block continues to execute using the last process variable, the output will continue to move based on the reset tuning and error between the last measured process variable and the setpoint. If the control block is only executed when a new measurement is communicated, then this could delay control response to setpoint changes and feedforward action on measured disturbances. Also, when control is executed, calculating the reset contribution based on the scheduled period of execution or on the time since the last contribution may result in changes that increase process variability [1].

In [1], we proposed an enhanced PI algorithm to reduce wireless communications between sensors and controllers without impacting the control performance significantly. Based on that work, we further improve the derivative part and apply the new algorithm to address communication losses.

To provide best control when measurements are not updated on a periodic basis, the PID may be restructured to reflect the reset and derivative contributions for the expected process response since the last measurement update. One means of doing this is illustrated in Figure 5.

As shown in Figure 5, the reset/rate contributions (integral/derivative parts of the PID) are determined based on the use of a new value flag from the communications stack, the same idea as in [1]. To account for the process response, the filter output is calculated in the following manner when a new measurement is received:

$$F_N = F_{N-1} + (O_{N-1} - F_{N-1}) * \left(1 - e^{\frac{-\Delta T}{T_{Reset}}}\right)$$

where F_N = *New filter output*

F_{N-1} = *Filter output for last execution*

O_{N-1} = *Controller output for last execution*

ΔT = *Elapsed time since a new value was communicated*

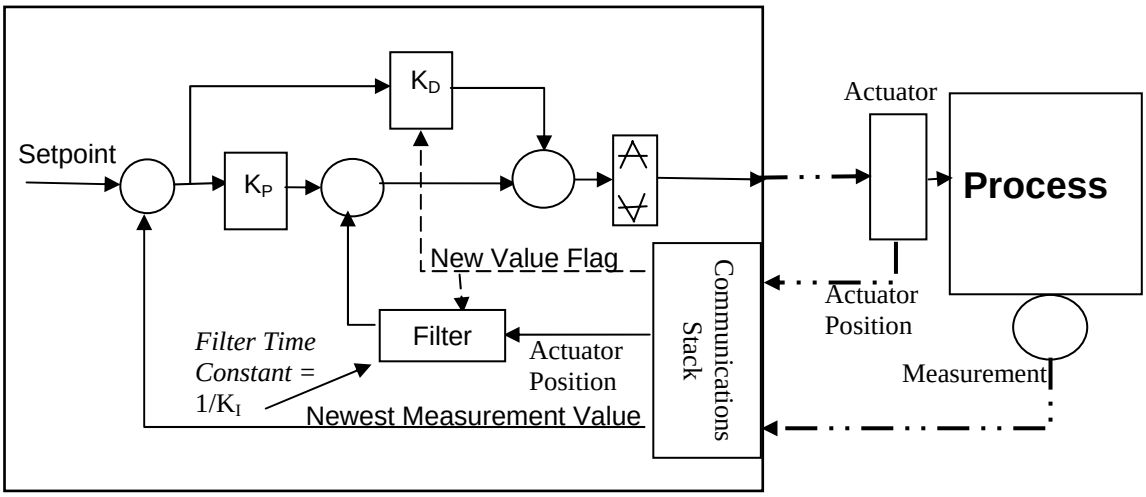

Figure 5. The enhanced PID algorithm application

Since the last communicated actuator position as reflected in the feedback of actuator position is used in the integral calculation, this automatically compensates for any loss in the output communicated to the downstream element.

The derivative part for this example (rate limiting not applied) is determined by the following equation:

$$O_D = K_D \bullet \frac{e_N - e_{N-1}}{\Delta T}$$

where e_N = *current error*

e_{N-1} = *last error*

ΔT = *Elapsed time since a new value was communicated*

O_D = *controller derivative term*

Consider the contribution of the derivative part when the inputs are lost for several periods. When the communication is reestablished, e_N - e_{N-1} in the equation above would be the same for the original and modified algorithms. However, for the standard PID algorithm, the divisor in the derivative part would be the period, while that in the new algorithm is the elapsed time between two successfully received measurements. It is obvious that the modified algorithm would produce smaller derivative action than the standard PID algorithm.

There are two major problems with standard PID algorithm when dealing with communication losses: continued execution during communication loss, and sudden output change when communication is reestablished. The enhanced PID algorithm solves these problems by only computing the integral and derivative components when communication is established and incorporating actuator feedback into the reset calculation.

4. Experiments and Results

We carried out several experiments to validate our algorithm. First, we prove that the new algorithm will produce the same result as the regular algorithm when the communication is reliable. Then, the revised algorithm is shown to have better performance than the existing PID algorithm when communications are unreliable.

4.1 Experimental Setup

We create two simple PID control loops, as shown in Figure 6.

PROC_1 and PROC_2 are two identical processes, each of which consists of a second order process with a delay of 1 second and time constants of 6 seconds and 3 seconds.

The modified PID algorithm is implemented in PIDPLUS, and PID2 is a standard PID block. The parameters for PID2 are determined by testing it with a tuning application, which suggests a gain of 0.85, a reset of 10.71 and a rate of 1.71. Then the tuning parameters of PIDPLUS are set the same as PID2. PIDPLUS is configured to utilize the BKCAL_IN value for the reset components.

The process variable communication is simulated by the COM_IN_1 and COM_IN_2 block, which are controlled by COM_STATUS_IN. If COM_STATUS_IN is set to 1, block COM_IN_1 and COM_IN_2 relay measurements accurately. Otherwise, the two blocks drop the measurements. The same logic is applied to the output using COM_OUT_1, COM_OUT_2 and COM_STATUS_OUT.

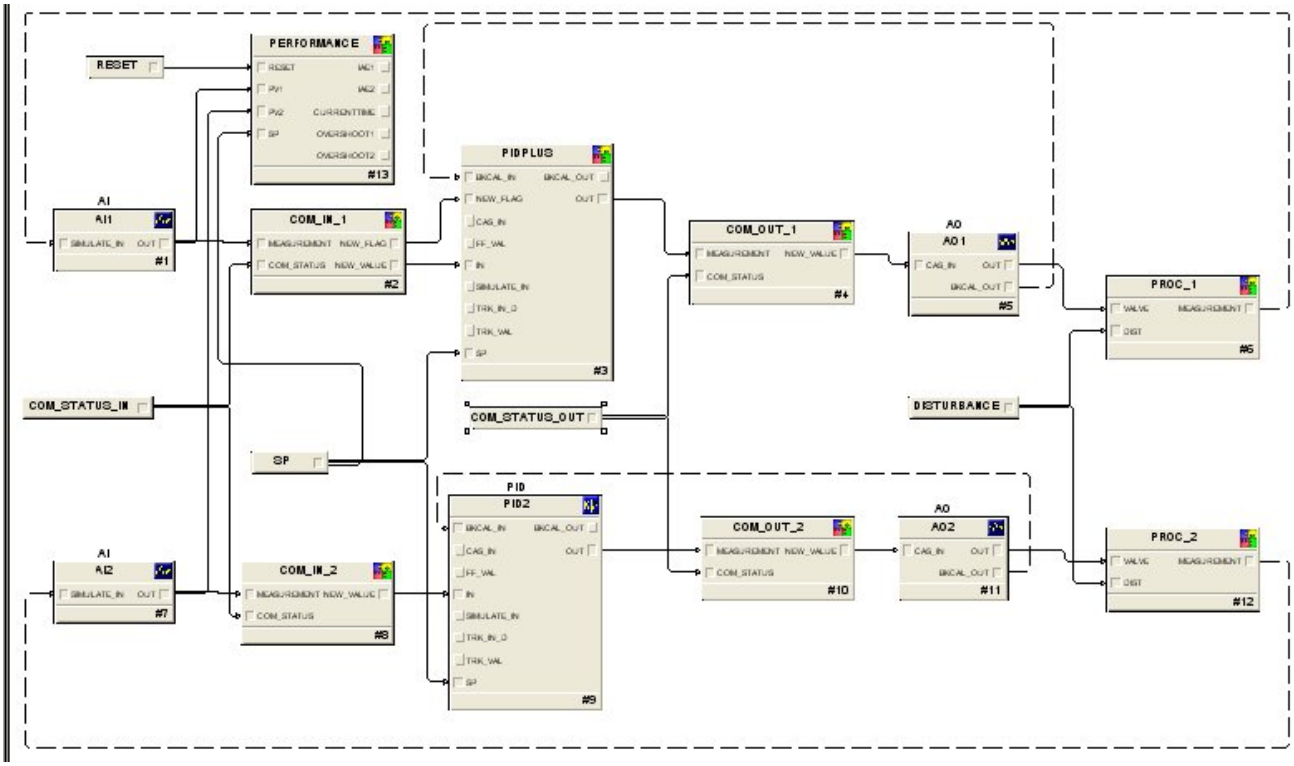

Figure 6. Experimental Setup

By changing the external setpoint and introducing some disturbances that impact each PID and associated process equally, we can evaluate the performance of the two PID blocks. The performance of the two PID blocks is collected in the PERFORMANCE block. The metrics used in this paper is Integral Absolute Error (IAE).

The scan rate for all blocks is set to 0.2 second. Initially, the uncontrolled disturbance (DISTURBANCE) to the processes is 20.

4.2 Reliable Communications

To simulate the case of reliable communications, we set both COM_STATUS_IN and COM_STATUS_OUT to 1. SP is changed from 50 to 60. The result is shown in the left part of Figure 7 (from time 11:10 to 11:11).

The curve of AI1/OUT.CV matches nicely with that of AI2/OUT.CV, and AO1/SP.CV matches nicely with AO2/SP.CV. Therefore, we conclude that the two PID blocks work in the same way when the communication is reliable.

4.3 Unreliable Communications

To study the effect of unreliable communications on the two PID blocks, we consider two cases: unreliable input and unreliable output.

4.3.1 Unreliable Input

During the period of lost inputs, the last communicated process variable is maintained and used in the PID blocks. We first experiment with changing setpoints. The result is shown in the right part of Figure 7, where SP is decreased from 60 to 50.

When the input channel is shut down, the error between the setpoint and process variable fed to PID blocks is a constant. For the standard PID block, PID2, the integral part would keep integrating, which explains the linear decreasing of AI2/OUT and AO2/SP. However, as it has a flag for missed communications, PIDPLUS would simply freeze the reset component during loss of communication, which explains the level-off in AO1/SP during lost communications. Since AO1 is given a constant value, the process variable of PROC_1 approaches the setpoint gradually, as shown by AI1/OUT.

When communications are re-established, the input process variables to PIDPLUS/PID2 reflect the true measurement provided by AI1/AI2, respectively. For PID2, AI2/OUT is very low compared to the setpoint, which causes a sharp spike in AO2 by the derivative part of PID2 at the moment communications are re-established. For PIDPLUS, AI1/OUT is close to the setpoint, and that small deviation is further evened out by the divisor used in the derivative part of the new algorithm. Thus both AI1 and AO1 transit to their steady states smoothly.

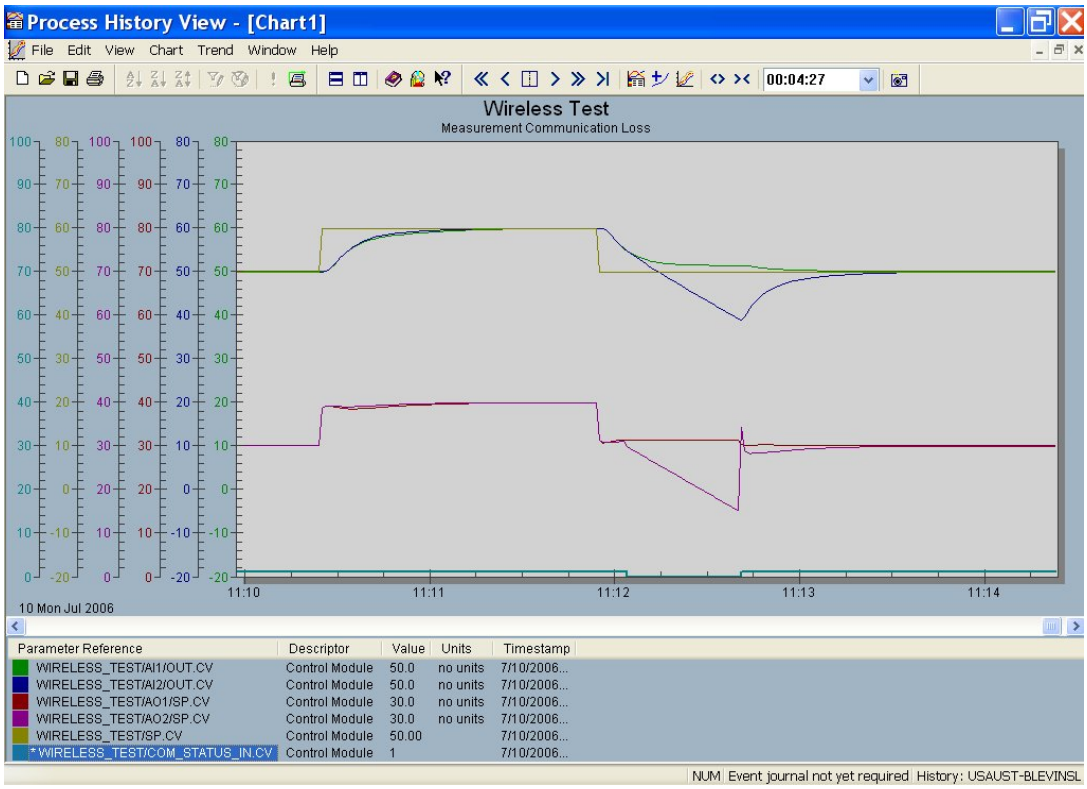

Figure 7. Lost Inputs coupled with a setpoint change

The different behaviors of the two PID bocks are further proved by the performance data. In a duration of 121 seconds, the IAE for PIDPLUS is 169, while that for PID2 is 372.

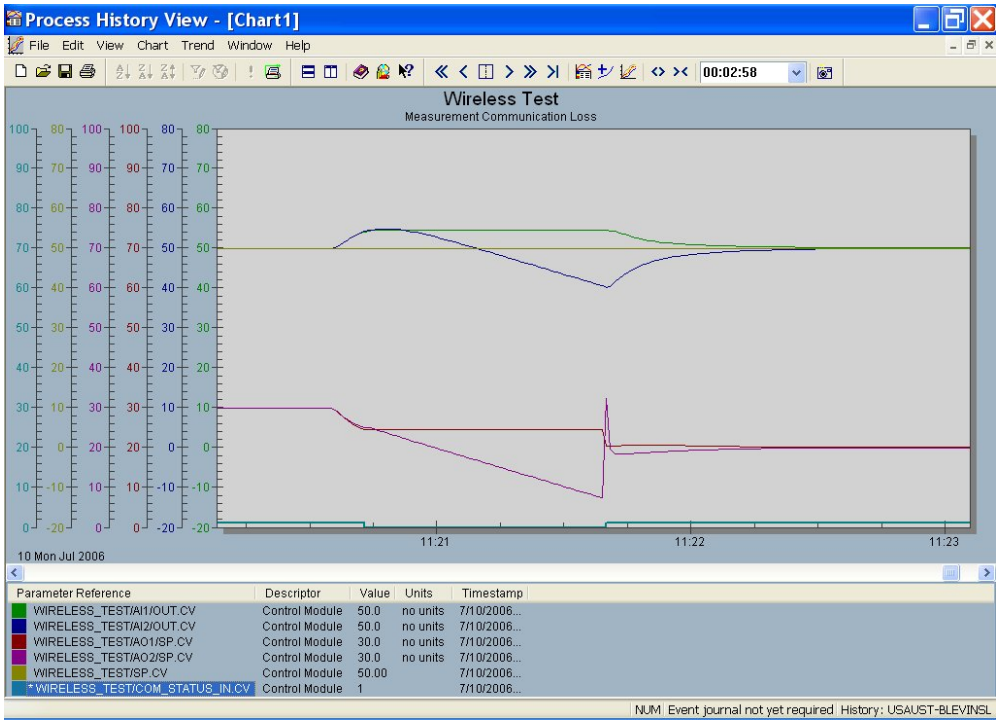

Figure 8. Lost Inputs coupled with unmeasured disturbances

We also test the two PID blocks with unmeasured disturbances. The DISTURBANCE is increased from 20 to 30. The result curves are shown in Figure 8. The behavior of PID2 is the same as in Figure 7, which can be explained by the same reason for Figure 7. For PIDPLUS, the reset component is maintained constant during the loss of communications. Thus AO1 is kept the same output value, which in turn produces the same AI1 value. When the input communications resume, PIDPLUS starts to change its output (AO1/SP) to bring AI1/OUT back to the setpoint. During the time (196 seconds), the IAE for PIDPLUS is 333, and that for PID2 is 366.

4.3.2 Unreliable Output

In this case, we examine the behaviors of the two PID blocks in two scenarios too: setpoint changes and uncontrolled disturbances.

For setpoint changes, we first change the setpoint from 50 to 60, when the communication is reliable. Then, after the processes settle at the setpoint 60, the setpoint is changed back to 50, and the output channels are cut off. Figure 9 shows the resulting curves.

When the communication is lost, the outputs of PIDPLUS/PID2 are equal. Since the two controlled processes are the same, the values of AI1.OUT and AI2.OUT follow the same curve. When the communication is reestablished, the input errors for PIDPLUS and PID2 are the same. However, the divisor in the derivative part of PIDPLUS is much bigger than that in PID2, which explains the sharper spike in the curve for AO2/OUT. During the transition period, the IAE for PIDPLUS is 190, while that for PID2 is 196.

Again, we test the two blocks by introducing uncontrolled disturbance to the processes. The DISTURBANCE is changed from 20 to 30. The results are shown in Figure 10.

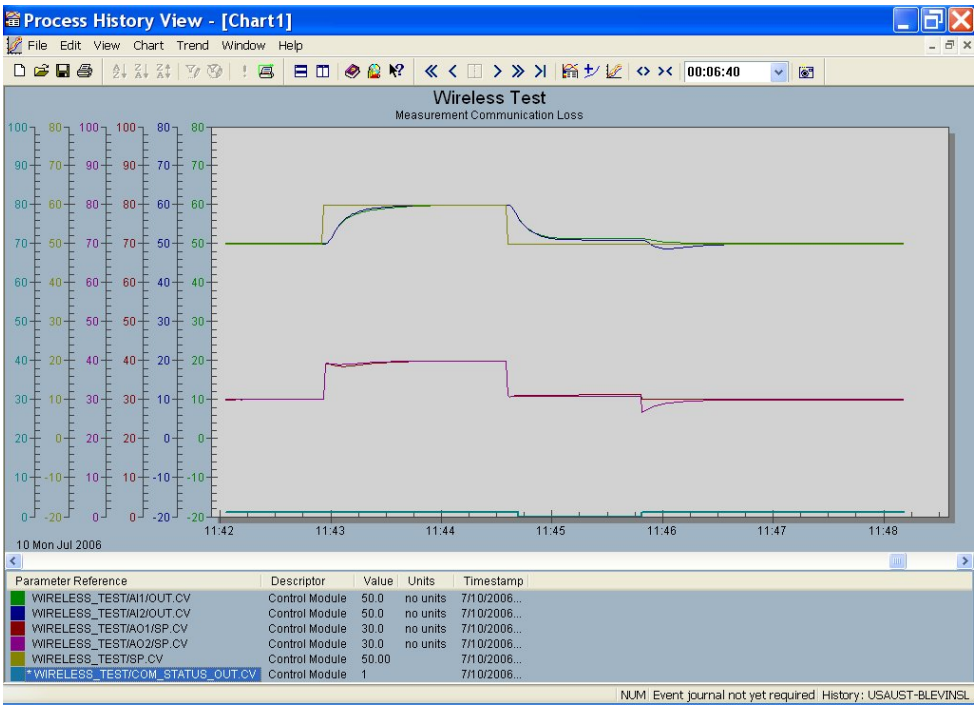

Figure 9. Missed Outputs with a setpoint change

The curves in Figure 10 look similar to their counterparts in Figure 9 and they can be explained the same way as above. We also notice that the improvement of PIDPLUS over PID2 is more pronounced in Figure 10 than in Figure 9. This is because at the time of communication reestablishment, the input errors to the PID blocks are bigger in Figure 10 than in Figure 9.

During the transition period (158 seconds) in Figure 10, the IAE for PIDPLUS is 267, while that for PID2 is 388.

5. Comments and Conclusions

Using actuator feedback for smooth output transition is not a new idea. The Foundation standard already defines back-calculate-in/out links to smoothen transition during start up. Certain control systems continue to use this link after startup.

PID blocks in modern control systems have status flags for data values. In this way a data value could be tagged with communication lost status. Some systems provide fail-safe mechanism by making use of this flag. For example, Foundation Fieldbus standard allows a limited number of communication failures, after which error is declared and the block enters failure state. This in turn forces the actual block mode to manual during communication failure. This approach alleviates the problem associated with communication loss but does not eliminate them. The proposed modifications to the PID to compensate for communication loss differs in that we explicitly address communication failures and take advantage of the related information.

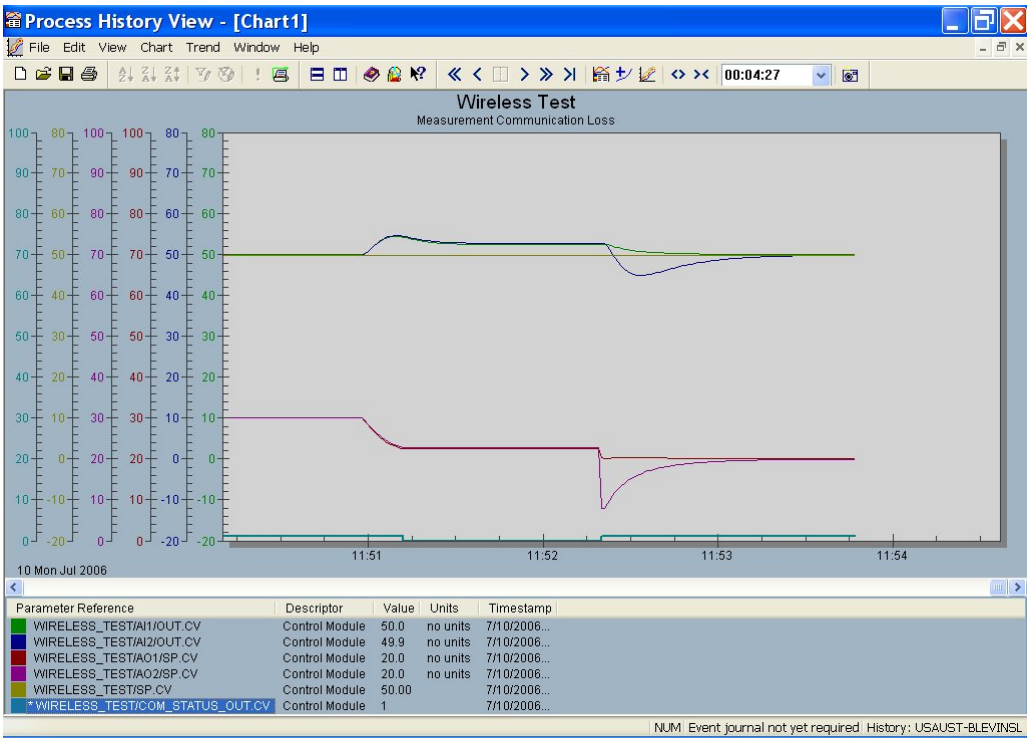

Figure 10. Missed Outputs with unmeasured disturbances

Current control designs assume periodic samplings. However, this assumption does not hold in a wireless environment. In this paper, in order to cope with possible measurement lost, we propose a modified PID algorithm. The enhanced algorithm acts in the same manner as the standard PID algorithm when the communication is reliable. When it detects any communication lost, this algorithm can smoothen possible spikes in the output. We validate the new algorithm with several experiments in industrial DCS.

REFERENCES

1. Deji Chen, Mark Nixon, Tom Aneweer, Rusty Shepard, Terry Blevins, Greg McMillan and Aloysius K. Mok, Similarity-based Traffic Reduction to Increase Battery Life in a Wireless Process Control Network, *ISA EXPO 2005*, Chicago, USA.
2. Foundation Fieldbus standard, http://www.fieldbus.org/.
3. The Instrumentation, Systems and Automation Society, http://www.isa.org/.
4. Profibus standard, http://www.profibus.org/.
5. Wireless Industrial Networking Alliance, http://www.wina.org/.
6. ZigBee Alliance, http://www.zigbee.org/.
7. Chipcon Products, http://www.chipcon.com/.
8. DeltaV digital control system, http://www.easydeltav.com/.
9. Control Information from John Shaw, http://learncontrol.com/pid/
10. PID controller, http://en.wikipedia.org/wiki/PID_controller