Performance Characterization of IP Network-based Control Methodologies for DC Motor Applications – Part II

Tyler Richards, Mo-Yuen Chow
Advanced Diagnosis Automation and Control Lab
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695, USA
trichar@gmail.com, chow@ncsu.edu

Fen Wu
Department of Mechanical Engineering
North Carolina State University
Raleigh, NC 27695, USA
fwu@ncsu.edu

Abstract – Using a communication network, such as an IP network, in the control loop is increasingly becoming the norm. This process of network-based control (NBC) has a potentially profound impact in areas such as: teleoperation, healthcare, military applications, and manufacturing. However, limitations arise as the communication network introduces delay that often degrades or destabilizes the control system. Four methods have been introduced in Part I of these two companion papers that alleviate the IP network delays to provide stable real-time control. Part II of these papers defines a performance measure for a case study on a DC motor with a networked proportional-integral (PI) speed controller with various network delays and noise levels. Simulation results show that NBC combined with these techniques can successfully maintain system stability. This allows for control of real-time applications.

I. INTRODUCTION

Network-based control (NBC) uses a communication medium in the control loop. The introduction of the communication medium, such as the Internet, also introduces network delay. This delay can degrade performance or destabilize the system. Real-time applications require an improved NBC algorithm such that performance can be guaranteed. Part I of these two companion papers [1] has introduced four methodologies that can be used with NBC for time sensitive applications.

A case study on a DC motor with a networked proportional-integral (PI) controller is investigated in this paper. The performance of each method is compared in the presence of various network delays and noise levels.

Network-based control systems experience network delay from the controller to plant and from the plant to controller. Let us denote the time delay experienced from the controller to the actuator of the plant as $\tau_c$, and the time delay from the sensor of the plant to the controller as $\tau_p$. For this paper, the delays associated with processing time are assumed constant and much smaller than $\tau_c$ and $\tau_p$, and therefore can be lumped into those terms to simplify the problem.

This paper is outlined as follows: Section II presents the DC motor plant and performance measure; Section III shows the simulations results; Section IV reports noise results; Section V concludes the paper.

II. CASE STUDY: DC MOTOR

A. System Description

The DC motor is governed by the following equations:

\[ \begin{align*}
\dot{x}_1 &= \frac{-R}{L} \dot{i}_a - \frac{K_a}{L} \omega_c + \frac{1}{L} e_a \\
\dot{\omega}_c &= \frac{K}{J} \left( i_a - \frac{B}{J} \omega_c + \frac{1}{J} F \right)
\end{align*} \tag{1} \]

where $e_a$ is the armature winding input voltage, $L$ is the armature winding inductance, $R$ is the armature winding resistance, $i_a$ is the armature winding current, $K_a$ is the back EMF constant, $K$ is the torque constant, $J$ is the moment of inertia of the wheel and rotor, $B$ is the damping coefficient of the wheel and rotor, $\omega_c$ is the angular velocity of the wheel, and $F$ is the load torque. Table I shows the values used during simulation for these constants that were modified from [2].

The sampling time, $T$, for the controller and plant is chosen to be 0.1ms. This is to capture the dynamics of the electrical portion of the DC motor, whose time constant is:

\[ \gamma_c = \frac{L}{R} = 4.48 \text{ msec} \tag{2} \]

The load torque is assumed to be zero during the simulations. (1) can be arranged into the standard state space form shown below:

\[ \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \frac{-R}{L} & -\frac{K_a}{L} \\ \frac{K}{J} & -\frac{B}{J} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1/L \\ 0 \end{bmatrix} u \tag{3} \]

During the simulations, the plant is considered to be described by (3) that is sampled every $kT$ seconds. The plant output is assumed to be fully observable.

The authors will use a PI controller that has been tuned for nominal performance in the non-delay case for the controller of the system. The optimal PI gains for the non-delay case are determined by the root locus approach and must satisfy the following design requirements:

- Relative damping ratio: $\zeta = 0.707$;
- Percentage overshoot: $P.O. \leq 5\%$;
- Settling time: $t_s \leq 0.0321$sec;
- Rise time: $t_r \leq 0.014$sec.

Using the above requirements, $(K'_1,K'_2) = (25.572,0.995)$

### TABLE I. DC MOTOR PARAMETERS.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$</td>
<td>2.55e-3 N-m/A</td>
</tr>
<tr>
<td>$R$</td>
<td>6.43 $\Omega$</td>
</tr>
<tr>
<td>$K_b$</td>
<td>.255e-3 V-sec/rad</td>
</tr>
<tr>
<td>$L$</td>
<td>28.8e-3 $H$</td>
</tr>
<tr>
<td>$B$</td>
<td>0.1e-3 N-m-sec/rad</td>
</tr>
<tr>
<td>$J$</td>
<td>3.53e-6 $Kg-m^2$</td>
</tr>
</tbody>
</table>
will satisfy the conditions above. This will be used as the baseline performance to compare to, also referred to as the nominal controller. The reference speed is set to \( r(k) = 1 \text{ rad/sec} \).

### B. Performance characterization

In order to characterize the performance of each methodology, the authors adopt the following performance measure [3]:

\[
J = w_1 J_1 + w_2 J_2 + w_3 J_3 , \quad (5)
\]

where

\[
J_1 = \begin{cases} 
(MSE - MSE_0)^2 , & MSE > MSE_0 \\
0 , & MSE \leq MSE_0 
\end{cases} , \hspace{0.5cm} (6)
\]

\[
J_2 = \begin{cases} 
(P.O. - P.O._0)^2 , & P.O. > P.O._0 \\
0 , & P.O. \leq P.O._0 
\end{cases} , \hspace{0.5cm} (7)
\]

\[
J_3 = \begin{cases} 
(t_r - t_{r_0})^2 , & t_r > t_{r_0} \\
0 , & t_r \leq t_{r_0} 
\end{cases} , \hspace{0.5cm} (8)
\]

\[
MSE = \frac{1}{N} \sum_{k=0}^{\infty} e^2(k) \quad (9)
\]

is the mean-squared error, \( MSE_0 \) is the nominal mean-squared error, \( P.O._0 \) is the nominal percentage overshoot, \( t_{r_0} \) is the nominal rise time. The weights \( w_1, w_2, \) and \( w_3 \) specify the relative significance of \( J_1, J_2, \) and \( J_3 \), respectively, on the overall system performance. The error, \( e(k) = y(k) - r(k) \), is computed by sampling \( y(t) \) every \( kT \) time instants. The costs \( J_1, J_2, \) and \( J_3 \) penalize any degradation in performance from the nominal case. The nominal performance measures that meet the design specification are: \( MSE_0 = 0.003405 \), \( P.O._0 = 5\% \), and \( t_{r_0} = 0.014 \). \( MSE_0 \) is determined by simulation or by experiment. In the nominal non-delay case, the cost function in (5) will equal zero. \( J_1 \) penalizes tracking error. \( J_2 \) penalizes higher overshoot. \( J_3 \) penalizes the slower response time. The weights \( w_1, w_2, \) and \( w_3 \) are found by simulation using the GSM method with \( \beta = [0, 2] \) and no delay using the following equation:

\[
w_i = \frac{1}{\max(J_i) - \min(J_i)} \quad \text{for } i = 1, 2, 3 . \hspace{0.5cm} (10)
\]

When \( \beta = 1 \), the nominal gain is applied to the system and therefore the cost \( J = 0 \). Therefore, \( \min(J_i) = 0 \) for \( i = 1, 2, 3 \) and (10) reduces to

\[
w_i = \frac{1}{\max(J_i)} \quad \text{for } i = 1, 2, 3 . \hspace{0.5cm} (11)
\]

For the DC motor parameters listed in Table I, \( w_1 = 17.4572, w_2 = 0.0066771, \) and \( w_3 = 0.8777 \).

### C. Network Delay

The communication medium is assumed to be an IP network with TCP/IP protocol. Each methodology has its own specific way of modelling the network delay experienced on the system. In general, network delay will be modelled as an exponential distribution unless otherwise noted. The generalized exponential distribution to describe IP network delays is as follows [3]:

\[
P[\tau] = \begin{cases} 
\frac{1}{\phi} e^{-(\tau - \eta)/\phi} , & \tau \geq \eta \\
0 , & \tau < \eta 
\end{cases} , \hspace{0.5cm} (12)
\]

where \( \eta \) is the minimum transportation delay of the application. The expected value of the RTT delay is \( E(\tau) = \phi + \eta \) with variance \( \sigma^2 = \phi^2 \), and median, \( \eta \).

Keeping in mind the DC motor’s time constant denoted in (2), the authors have chosen \( \eta = [0.0001, 0.001, 0.005, 0.01, 0.05, 0.1] \) to simulate the RTT network delay and be representative of a variety of minimum network delays possibly experienced. The first two delays should have smaller effects on the system performance as they are less than \( \gamma_c \) found in (2), while the larger delays should cause more rapid system degradation.

### III. SIMULATION RESULTS

For all simulations, the final time, \( t_f \), is 2 seconds, with a sampling time, \( T \), of 0.0001 seconds. Matlab 7.0 was used to run the simulations. The following sections will present the results for each of the four methods in alphabetical order when applied to the DC motor model described previously.

#### A. Gain scheduling middleware (GSM)

1) DC Motor Application

GSM will be applied on the networked PI controller for a DC motor. The transfer function of the PI controller is given as:

\[
G_c(s) = \frac{K_p(s + z_c)}{s} , \hspace{0.5cm} (13)
\]

where \( z_c = K_c/K_p \) is a constant. The optimal PI gains for the non-delay case are determined by the root locus approach and satisfy the following design requirements as shown in (4). \( (K'_c, K'_p) = (25.572, 0.995) \) will satisfy the requirements. With this choice of gains, \( z_c = 25.7 \). The response at this gain setting will be the baseline performance to compare to. Therefore, the cost function (5) will be minimal for this point.

Let \( \beta \) be the gain scheduling parameter that will modify the gain. In this case, \( \beta \) will modify the PI gain in the following way:

\[
\beta G_c(s) = \frac{\beta K_p(s + z_c)}{s} . \hspace{0.5cm} (14)
\]

By keeping \( z_c \) a constant, the optimization of \( \beta \) becomes a one-dimensional problem rather than a two-dimensional problem where \( K_c \) and \( K_p \) were allowed to vary independently.

The optimal \( \beta \) for a given delay is found by minimizing the
cost function given in (5). This can be done for several expected delays. GSM can then use a look-up table to select the appropriate \( \beta \) for the delay measured by the network traffic estimator. Table II shows the optimal \( \beta \) for the delays presented previously using the cost function in (5).

2) Results

As the delay time increases, the optimal \( \beta \) decreases. The system responds slower in response to an increase in network delay. Also, the cost increases as the delay increases as shown in Fig. 1. The step response of the nominal PI controller in the presence of the set of network delays, \( \eta \), is shown in Fig. 2.

As the delay increases past 0.01 secs, the PI controller is unable to stabilize the system. The GSM results for the same set of \( \eta \) is shown in Fig. 3. Using the optimal \( \beta \) value found offline, the GSM method is able to maintain system stability at the cost of slower response time. The responses with GSM have fewer oscillations as they approach the reference value. In all cases, the GSM method is able to stabilize to the reference value, while the nominal PI controller fails to stabilize for the 0.05 and 0.1 sec delays.

Table III shows the performance comparison between the nominal case and all four methods. For all delays larger than 0.0001 seconds, GSM outperforms the uncompensated nominal PI controller.

B. Optimal stochastic

1) DC motor application

The constraining assumption of the optimal stochastic method is that \( \tau^c + \tau^a < T \). Since \( \tau^c \) and \( \tau^a \) must be measured and used in the calculation of the LQG gain, timestamps are added to each packet in order to measure the delay experienced.

The standard LQG problem regulates the plant output around zero. In this case, the authors would like the system to track a reference input. In order to formulate the regulator as a tracking problem, the output, \( y(k) \), must be compared to the reference signal, \( r(k) \). The goal is then to drive the error between the output and the reference to zero. A common practice is to add an integrator to the error signal, \( e(k) = y(k) - r(k) \), to drive it to zero[4]. The reformulated LQG problem can then be solved using the Matlab Control System toolbox.

For the DC motor, the control objective is to minimize the cost function:

\[
J(u) = \int_0^T (x^T Q x + u^T R u + 2 x^T N u) dt, \tag{15}
\]

where \( Q = diag(1e6,10,10) \), \( R = 1 \), and \( N = 0 \), for the augmented plant description with the integrator on the in error signal, where the states are defined as \( x = [e,i,o]^T \). \( Q \), \( R \), and \( N \) are determined experimentally by finding the LQR gain whose performance best mirrors that of the PI controller during a non-delay case trial. This puts the largest penalty on the error while also keeping the control values within the acceptable limits of \( \pm 12V \). The white noises \( v(t) \) and \( w(t) \) are assumed to be zero in order to properly compare the performance of each methodology. The assumption is that the network delay is the dominating factor of the system performance and stability.

Using the problem formulation above, the LQG controller is designed without taking any delays into account. This yields a gain \( L = [-1000, 12.404, 7.2913] \).

![Fig. 1. Cost vs. \( \beta \) for all network delays.](image1)

![Fig. 2. Nominal PI controller for all network delays.](image2)

![Fig. 3. GSM step response for all network delays.](image3)

<table>
<thead>
<tr>
<th>Delay Time (sec)</th>
<th>0.0001</th>
<th>0.001</th>
<th>0.005</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal ( \beta )</td>
<td>1.008</td>
<td>0.93</td>
<td>0.535</td>
<td>0.355</td>
<td>0.095</td>
<td>0.005</td>
</tr>
</tbody>
</table>

Table II. Optimal \( \beta \) gains for various time delays.
2) Results

Table III shows the comparison of (5) between the nominal PI controller and the LQG controller. The step responses are shown for the LQG controller in Fig. 4. It performs very well for 0.0001 sec delay since the major assumption of $\tau_\alpha + \tau_\omega < T$ is met. Notice that it does manage to lower the cost at the highest delays of 0.05 and 0.1 seconds. The performance degrades faster than the nominal case since $T$ is held constant at 0.0001, while the delay is increased. This invalidates the original assumption.

C. Queuing methodology

1) DC motor application

Since full state information is available, the observer block shown in the Part I paper is omitted. Thus for a $\delta$-step delay, where $\delta$ is the delay measured in terms of the number of sample times, $\delta$ predictor blocks will be needed. For example, the 0.001 RTT delay will require $0.001/0.0001$ or 10 predictor blocks. It should be noted that the queuing method depends on the accuracy of the model and observer. Since the model is accurate and no observer is used, we should expect to see very good results.

2) Results

The results of the queuing method are shown in Fig. 5. Note that the response for each delay overlaps each other. This is because of the perfect model and full state information. Table III shows the costs associated with the queuing methodology for the different delays. As noted before, because of the perfect model accuracy, the predictor block is able to predict the states perfectly regardless of the delay. However, the queue length increases as the delay increases, as shown in Table IV.

The queuing methodology works well when the delay is bounded by a constant and the plant model is known. Also, it is more feasible when the sample time is close to the delay time to limit the number of predictor blocks required. These multiple calculations required for the predictor blocks could introduce more delay than actually experienced if done in real-time.

D. Robust Control

1) DC motor application

$G$ represents the plant dynamics and are given for the DC
motor in (3). These will be used to form the interconnection matrix $P$. By experimentation, the performance weight, $W_i$, was found to be:

$$W_i(s) = \frac{25}{s^2 + 20s + 1}.$$  

(16)

This gives satisfactory performance for all delays. $W_2$ and $W_i$ are given in [1].

Next the system is analyzed using the $dkitgui$ tool in Matlab. This tool uses DK iteration to generate controllers for the system and analyze the robustness. For the DC motor, the frequency range 0.001 to 1000 rad/sec with 1000 points was used. The uncertainty structure is a 2x2 matrix. Eight iterations were completed for each delay time, generating eight controllers. Each one was tested for optimal performance based on the cost function in (5). The best one was chosen for the final trials. The controller chosen also had the lowest peak $\mu$ value given by the $dkitgui$ tool. A lower $\mu$ value indicates a smaller effect of a disturbance (i.e. network delay) on the system.

Overall, the robust methodology does a great job as compensating for network delay. The major benefit is that the controller does not need a priori knowledge of the network delays, nor does it need the past delays as many of the other methods need. It does, however, require that the weights be carefully set such that they cover the uncertain delays.

2) Results

The optimal robust controller for each delay is shown in Table V. The controller number indicates during which iteration the optimal performance, or lowest cost, was achieved. Thus for the first delay of 0.0001 sec, the 4th D-K iteration controller achieved the best results.

The step responses are shown in Fig. 6. The cost for the 0.0001 sec delay is larger than the 0.01 sec delay because the D-K iteration tries to optimize its control to the $H_\infty$ performance measure that differs from the one given in (5). Aside from that result, the cost increases as the delay time increases. On the whole, the robust method compensates for the delays as shown by the decreased costs from the nominal cases.

IV. NOISE

The previous results were conducted under ideal conditions. The authors have also investigated the performance of each method under noisy conditions. Noise was introduced into the control loop at the sensor measurement to simulate modeling errors (both state and measurement). The noise is Gaussian white noise with the zero mean and a given variance, $v(k) = G(\mu, \sigma^2)$. The mean, $\mu$, is set to zero and the variance, $\sigma^2$, was chosen so that the distribution effects between 0 and 10% of the final reference value of 1 rad/sec. This was determined by $\Psi = 3\sigma$, where $\Psi$ stands for the noise level with respect to the final value, and $\sigma$ is the standard deviation. $3\sigma$ was chosen as it accounts for 99.7% of a normal distribution. Thus for

$\eta \in \{0.5\%, 1\%, 2\%, 5\%, 10\%\}$, the variance used is:

$$\sigma^2 \in \{2.7778e-006, 1.1111e-005, 4.4444e-005, 0.0027778, 0.0011111\}.$$  

(17)

The performance is measured with the same cost function described earlier in (5). It is expected that the introduction of noise will decrease the performance of the system. As the noise increases, the performance should decrease which will be indicated by a higher cost measured by (5).

In general, as the delay and noise increase so does the cost, $J$. The queuing and GSM methods have the lowest cost with respect to noise or network delay.

A. GSM

Fig. 7 shows the performance of the GSM methodology with noise. For small delays (i.e. under 0.01 sec), the noise had a larger effect on the performance than for larger delays. In the case of larger delays, the noise had a much smaller effect on the performance than the delay, as shown by cost values very similar regardless of the noise level. However, the total cost at even the worse condition is still much lower than the nominal PI controller.

B. Optimal

Fig. 8 presents the performance of the optimal stochastic method with noise. As with the GSM method, the noise effect is more apparent at low delays, 0.001 sec and smaller, than at higher delays. This is partly due to the noisy results of the optimal method even without noise, as shown in Fig. 4. Those results are due to the fact that $\tau^w + \tau^o \leq T$, where $T = 0.0001$ sec in this case. Therefore, additional noise will have little effect on the performance of the optimal methodology. However, for where $\tau^w + \tau^o \leq T$ holds true, increased noise causes an increase in cost which indicates a decrease in performance.

C. Queuing

Fig. 9 presents the queuing method’s results with respect to noise levels and network delay. As mentioned earlier, because of the high accuracy of the model, the queuing method is able
to accurately predict the plant’s state regardless of the delay length. Thus an increase in delay has no effect on the performance, as the cost does not rise. However, as noise is introduced, the inputs to the predictor blocks are no longer accurate. This inaccurate prediction results in an increase in the cost as the noise level increases. Therefore, system performance degrades as noise levels increase. Similar to the GSM method, the total cost of the queuing method at the worst condition is still low compared to the nominal PI controller.

D. Robust

Fig. 10 shows the performance of the robust methodology for various noise levels and network delays. Unlike the other methods, regardless of the delay an increase in the noise level resulted in an increase in the cost, just as an increase in the network delay resulted in an increase in the cost. Thus, performance degrades as both the network delay and noise levels increase.

V. CONCLUSION

Four methodologies were investigated for use with real-time applications in a NBC setting. The simulation results have shown promising results to apply these methodologies on real-time network-based control systems. The GSM and queuing methods presented the best performance for various delays and noise levels, as compared to the networked nominal PI controller. The robust methodology also showed a significant improvement over the nominal PI controller. The optimal stochastic method showed increased performance over its limited range set by its delay assumption. Between the top two performers, GSM has considerable off-line calculations that must be done in order to schedule the gain for the network traffic conditions, while the queuing method depends significantly on the plant model’s accuracy. For critical real-time applications, the appropriate method must be chosen based on the application. All methods have shown the ability to stabilize the closed loop control system and minimize the detrimental delay effect in an IP network environment.

VI. REFERENCES