Random delay effect minimization on a hardware-in-the-loop networked control system using optimal fractional order PI controllers

Varsha Bhambhani *, Yiding Han *, Shayok Mukhopadhyay *, Ying Luo * and YangQuan Chen *

* Center for Self-Organizing and Intelligent Systems (CSOIS), Electrical and Computer Engineering Department, Utah State University, Logan, UT 84322 – 4160, USA

Abstract:
Random delays have serious effects in networked control systems, which deteriorate the performance and may even cause instability of the system. Hence a controller which can make the plant stable at large values of delay is always desirable in NCS systems. Our previous work on OFOPI controller showed that fractional order PI controllers have larger jitter margin (maximum value of delay for which the system is stable) for lag-dominated systems as compared to traditional PID controllers, whereas integer order PID controllers have larger jitter margin for delay-dominated systems. In this paper, a telepresence controller based on optimal fractional order proportional and integral (OFOPI) tuning rules is used to obtain the maximum value of delay (jitter margin) at which the system will be stable. To illustrate this, an extensive experimental study on the real-time Smart Wheel networked speed control system is performed using hardware-in-the-loop control. For this purpose, the real-time random delay in the world wide network is collected by pinging different locations, and is considered as the delay in our simulation and experimental systems. Comparisons are made with existing integer order PID controller. It is observed that the proposed OFOPI controller is a promising controller and has faster response time than the traditional integer order PID controllers. Also it is verified that since the plant into consideration viz. the Smart Wheel is a delay-dominated system, PID achieves larger jitter margin as compared to OFOPI tuning rules. Simulation results are presented to illustrate the effectiveness of the proposed OFOPI method.

Keywords:
networked control system, optimum fractional order proportional integral controller, Smart Wheel, random delays, hardware-in-the-loop.

1. INTRODUCTION

A networked control system is a feedback control system in which the control loop is closed through a communication network [Beldiman, 2001]. In recent years, due to its cost-effectiveness and flexible applications, the research on the use of a data network in a control loop has gained increasing attention [Tipsuwan and Chow, 2003, Zhang et al., 2001, Lian et al., 2001]. Normally, the objective of the networked control system is to make use of the finite network capacity to achieve system stability and good closed loop performance, including stability, rise time, overshoot and other design criteria [Beldiman, 2001, Ramaswamy et al., 2008]. Two approaches are commonly used in the design of networked control systems. One approach is to make modifications to the network protocol for a given plant and controller (in this case, the controller is designed in advance without considering network-induced effects). The other method, followed in this paper, is to consider network-induced effects and set the design criteria as well as network protocol definition in the controller design for a given plant. Random delays in the network are a serious problem, which deteriorate the performance and may even cause instability of the system. A controller which could ensure the stability of system at large values of delay will revolutionize internet based controllers. Such a jitter-robust controller based on fractional order proportional and integral (FOPI) tuning rules, proposed in [Bhambhani et al., 2008], is used for speed control of stand-alone Smart Wheel platform at CSOIS. For this purpose, the real random delay in the network is taken into account by pinging different locations and then considering it as the delay in our simulation and experimental systems.

Simulation results are presented to illustrate the effectiveness of the proposed OFOPI controller by a comparison between the fractional order controller and the integer order PID controller [Eriksson and Johansson, 2007a].

This paper is organized as follows. Section 2 describes the “Smart Wheel” system and briefly explains its system identification using step response analysis. Section 3 introduces the OFOPI controller design and tuning rules. Section 4 describes the laboratory setup of the networked control system for speed control of the steering axis of the Smart Wheel using hardware-in-the-loop implementation. Section 5 shows the results for the simulated networked control of the Smart Wheel using OFOPI controller at random time delays. This is compared to optimal PID [Eriksson and Johansson, 2007a,b] controller. Section 6 presents the real-time hardware-in-the-loop control. Finally Section 7 summarizes the results and provides concluding remarks.

2. ARCHITECTURE OF THE CSOIS SMART WHEEL

The Smart Wheel is a self-contained robotic wheel which has a steering axis, drive axis and z-axis, each of which can be actuated independently [Flann and Moore, 2000]. The CSOIS has a stand-alone Smart Wheel assembly (shown in Fig. 1). It is equipped with steering and drive motors. It also has a linear actuator for z-axis movement. The Smart Wheel also has a power distribution unit, drive circuitry for the motors and actuators, encoders for drive and steering feedback and a microcontroller.

Furthermore, the Smart Wheel’s microcontroller is connected to a serial server. Hence anyone with a computer connected to the network can control the Smart Wheel. The microcontroller on the wheel polls the position encoder on the Smart Wheel. The data read is transmitted over the network through its serial
port. It accepts velocity values through the serial port and
converts them to pulse width modulated (PWM) signals. These
signals drive the motor. An operator with a computer and access
to the internet can install a virtual COM driver and can gain
access to the Smart Wheel. Algorithms can be used to read the
data from the Smart Wheel and to calculate the wheel’s velocity
based on the encoders position data. A controller can thus be
designed on a remote computer to send velocity commands to
the Smart Wheel.

An internet camera (DLink - model DCS5300) [DLink Systems
Inc., 2003] located near the Smart Wheel assembly sends
streaming audio and video of the wheel’s motion to the remote
computer. The audio and video streams are independent of the
controller data which uses the serial server. The encoder data
can be plotted on the screen of the remote computer to analyze
the performance of the closed loop system.

2.1 System Identification of the Smart Wheel

The first step is the system identification of the stand-alone
Smart Wheel assembly at the CSOIS. The system identification
is also done through a hardware-in-the-loop simulation. For
this code was written in MATLAB to enable communication
between the remote computer and the Smart Wheel. The first
step is to connect to the port on the Smart Wheel and obtain
access to it. This is done by sending the REMOTE_REQ or
ASCII character 47 to the Smart Wheel. The Smart Wheel
responds with a REMOTE_ACCEPT or ASCII character 33 in
its buffer. Connection is verified if this character is received.
When connected, data is read in from the buffer on the Smart
Wheel. Size of the data available is limited by the size of the
buffer. Longer data produces greater computational delay and
noise when differentiated to get velocity, hence a proper data
length selection is very important. The Smart Wheel reports
data in a particular format, hence it is parsed to get the value
of steering angle ($\theta$) and time ($t$) from it.

Also an s-function is written which implements the same MAT-
LAB code for communication. The s-function block properly
sizes the $\theta$ and $t$ vectors so that they are of the same length
and are synchronized. Figure 2 shows the steering speed of
Smart Wheel corresponding to a control input of 19. X-axis is
the time in seconds and Y-axis is the steering speed of Smart
Wheel obtained by taking derivative of $\theta$ with respect to $t$ in
radians/seconds.

Furthermore, it should be noted that the input issued to the
motor over the network is in the range of (0 to $\pm 19$), with the
signs representing the direction of rotation. Also, since the input
is not 1 or 5 volts, as is the case in a traditional step response
system ID, the value of $K$ obtained for the motor is low. A hard-
ware in loop implementation of the Smart Wheel is done using
an s-function block which handles communication between the
plant and the remote computer. The sampling interval for the
s-function block in Simulink is set equal to the time needed
to read data from the buffer to attain synchronization. A system
identification is performed by obtaining the step response of the
Smart Wheel. From the step response, the transfer function for
the FOPDT model of the Smart Wheel is obtained as

$$P(s) = \frac{K}{Ts + 1} e^{-\frac{Ls}{s}} = \frac{0.1484}{0.045s + 1} e^{-0.592s}$$

(1)

3. OFOPI DESIGN METHODS & PRACTICAL TUNING
RULES

The past decade has seen an immense amount of research work
on fractional order controller design and tuning methods [Old-
ham and Spanier, 1974, Vinagre and Chen, 2002, Chen et al.,
1977, Lubich, 1986, Podlubny and Misanek, 1993, Podlubny,
1994a,b]. This section provides a brief summary of the design
method and tuning strategy developed in [Bhambhani et al.,
2008]. The motivation for the research was from ideas devel-
oped in [Eriksson and Johansson, 2007a,b, Bhaskaran et al.,
2007b,a] where tuning rules for PI/PID controllers where de-
developed for a class of systems which can be approximated
with a good FOPDT model. The objective was to design an
optimal controller such that the jitter margin and system per-
formance are maximized and yet the closed loop feedback sys-
tem is robust and stable. For this a multi-objective optimiza-
tion method was used which simultaneously minimizes two
objective functions namely the ITAE factor and jitter margin
which are functions of controller gain parameters $x$ bounded by
some non-linear equality and inequality constraints. Expressed mathematically, the two objective functions targeted are:

\[ O_1(x) = \int_0^\infty t[e(t)]dt \]  

and  

\[ O_2(x) = \frac{1}{\delta_{\text{max}}} \]

Here \( \delta_{\text{max}} \) can be computed from (3) as:

\[ \delta_{\text{max}} = \min_{\omega \in [0, \infty]} |1 + G(j\omega)C(j\omega)| \omega \frac{\partial C(j\omega)}{\partial \omega} \]  

(3)

Hence the multi-objective optimization problem takes the form as in (4) and (5). Minimize

\[ O(x) = [O_1, O_2] \]  

at

\[ x = [K_p, K_i, \alpha] \]

such that \( x \) satisfies the equality and inequality constraints given by

\[ \sigma = \left\{ \begin{array}{ll} |D + G(i\omega)C(i\omega)|^2 & \geq R^2 \quad i = 1, \ldots, a1 \\ \partial D + G(i\omega)C(i\omega) & = 0 \quad i = 1, \ldots, a2 \end{array} \right. \]

(5)

Here objective function \( O_1(x) \) is the ITAE criterion and \( O_2(x) \) is the inverse of jitter margin. These values should be minimized while still ensuring robustness of the system. The set of equations defined by \( \sigma \) ensures robustness and stability. The inequality constraint \( |D + G(i\omega)C(i\omega)|^2 \) is the sensitivity constraint and is a function of \( K_p, K_i, \alpha \) and \( \omega \) and must be greater than \( R^2 \). Here \( D \) and \( R \) are the center and radius of the circle which encloses both the \( M_s \) and \( M_p \) circles and are given by (6) as

\[ D = \frac{M_s - M_p M_p - 2 M_s M_p^2 + M_p^2 - 1}{2 M_s (M_p^2 - 1)} \]

\[ R = \frac{M_s + M_p - 1}{2 M_s (M_p^2 - 1)} \]

\( M_s \) and \( M_p \) are the maximum absolute values of sensitivity and complementary sensitivity functions respectively. Furthermore, \( |1 + G(i\omega)C(i\omega)|^2 = 0 \) is the stability region of the sensitivity constraint and satisfies the boundary condition at critical point or the point at which \( D = 1 \) and \( R = 0 \). For more information read [Bhaskaran et al., 2007b,a]

The OFOPI tuning rules obtained by the above procedure as derived in [Bhambhani et al., 2008] are stated as:

\[ K_p = \frac{0.2T}{L} + 0.16 \]

\[ K_i = \frac{0.25}{TT} + \frac{0.1983}{L} \]

\[ \alpha = \frac{\tau - 0.04L + 1.2399}{L} \]

where \( \tau = \frac{L}{\alpha} \). Whereas the PID tuning rules for varying time delay systems [Eriksson and Johansson, 2007a,b] are listed as:

\[ k_i = \frac{0.4T - 0.04}{K_p L} + 0.16 \]

\[ k_d = 0.01 \left( -0.17T^3 + 1.5T^2 - 1.5 \right) \]

\[ k_d = 0.01 \left( \frac{0.4T^2 + 11T}{K_p} \right) \]

It should be noted that the OFOPI tuning rules derived in [Bhambhani et al., 2008] were based on a set of FOPDT systems such that their delay values are \( L = [1, 2, 3, \ldots, 9, 10]^T \), values of the time constant are \( T = [1, 2, 3, \ldots, 9, 10]^T \) and steady state gain \( K = 1 \), whereas the Smart Wheel lies out of the range. Efforts were made to simulate another set of FOPDT systems with time delay and time constant values in the range \( L = [0.1, \ldots, 1]^T \) and \( T = [0.1, \ldots, 1]^T \). However due to computational limitations and frequency mismatch, this still remains a difficult task and is one of the major challenges. Similar difficulties were encountered for a PID controller using the approach found in [Eriksson and Johansson, 2007a] for the new set of FOPDT systems, although the range of plants considered in that paper were \( L = [0.1, 1, \ldots, 10]^T, T = [0.1, 1, \ldots, 10]^T \) and \( K_p = 1 \). To see how well the existing OPID/OFIPI tuning rules perform for the Smart Wheel system at different random delays, real-time experiments were conducted. It was noticed that the tuning rules still hold true i.e. the networked system is still stable for delays less than jitter margin. However the upper bound of jitter margin increases by 60% to 70%. This was concluded from simulating several systems lying in the new range.

\[ \delta_{\text{max}}' = 1.6 \times \delta_{\text{max}} \]

(9)

where, \( \delta_{\text{max}} \) is the jitter margin for new set of FOPDT systems at the same OFOPI/OPID tuning tuning parameters.

4. LABORATORY SET UP AND HARDWARE-IN-THE-LOOP

This section gives a brief introduction about the implementation of OFOPI control of the steering axis speed of the Smart Wheel over the network loop. Figure 3 shows a simple implementation of a OFOPI controller with the Smart Wheel in the loop. The block labeled ‘smartwheel’ is the actual block which handles the entire process of communicating remotely with the hardware. Operational details of that block are illustrated with help of a flow chart shown in Fig. 4. As inferred from the flow chart, the first step is to connect the Smart Wheel’s port and obtain access to it. The connection process is done at the start of the simulation. A count is maintained and connection is performed only on the first ever operation. In the event of timeout or no receipt of connection code, the simulation shuts down. When connected, data is read in from the buffer on the Smart Wheel.

Size of the data available is limited by the size of the buffer, longer data produces greater computational delay and noise when differentiated to get velocity, hence a proper data length selection is very important. The Smart Wheel reports data in ASCII format, whereas the Smart Wheel lies out of the range. Efforts were made to simulate another set of FOPDT systems with time delay and time constant values in the range \( L = [0.1, \ldots, 1]^T \) and \( T = [0.1, \ldots, 1]^T \). However due to computational limitations and frequency mismatch, this still remains a difficult task and is one of the major challenges. Similar difficulties were encountered for a PID controller using the approach found in [Eriksson and Johansson, 2007a] for the new set of FOPDT systems, although the range of plants considered in that paper were \( L = [0.1, 1, \ldots, 10]^T, T = [0.1, 1, \ldots, 10]^T \) and \( K_p = 1 \). To see how well the existing OPID/OFIPI tuning rules perform for the Smart Wheel system at different random delays, real-time experiments were conducted. It was noticed that the tuning rules still hold true i.e. the networked system is still stable for delays less than jitter margin. However the upper bound of jitter margin increases by 60% to 70%. This was concluded from simulating several systems lying in the new range.

\[ \delta_{\text{max}}' = 1.6 \times \delta_{\text{max}} \]

(9)

where, \( \delta_{\text{max}} \) is the jitter margin for new set of FOPDT systems at the same OFOPI/OPID tuning tuning parameters.
Fig. 4. Flowchart describing working of Smart Wheel

ASCII values 70 to 51 (70 - stopped, 51 - maximum speed).
The control input lies in the range of \([0, 20]\) for either direction of rotation.

As seen in Fig. 3, a saturation block is placed in the loop which modulates the controller output to acceptable values. This necessitates the use of an anti-windup feedback block to the integrator to prevent integrator saturation. The correct values are fed into the s-function block which communicates with the Smart Wheel and sends appropriate commands. In each cycle, the s-function block reads data, parses data, calculates error, takes input from the controller, sends commands to the Smart Wheel and plots current received data to the screen.

### 5. SIMULATION ILLUSTRATION

The next step is to compute the controller gain parameters for the two controllers, i.e. OFOPI and OPID based on equations (7) and (8). Two different sets of random time delays are obtained by pinging two different places located far apart from the Smart Wheel platform. These are HuaZhong University of Science and Technology in China and the Australian National University. The gain parameters so obtained for different controllers are listed in Tables 1 and 2, where AMIGO / FMIGO stand for Approximate / Fractional \(M_s\) Constrained Integral Gain Optimization.

It should be noted that the jitter margin values for OPID are larger as compared to OFOPI controller, which is true as the Smart Wheel is a delay dominated system. Based on the above parameters, OFOPI and OPID controllers are used to control the steering speed of the Smart Wheel for different delays. Figure 5 and Fig. 6 show the simulated result for random delays obtained by pinging the university in China whereas Fig. 7 and Fig. 8 are for random delays obtained by pinging the Australian university.

#### Table 1. Gain parameters for different PID controllers

<table>
<thead>
<tr>
<th></th>
<th>PID</th>
<th>(K_p)</th>
<th>(K_i)</th>
<th>(K_d)</th>
<th>(JM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aus PID</td>
<td>AMIGO</td>
<td>1.524</td>
<td>0.43189</td>
<td>0.009323</td>
<td>1.8696</td>
</tr>
<tr>
<td></td>
<td>OPID</td>
<td>0.8866</td>
<td>4.204</td>
<td>0.05541</td>
<td>1.5596</td>
</tr>
<tr>
<td>China PID</td>
<td>AMIGO</td>
<td>1.3013</td>
<td>0.38106</td>
<td>0.009323</td>
<td>2.1428</td>
</tr>
<tr>
<td></td>
<td>OPID</td>
<td>0.91105</td>
<td>3.684</td>
<td>0.05541</td>
<td>1.7755</td>
</tr>
</tbody>
</table>

#### Table 2. Gain parameters for different FOPI controllers

<table>
<thead>
<tr>
<th></th>
<th>FOPI</th>
<th>(K_p)</th>
<th>(K_i)</th>
<th>(\alpha)</th>
<th>(JM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aus FOPI</td>
<td>FMIGO</td>
<td>2.1227</td>
<td>4.5671</td>
<td>1.1</td>
<td>0.2168</td>
</tr>
<tr>
<td></td>
<td>OFOPI</td>
<td>0.1716</td>
<td>7.524</td>
<td>2.154</td>
<td>0.6384</td>
</tr>
<tr>
<td>China FOPI</td>
<td>FMIGO</td>
<td>2.1079</td>
<td>3.9876</td>
<td>1.1</td>
<td>0.2186</td>
</tr>
<tr>
<td></td>
<td>OFOPI</td>
<td>0.17015</td>
<td>6.5769</td>
<td>2.1561</td>
<td>0.6394</td>
</tr>
</tbody>
</table>

Here \(L(t)\) denotes the random delay and includes both the network delays and inherent delays due to computation and other miscellaneous factors, \(JM\) represents \(\delta_{max}\) and \(JM''\) represent \(\delta''_{max}\). It can be said that the OFOPI controller has a faster response as compared to OPID controller. Also the simulation results are as expected.

#### 6. EXPERIMENT ON THE REAL-TIME SMART WHEEL SPEED CONTROL SYSTEM

The final step is the experiment. Figure 9 and Fig. 10 show the real-time results for random delays obtained by pinging the university in China whereas Fig. 11 and Fig. 12 are for random delays obtained by pinging the Australian university. Noise due to sensors and actuators are an inherent part of real-time network control unlike simulation results. Furthermore, oscillations are observed due to delays and physical phenomenon like suspension movements, frictional changes and motor-cogging. Also the motor velocity has a lower bound due to deadzone.

As can be seen in above figures, the OFOPI controller has a faster response as compared to the PID controller. Also for
Simulated network control of steering speed of Smart Wheel using OPID tuning rules: China

Fig. 6. Simulated Network control of steering speed of Smart Wheel using OPID: China

Simulated network control of steering speed of Smart Wheel using OFOPI tuning rules: Australia

Fig. 7. Simulated Network control of steering speed of Smart Wheel using OFOPI: Australia

every case considered, the system becomes more and more unstable as the delay is increased and finally reaches instability at delay greater than $\delta_{\text{max}}^{''}$. Moreover, the results obtained in real-time are in confirmation with simulation results obtained in Section 5.

7. CONCLUSION

This paper presents an intensive study and experimental work on networked control of the steering speed of the stand-alone Smart Wheel platform at CSOIS, Utah State University. Optimal fractional order proportional and integral tuning rules are used to determine the parameters at which the system has maximum delay and yet is stable. An s-function block is used in Simulink to implement the Smart Wheel in the network loop. It is noticed that since the Smart Wheel is a delay-dominated system, it will result in higher jitter margin using an OPID controller. This is in confirmation with the results obtained in the work of [Bhambhani et al., 2008]. However, it is seen that for systems with $L$ and $T$ in the range $[0.1, \cdots, 1]$, the upper bound on jitter margin is increased manyfold. Also the OFOPI controller has a faster performance as compared to OPID controller. Hence, the work presented in this paper justifies that the OFOPI tuning rules work well in the networked control setting having random delays.

Real-time network control of steering speed of Smart Wheel using OFOPI tuning rules: China

Fig. 9. Real-time Network control of steering speed of Smart Wheel using OFOPI: China

Simulated network control of steering speed of Smart Wheel using OFOPI tuning rules: Australia

Fig. 8. Simulated Network control of steering speed of Smart Wheel using OFOPI: Australia
Fig. 10. Real-time Network control of steering speed of Smart Wheel using OPID: China

Fig. 11. Real-time Network control of steering speed of Smart Wheel using OFOPI: Australia

Fig. 12. Real-time Network control of steering speed of Smart Wheel using OPID: Australia

REFERENCES


ACKNOWLEDGEMENT

Ying Luo is supported by the Ministry of Education of the P. R. China and China Scholarship Council (CSC). The authors acknowledge the benefits from the weekly Fractional Calculus Reading Group meeting at CSOIS (http://mechatronics.ece.usu.edu/foc/yan.11/).


T. Tatanské Matliare.


