The Development of Proportional-Integral-Plus Control Using Field Programmable Gate Array Technology Applied to Mechatronics System

Ayman A. Nada\textsuperscript{a}, E. M. Shaban\textsuperscript{b}

\textsuperscript{a}Benha Institute of Technology, Benha University, Benha 13512, Egypt, email: arobust@tedata.net.eg
\textsuperscript{b}Mechanical Engineering Department, Jazan University, Jazan – KSA, On leave from Faculty of Engineering (Mataria), Helwan University – Egypt, email: modern3@hotmail.com

Abstract

Proportional-Integral-Plus (PIP) controller can be considered as an extension of classical discrete PI controller, in which, the proportional and the integral actions are enhanced by higher order of input and feedback compensators (PLUS terms). Conventionally, the PIP control design is carried out entirely in discrete time domain, for which open-loop data-based identification and estimation for the linearized transfer function model is performed off-line. This is followed by an on-line implementation for the control operations. In case of on-line estimation of the transfer function model, the overall execution time, at each step, increases considerably. Here, the overall execution time is the sum of each operation, i.e. estimation and control implementation. In such cases, digital processors with single resources may fail to capture relatively fast mechatronic systems properly, causing a decrease in the controller performance. This paper investigates the use of parallel processing facilities available within the Field Programmable Gate Arrays (FPGAs) technology and reconfigurable Input/output chips, PIP-FPGA, for which a fixed-point PIP control design algorithm is developed and implemented upon relatively fast Mechatronic system with high-speed control and high channel count on an FPGA target. Here, parallel loops for measuring feedback signals, model parameter estimation, updating of the control gains, and generating pulse train for actuator's drivers, are illustrated through experimental test-rig of simple automation line system. The experimental results shows valuable enhancement of the implemented PIP control upon such mechatronic systems.

Keywords: Proportional-Integral-Plus (PIP) control, system identification and estimation, Field Programmable Gate Array (FPGA)


1. Introduction

Recently, the control design using State Variable Feedback (SVF) becomes a typical approach for many control applications [1, 2]. It is based on state space formulation of the system model, which allows the implementation of powerful control approaches, such as pole placement, Linear Quadratic Regulator (LQR), Linear Quadratic Gaussian (LQG) and other optimal approaches, e.g. $H_2$ and $H_{\infty}$ control [3, 4].
One of the major difficulties in the state space approach is that the state vector is not normally available for direct measurement. Therefore, an estimate of the states is required for the implementation of the control system. However, this estimation process, in turns, decreases the control system robustness [4]. The problem remains unresolved in discrete control theories based on minimal state space formulation [5].

The Proportional-Integral-Plus (PIP) control design approach, which is firstly introduced by Young and Wang [6], present a simple solution to this problem by defining the Non-Minimal State Space (NMSS) formulation. Here, the state vector is composed only of those states that can be measured directly and stored for the use of control law. These states are the present and past sampled values of the output variable, and the past sampled value of the input variable. The PIP control method has tuning parameters represented by the weights of the LQR cost function. In this regard, adequate closed-loop performance is obtained by direct tuning of these parameters [7, 8]. NMSS-PIP controllers have been successfully designed for a wide range of real applications [10-23]. Preliminarily, PIP control is based on the discrete time transfer functions (TF), which normally identified and estimated from several open-loop system experiments, by means of Simplified Refined Instrumental Variable (SRIV) which has the ability to smooth the noise subjected to data collection [24-27].

Mechatronic systems are defined as a computer-controlled mechanical system, including both digital computer and electromechanical components. Some examples of fast mechatronic systems include: a DVD player, computer hard disc drive, intelligent mechanisms, mobile robots, packing machines, and smart materials applications. These applications are characterized by its relatively low inertia and quick response time, beside the reference tracking requirements in some situations [28, 29].

The aim of this paper is to study the effectiveness of implementing the PIP control system design upon such fast mechatronic systems. The practical applications in references [9-15] have large capacitances and therefore their responses are relatively slow, which, thereby require low sampling rates. Another concern is the use of single processor computers in order to implement the control action on some of these applications using LabVIEW real-time environment [14, 23, 30] and MATLAB/Simulink [31]. Here, the overall execution time for multiple control process operations is the sum of the execution times of each operation, i.e. estimation and control implementation. Jun Gu, et. al. [16] had used three embedded PC104 computers, each for separate control task in order to decrease the execution time.

In this work, the Field Programmable Gate Arrays (FPGAs) technology [32, 33] is proposed to execute simultaneous parallel control operations. It is worth to note here that FPGA is No-Processor device and therefore can be used effectively for control processes demanding very high loop cycle time. Consequently, this enables FPGAs to score over general purpose computing chips like, such as DSP chips. This reduction in the execution time expects the increase of the efficiency of NMSS-PIP control for such relatively fast mechatronic systems. The ready-made LabVIEW-FPGA module [34, 35] can be used to generate the code and compiling procedures to develop a standalone on-shelf PIP controller for such mechatronic systems.

2. PIP Control

In terms of backward shift operator, $z^{-1}$, the dynamic representation of discrete time TF model of single-input single-output (SISO) system takes the form
The non-minimal state vector $x(k)$ is defined in terms of the present and past sampled outputs and the past sampled inputs, plus the discrete-time integral of error between the output $y(k)$ and the command input $y_d(k)$. It takes the following form

$$x(k) = \begin{bmatrix} y(k) & \ldots & y(k-n+1) & u(k-1) & \ldots & u(k-m+1) & z(k) \end{bmatrix}^T$$

where integral-of-error state variable, $z(k)$, is represented as

$$z(k) = z(k-1) + (y_d(k) - y(k))$$

The NMSS formulation can be written as [7]

$$x(k) = Fx(k-1) + gu(k-1) + k_d y_d(k)$$

$$y(k) = hx(k)$$

where $F$ is state transition matrix, $g$ is input vector, $k_d$ is the command input vector, and $h$ is output vector. Since the SVF control law in the NMSS case involves only the measured input and output variables and their past values, it avoids the need for an explicit state estimation [7]. The SVF control law associated with the NMSS model takes the form of

$$u(k) = -K^T x(k)$$

for which the SVF control gain vector can be represented as

$$K^T = \begin{bmatrix} f_o & \ldots & f_{o-1} & g_1 & \ldots & g_{m-1} & -k_I \end{bmatrix}$$

Regarding the control law, equation (5), the block diagram of the closed loop control system, can be constructed as shown in figure (1). The figure shows that the PIP controller can be considered as an extension of the classical discrete PI controller. The proportional action $f_o$ and the integral action $k_I(1-z^{-1})^{-1}$ are enhanced by higher order input $G_i(z^{-1})$ and the feedback $F_i(z^{-1})$ compensators. These filters can be represented as:

$$G_i(z^{-1}) = g_1 z^{-1} + \ldots + g_{m-1} z^{-m+1}$$

$$F_i(z^{-1}) = f_1 z^{-1} + \ldots + g_{n-1} z^{-n+1}$$

One possible approach to calculate the control gain vector, $K$, is the optimal state variable feedback control which minimizes the Linear Quadratic (LQ) type of performance criterion, i.e.

$$J = \sum_{k=0}^\infty [x^T(k)Qx(k) + ru^2(k)]$$

Nada, et al., 2014: Vol 2(4)

ajrc.journal@gmail.com
This represents the standard formulation of the infinite time optimal LQ servomechanism cost function for SISO system. The weighing square symmetric positive matrix of states, $Q$, has the dimension $n + m$, and $r$ is a positive scalar for weighing the control action. Considering the NMSS state vector, equation (2), the weighing state matrix takes $Q = diag[q_y \quad q_u \quad q_e]$, where $q_y = [q_1 \ldots q_n]$, and $q_u = [q_{n+1} \ldots q_{n+m-1}]$ are the weighting measured output and input respectively, whereas $q_e$ is the weighting constraint on the integral-of-error state variable $z(k)$.

Given the NMSS system description $\{F, G\}$, weighing matrix $Q$ and the control input weighing $r$; the SVF gain vector, $K^T$, is given by standard LQ theory as

$$K^T = (r + g^T P g)^{-1} g^T P F$$

(9)

where $P$ is the steady-state solution of the following *discrete-time Riccati equation*

$$P - F^T PF + F^T P g (r + g^T P g)^{-1} g^T P F - Q = 0$$

(10)

Because of the singularity of the state transition matrix $F$, the SVF gains are obtained recursively [38] as follows

$$K^T (i) = (r + g^T P (i + 1) g)^{-1} g^T P (i + 1) F$$

(11)

$$P (i) = Q + F^T P (i + 1) F - F^T P (i + 1) g K^T (i)$$

Here, the initial value of matrix $P$ takes the same value of $Q$, i.e. $P (N) = Q$ and $K (N) = 0$.

3. **Field Programmable Gate Arrays (FPGA)**

An FPGA is analogous to a printed circuit board that has a number of unconnected devices on it. The connection in an FPGA circuit can be dynamically defined by appropriate software programming. The program causes semiconductor switches to turn ON or OFF, thereby defining the connections between gates. The FPGA is defined as a programmable chip composed of three basic components: logic blocks, programmable interconnects, and Input/Output (I/O) blocks. A single FPGA chip can replace thousands of discrete components by incorporating millions of logic gates. Figure (2) shows an FPGA as a reconfigurable digital architecture with a matrix of configurable logic blocks with horizontal and vertical routing channels surrounded by a periphery of IO blocks. Signals can be routed within the FPGA matrix in any arbitrary manner by programmable interconnect switches and wire routes.
However, programming these FPGAs require experienced digital designers and hardware engineers. NI\textsuperscript{1} has simplified programming of these devices via visual programming system design with LabVIEW-FPGA module, hence the advantages of these powerful reconfigurable chips can be explicitly implemented.

The LabVIEW-FPGA module is used to define the FPGA logic, similar to some low-level languages, such as very high speed integrated circuit hardware description language (VHDL). Here, LabVIEW-FPGA generates the VHDL code and passes it to the Xilinx\textsuperscript{2} compiler. Then the Xilinx compiler synthesizes the VHDL and routes all synthesized components into a “bitfile”. The compiled “bitfile” is downloaded to the FPGA chip. FPGA logic provides timing, triggering, processing and custom IO measurements. One can implement multi-loop analog control systems at loop rates exceeding 100 KS/s\textsuperscript{1}, and digital control systems at loop rates up to 1 MS/s, and it is possible to evaluate multiple rungs of Boolean logic using single-cycle “while” loops at 40 MHz.

![FPGA Diagram](image)

**Figure 2** Reconfigurable FPGA chip showing the ability of parallel processing.

LabVIEW and FPGA, together, can implement synchronous or asynchronous parallel tasks in hardware to process and generate synchronized analog and digital signals rapidly and deterministically. FPGA module creates dedicated hardware for each independent function in FPGA VIs. Besides, there is no operating system on the FPGA module that divide the Central Processing Unit (CPU) time between several tasks. This parallel nature of the Reconfigurable Input/Output (RIO) core provides additional computation without reduction in the speed of the FPGA application.

\begin{itemize}
\item[\textsuperscript{1}] National Instruments Corporation, www.ni.com
\item[\textsuperscript{2}] www.xilinx.com
\end{itemize}
In order to attain the parallel processing, the code should be divided into several independent segments “loops”. For example, in mechatronic demonstrator, independent loops can be created for acquiring the measured analog data with distinct loop rates.

The FPGA module restricts the use of mathematical operations in FPGA VIs to integer numeric data types and it is not possible to use the floating point operations. Also, even as integer math is used, the results may overflow, when the result of a mathematical operation exceeds the range of the output data type. One possible approach to avoid the integer overflow is scaling down the magnitude of inputs by power of 2, or the use of larger output data type. Nevertheless, the scaling minimize the amount of space used in FPGA device, it reduces the precision of the mathematical operations. On the other hand, larger output data type take more space on FPGA device, yet perform the processing more quickly and receive more accurate data. In this investigation, scaling down of control action has been used with acceptable control performance. Furthermore, only fixed-size, one dimensional arrays in FPGA VIs can be used, i.e., the arithmetic manipulation that returns a variable-size array cannot be used. Since arrays consume significant amounts of space on FPGA, therefore, arrays larger than 32 elements should be avoided.

In the current experimental work, single board RIO (sbRIO), see figure (3), is used. Here sbRIO-9631 integrates a 266MHz real-time processor with 1M gate Xilinx Spartan FPGA, 110 digital IO lines. It provides 64MB of DRAM for embedded operation and 128MB of nonvolatile memory for storing programs and data logging. This device features a built-in 10/100 Mbits/s Ethernet port which enables to conduct signals communication over the network.

A simple Mechatronic demonstrator is constructed using the following main components: actuator (DC motor), sensor (Encoder), driver (linear/pulses) and belt conveyer as a simple automation line for
Controller evaluation, see figure (4). For the sake of controlling the mechatronic system, FPGA layout should include two basic independent loops; the first loop concerns measuring and feedback the angular velocity by the optical quadrature encoder, whereas the second is responsible for generating the necessary pulse width modulation (PWM) to drive the DC motor. These two loops are also used, preliminarily, for TF parameters estimation process. During the implementation process, some other loops are incorporated with these two basic loops, such as control action loop, and another loop data collection and transmission to the real-time processor for TF parameters updating process. Figure (5) shows the layout of the parallel processing loops. The details of measuring and driving loops are found in [37].

4. Development of FPGA-PIP Controller

In order to develop the procedure of fixed-point PIP control for FPGA reconfigurable chip, the limitation of the available space in the FPGA chip as well as the numerical manipulation using only fixed-size integer numbers should be considered. Therefore, in this work, the identification of the TF model and preliminarily estimation of its parameters are carried out offline. This is followed by the development of fixed-point PIP controller.

4.1. System identification and estimation

Consider the following revised TF model in which the sample delay $\delta$ is explicitly acknowledged,

$$y(k) = \frac{b_\delta z^{-1} + b_{\delta+1} z^{-(\delta+1)} + \ldots + b_m z^{-m}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}} u(k)$$

(12)

Recalling that $n$ is the number of output parameters, and $m - \delta + 1$ is the number of input parameters. Equation (12) can be represented incrementally as
\[ y(k) = -a_1 y(k-1) - \ldots - a_n y(k-n) + b_{\delta} u(k-\delta) + b_{\delta+1} u(k-(\delta+1)) + \ldots + b_n u(k-m) \]  \hspace{1cm} (13)

for which the TF parameters, \( a_i \ (i = 1,2,\ldots,n) \) and \( b_j \ (j = \delta,\delta+1,\ldots,m) \) are explicitly appeared.

Equation (13) can be written in following form

\[ y(k) = \Phi^T(k) \xi \]  \hspace{1cm} (14)

given that:

\[ \Phi(k) = \begin{bmatrix} -y(k-1) & \ldots & -y(k-n) & u(k-\delta) & u(k-(\delta+1)) & \ldots & u(k-(\delta+m)) \end{bmatrix}^T \]

\[ \xi = \begin{bmatrix} a_1 & \ldots & a_n & b_\delta & b_{\delta+1} & \ldots & b_{\delta+m} \end{bmatrix}^T \]
Linear Least Squares (LLS) can be used to estimate \( \hat{\xi} \), based on the data samples collected; by the minimization of the error between the measured output \( y(k) \) and the estimated model response \( \Phi^T(k)\hat{\xi} \). Therefore, the cost function, \( J(\xi) \), takes the following form:

\[
J(\xi) = \sum_k \left[ y(k) - \Phi^T(k)\hat{\xi} \right]^2
\]  

(15)

The concept of solving, equation (15) for \( \hat{\xi} \) is based on the partial differentiation of the cost function \( J(\xi) \) with respect to each element in the vector \( \xi \), i.e., \( \partial J(\xi) / \partial \xi = 0 \). The resulting \( n + m \) system of equations is solved simultaneously to estimate the vector of TF parameters \( \hat{\xi} \) as follows:

\[
\hat{\xi} = \left[ \sum_k \Phi(k)\Phi^T(k) \right]^{-1} \left[ \sum_k \Phi(k)y(k) \right]
\]

(16)

Equation (16) can be used for preliminary estimate of the system parameters, \( \hat{\xi} \), in an offline manner, provided that the matrix, \( \Gamma \) exists. In order to update the system parameters during the real-time operation; Recursive Least Squares (RLS) algorithm is used instead. At a given sample \( k \), the matrix \( \Gamma^{-1}(k) \) and the vector \( \beta(k) \) can be estimated recursively as follows:

\[
\Gamma^{-1}(k) = \Gamma^{-1}(k-1) + \Phi(k)\Phi^T(k)
\]

(17)

\[
\beta(k) = \beta(k-1) + \Phi(k)y(k)
\]

Using the matrix inversion lemma, which provides the inverse of the accumulated `cross-product' matrix \( \Gamma(k) \),

\[
\Gamma(k) = \Gamma(k-1) - \Gamma(k-1)\Phi(k)\left[1 + \Phi^T(k)\Gamma(k-1)\Phi(k)\right]^{-1}\Phi^T(k)\Gamma(k-1)
\]

(18)

Regarding equation (16), the updated TF parameters vector at instant \( k \) is \( \hat{\xi}(k) = \Gamma(k)\beta(k) \), which yields

\[
\hat{\xi}(k) = \hat{\xi}(k-1) + \Gamma(k-1)\Phi(k)\left[1 + \Phi^T(k)\Gamma(k-1)\Phi(k)\right]^{-1}\left[y(k) - \Phi^T(k)\hat{\xi}(k-1)\right]
\]

(19)

Equations (18 and 19) summarize the RLS algorithm used to update the TF parameters in the LabVIEW VI.

Primarily, the collected data of the normalized input, scaled from \(-1000\) to \(1000\), applied to the DC motor driver, and the corresponding measured angular velocity of the motor's shaft shows that the best form of the TF that best describe the dynamic behavior of the mechatronic system is

\[
\frac{y(k)}{u(k)} = \frac{b_2z^{-4}}{1 + a_1z^{-1}}
\]

(20)
As shown in equation (20), the system is first order, with 4-samples delay, for which \( y(k) \) is the angular velocity of the DC motor in rad/s. Also, TF parameters values \( a_4 = -0.8993 \) and \( b_4 = 0.0024 \) return a coefficient of determination \( R^2 = 0.96 \).

4.2 PIP controller design and implementation

Based on the identified TF, equation (20), the NMSS formulation takes the form of equation (4), for which the state vector \( x(k) \), the state transition matrix \( F \), and the associated input vector \( g \) are presented as

\[
x(k) = \begin{bmatrix} y(k) & u(k-1) & u(k-2) & u(k-3) & z(k) \end{bmatrix}^T
\]

\[
F = \begin{bmatrix}
-a_4 & 0 & 0 & b_4 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
-a_1 & 0 & 0 & -b_4 & 1
\end{bmatrix}
\]

\[
g = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T
\]

(21)

According to equation (6), the SVF gain vector has 5 elements and takes the following form

\[
K = \begin{bmatrix} f_o & g_1 & g_2 & g_3 & -k_f \end{bmatrix}^T
\]

(22)

The incremental form of the control law can then be written as

\[
u(k) = u(k-1) - f_o (y(k) - y(k-1)) - g_1 (u(k-1) - u(k-2)) - g_2 (u(k-2) - u(k-3)) - g_3 (u(k-3) - u(k-4)) + k_f (y_d(k) - y(k))
\]

(23)

The control law depicted in equation (23) should be constructed upon the FPGA chip, taking into consideration the space occupied by mapping the variable into fixed-point numbers before converting it into integer data types within LabVIEW environment, such that the control action sub-VIs avoids values overflow. This PIP control action is then fed-forward to the PWM motor driver via sbRIO-9631 for implementation process. The compilation process is carried out, using LabVIEW-FPGA compiler.

Three values for the weightings \( r \) and \( Q \) have been tried out, the corresponding control gain vectors, \( K \), are evaluated using equation (11) and Table (1) summarizes their numerical values. The simulation of the three controllers give acceptable closed loop responses with little to choose between them, see Figure (6), however the third controller is chosen since it gives relatively fast closed loop response.

<table>
<thead>
<tr>
<th>#</th>
<th>( r )</th>
<th>( Q )</th>
<th>( K^T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.25</td>
<td>[1 0.25 0.25 0.25 1]</td>
<td>[8.2538 0.0214 0.0216 0.0218 -0.9893]</td>
</tr>
<tr>
<td>II</td>
<td>0.25</td>
<td>[1 0.25 0.25 0.25 10]</td>
<td>[23.1323 0.0584 0.0599 0.0611 -3.0710]</td>
</tr>
<tr>
<td>III</td>
<td>0.25</td>
<td>[1 0.25 0.25 0.25 20]</td>
<td>[30.9837 0.0772 0.0797 0.0819 -4.3023]</td>
</tr>
</tbody>
</table>
Figure 6  The simulation of the closed loop responses of the three selected controller, see Table (1).

The selected controller has been implemented twice to the mechatronic demonstrator; once using the real-time processor of the sbRIO-9631 and another using the FPGA device. The two practical responses are plotted in Figure (7). It is obvious that the reconfigurable PIP-FPGA control system is faster by about a factor of 1.5 than PIP real-time. This is expected since the parallel processing effectively reduces the calculation time. On other words, the PIP-FPGA calculates the control action faster than PIP/real-time; this allows the PIP-FPGA control action to take place earlier than PIP real-time. This causes the demonstrator to interact with PIP-FPGA control action before the end of the time step with a considerable time.

Figure 7  Practical implementation of PIP control using both real-time (dashed) and FPGA (solid) applied on mechatronic demonstrator.
5. Summary and Conclusion

The work shows a successful implementation of PIP control algorithm within Reconfigurable IO board with FPGA technology. Both transfer function parameters estimation step and calculating the control of the updated transfer function parameters are employed to a fast mechatronic demonstrator. The control law has been applied twice via real-time processor and FPGA device.

The implementation of PIP over reconfigurable IO board with FPGA technology shows a considerable enhancement in the response time by a factor of 1.5 over the real-time processor. This is due the parallel processing available within the FPGA chip. The construction of parallel circuits for the control algorithm minimizes the calculation time. This fact allows the control action to take place before the end of the time step by a considerable time. However, the disadvantages are limited in the memory size and the arithmetic manipulation. Signed-Fixed-point math with appropriate range is used in order to simplify the computations and hold the suitable memory size.

Acknowledgement

The authors are grateful to Jazan University, Automatic Control Laboratory, for using the licensed LabVIEW software package.

References


24. C.J. Taylor, P.C. Young, A. Chotai, (2013), True digital control Statistical modelling and non-minimal state space design, John Wiley and Sons Ltd.