Design of a linear time-varying cross-coupled iterative learning controller

K. L. Barton

Douglas A. Bristow
Missouri University of Science and Technology, dbristow@mst.edu

Andrew G. Alleyne

Follow this and additional works at: http://scholarsmine.mst.edu/faculty_work

Part of the Aerospace Engineering Commons, and the Mechanical Engineering Commons

Recommended Citation
http://scholarsmine.mst.edu/faculty_work/1934

This Article - Conference proceedings is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. For more information, please contact weaverjr@mst.edu.
Design of a Linear Time-Varying Cross-Coupled Iterative Learning Controller

Kira L. Barton, Member, IEEE; Douglas A. Bristow, Member, IEEE; and Andrew G. Alleyne, Senior Member, IEEE

Abstract—In many manufacturing applications contour tracking is more important than individual axis tracking. Many control techniques, including Iterative Learning Control (ILC), target individual axis error. Because individual axis error only indirectly relates to contour error, these approaches may not be very effective for contouring applications. Cross-Coupled ILC (CCILC) is a variation on traditional ILC that targets the contour tracking directly. In contour trajectories with rapid changes, high frequency control is necessary in order to meet tracking requirements. This paper presents an improved CCILC that uses a linear time-varying (LTV) filter to provide high frequency control for short durations. The improved CCILC is designed for raster-scan tracking on a Cartesian robotic test platform. Analysis and experimental results are presented.

I. INTRODUCTION

Many manufacturing applications use repetitive processes in order to build numerous identical parts. The tracking performance of these systems is critical to the quality of these parts and ultimately to the success of the company [1]. As these parts move from the macro to the micro and nano-scale, the demand for enhanced precision motion control increases. Iterative learning control (ILC) is a feedforward control technique which improves individual axis tracking of a system through a process known as learning [1,2,3]. ILC uses the tracking errors from past iterations to update the feedforward control input, leading to improved axis tracking. Convergence of the learning process leads to a control input which is optimized for the specific trajectory being learned thereby yielding very low tracking error.

The first order learning algorithm is given by,

\[ u_{j+1}(k) = u_j(k) + L(q)e_j(k) \]  \hspace{1cm} (1)

where \( u \) is the control input, \( e \) is the error, \( k \) is the discrete time index, \( j \) is the iteration index, \( q \) is the forward time-shift operator \( q x(k) \triangleq x(k+1) \), and \( L(q) \) is a linear time-invariant (LTI) learning function. One of the challenges with (1) is that high-frequency signals can propagate from iteration to iteration. High-frequency propagation can lead to undesirable outcomes such as large transient growth and long convergence times [1,4,3]. Some designs for \( L \), such as the plant inverse method, dynamically couple with the plant dynamics in such a way as to nominally prevent high-frequency propagation. These designs, particularly plant model inversion [5], risk being over-designed and may not be robust to uncertainty in the plant dynamics. To prevent high-frequency propagation and achieve robustness, an additional variable, \( Q \), is included in a modified first-order learning algorithm as

\[ u_{j+1}(k) = Q(q)\left( u_j(k) + L(q)e_j(k) \right) \]  \hspace{1cm} (2)

\( Q(q) \) is known as the Q-filter and is often a lowpass filter that removes high-frequency signals from the learning algorithm.

For many precision motion control applications, the reference trajectories include short durations of high frequency content combined with large durations of low frequency content [6], such as Fig. 1.

Fig. 1. Reference raster scanning trajectory.

For these types of trajectories, a time-invariant lowpass Q-filter may be the limiting factor in the ability of the learning controller to achieve low converged tracking error. One method for improving the tracking performance may be to redesign \( L \) so that the bandwidth of \( Q \) can be increased, which may require very accurate models of the plant dynamics. Alternatively, improvements can be made by focusing our attention on the design of a linear time-varying (LTV) Q-filter. Previous work [6] has shown that Q-filter
bandwidth can be increased for short periods of time without inducing large transient growth and long convergence times. This LTV Q-filter is highly effective for trajectories with short durations of high frequency content.

In many manufacturing systems, the desired trajectory requires motion from more than one axis. Typically in these multiaxis systems, individual controllers are designed for each motion axis. This decoupled approach works well for linear trajectories defined by single-axis movements because this control design technique focuses on improving individual axis tracking. However, in many applications, the emphasis is placed on the contour tracking performance. MIMO and SISO ILC only indirectly attack the problem of contour tracking through individual axis control. For applications in which the focus is on contour tracking, it may be beneficial to use a controller that directly focuses on minimizing the contour error.

Cross-coupled ILC (CCILC), introduced in [7], is a new variation on ILC that directly targets contour tracking. This new technique, derived from cross-coupled control (CCC) [8], uses the contour error directly in the learning algorithm. Just as CCC is combined with individual axis feedback control to provide low contour error with good command following, the CCILC can be combined with individual axis ILC. This combined design approach has been shown to be effective in reducing contour errors for a multi-axis robotic test platform [9,7].

This paper focuses on the use of the combined ILC and CCILC algorithm for tracking contour trajectories that exhibit large durations of low frequency control combined with short periods of high frequency control, Fig. 1. To this end, we combine CCILC with an LTV Q-filter design [6] to create a new LTV CCILC.

The remainder of this paper is structured as follows. Section II of this paper introduces the class of systems considered in this paper. The LTV Q-filter learning algorithm is described in Section III. Section IV gives a brief overview of the combined ILC and CCILC system. Stability and convergence analysis is provided in Section V, while a design procedure for the LTV CCILC learning algorithm is presented in Section VI. Experimental results are given in Section VII. Conclusion and future work are discussed in Section VIII.

II. CLASS OF SYSTEMS

While ultimately we are concerned with controlling multiaxis systems, we focus the initial discussion of this work on single-input single-output (SISO) systems. The process can be extended to additional axes, as well as to the cross-coupled learning controller. In order to use learning controllers, the system is assumed to perform repeated operations in which the initial conditions, disturbances and trajectory are assumed identical from iteration to iteration. The input-output relationship of a discrete-time, LTI, SISO system can be given by [3],

\[ y_j(k) = P(q)u_j(k) + d(k) \]  

where \( y \) is the output signal of the plant model, \( u \) is the system control input signal, \( d \) is the exogenous signal (e.g. contains information about the initial states, as well as the disturbances) that is time-varying but not iteration varying, and \( P \) is the plant that is assumed open loop stable.

One can now consider that each of the above signals actually contains N-sampled values,

\[ u_j(k), k \in \{0,1,...,N−1\} \]

\[ y_j(k), k \in \{0,1,...,N−1\}, \]

\[ d_j(k), k \in \{0,1,...,N−1\} \]

including the desired system output defined as

\[ y_{d}(k) \in \{0,1,...,N−1\}. \]

The error signal for the \( j^{th} \) iteration is defined as

\[ e_j(k) = y_{d}(k) − y_j(k). \]

The goal of ILC is to generate a sequence of control inputs, each an improvement from iteration to iteration, that achieves rapid convergence of the error signal to a low converged value.

III. LTV Q-FILTER LEARNING ALGORITHM [6]

An LTV Q-filter can be written as

\[ Q_k(q) = \lambda_{k,N_0} q^{N_0} + \cdots + \lambda_{k,1} q + \lambda_{k,0} + \lambda_{k,1} q^{-1} + \cdots + \lambda_{k,N_0} q^{-N_0} \]  

(7)

where \( \lambda_{k,j} \) are the impulse response coefficients of a lowpass filter with bandwidth \( \Omega(k) \). The description, (7), is general enough to include many different types of filters including causal, noncausal, FIR, and IIR filters of any linear type (Gaussian, Butterworth, Chebychev, Bessel, etc.). Although any filter type may be used for the Q-filter, we will use a Gaussian filter because the filter impulse coefficients can be written explicitly as a function of the bandwidth, which is useful for an LTV filter. The coefficients (7) for the Gaussian filter are given by

\[ \lambda_{k,j} = \frac{1}{\sum_{j=-N_0}^{N_0} \exp \left( \frac{\pi^2 (2 \pi \Omega(k))^2}{S^2 \ln 4} \right)} \quad \left( \frac{\pi^2 (2 \pi \Omega(k))^2}{S^2 \ln 4} \right)^j \]  

(8)

where \( S \) is the sample frequency in Hz.

The first order learning algorithm with an LTV filter is defined as

\[ u_{j+1}(k) = Q_k(q) \left[ u_j(k) + L(q) e_j(k) \right] \]  

(9)

where \( L(q) \) is designed using any general learning function design such as PD-type [10], model inverse [11], and
optimal LQ design [12]. The focus for this paper is on the design of the LTV Q-filter. This does leave open the possibility of a future design procedure which incorporates the design of the learning function into the design of the optimal LTV Q-filter.

IV. COMBINED ILC AND CCILC ALGORITHM

Combined ILC and CCILC [7] is shown in Fig. 2. The combined controller contains three different learning controllers, the individual x- and y-axis controllers and the cross-coupled learning controller. The individual ILC controllers act on the individual axis errors, while the CC learning controller acts directly on the contour error defined as
\[ e = -c_x e_x + c_y e_y \]  
(10)
where \( c_x \) and \( c_y \) are called the coupling gains [13]. These gains are trajectory dependent and generally time-varying. For this work, linearized coupling gains [13] are used to simplify the design process of the cross-coupled learning controller.

![Fig. 2. Block diagram of the combined controller.](image)

A. Learning Algorithm

The general first order learning algorithms for the combined controller are given as
\[
\begin{align*}
    u_{x,i}(k) &= Q_x(q)\left[u_{x,i}(k) + L_x(q)e_x(k) - c_xL_x(q)c_x(k)\right] \\
    u_{y,i}(k) &= Q_y(q)\left[u_{y,i}(k) + L_y(q)e_y(k) + c_yL_y(q)c_y(k)\right]
\end{align*}
\]  
(11)
where \( L_i, i = [x,y,c] \), are the learning functions for the \( x \), \( y \) and contour controllers, respectively. Substituting the contour error \( e \) in (11) with (10) results in a linear time-varying combined learning controller the authors are calling LTV CCILC.

\[
u_{x,i}(k) = Q_x(q)\left[u_{x,i}(k) + L_x(q)e_x(k) - c_xL_x(q)c_x(k)\right]
\]
\[
u_{y,i}(k) = Q_y(q)\left[u_{y,i}(k) + L_y(q)e_y(k) + c_yL_y(q)c_y(k)\right]
\]
\[
\begin{align*}
    u_{x,i}(k) &= Q_x(q)\left[u_{x,i}(k) + L_x(q)e_x(k) - c_xL_x(q)c_x(k)\right] \\
    u_{y,i}(k) &= Q_y(q)\left[u_{y,i}(k) + L_y(q)e_y(k) + c_yL_y(q)c_y(k)\right]
\end{align*}
\]  
(12)

The new aspect of this approach versus [9] is the introduction of the time-varying Q-filter. Convergence analysis for this combined system is presented in the following section.

V. STABILITY ANALYSIS

In this section, stability and convergence properties for the combined control system are presented in a lifted system format. Define \( \hat{Y}_j, \hat{U}_j \) and \( \hat{P} \) as lifted matrices [3].
\[
\begin{bmatrix}
    y_j(1) \\
    y_j(2) \\
    \vdots \\
    y_j(n)
\end{bmatrix} = \begin{bmatrix}
    u_j(0) \\
    u_j(1) \\
    \vdots \\
    u_j(n-1)
\end{bmatrix}
\]
\[
\hat{P} = \begin{bmatrix}
    p_1 & 0 & \cdots & 0 \\
    p_2 & p_1 & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    p_n & p_{n-1} & \cdots & p_1
\end{bmatrix}
\]  
(13)
The linear plant dynamics, (3), can now be written as
\[
\hat{Y}_j = \hat{P}\hat{U}_j + \hat{d},
\]  
(14)
where \( \hat{d} \) is a vector of the same lifted form as \( \hat{Y}_j \) that contains periodic disturbance effects. The elements in the \( \hat{P} \) matrix, \( \{p_1, p_2, \ldots\} \) represent the impulse response of the plant model, also known as the Markov parameters. The lifted tracking error is given by
\[
\hat{E}_j = \hat{Y}_d - \hat{Y}_j
\]  
(15)
with \( \hat{E}_j = [e_j(0), \ldots, e_j(N-1)]^T \) and \( \hat{Y}_j = [y_j(0), \ldots, y_j(N-1)]^T \).

Using (12), (14), and (15) the lifted form of the closed-loop iteration domain dynamics of the combined ILC system is
\[
\begin{align*}
    \hat{U}_{x,i}(k) &= \hat{Q}_x\left[\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\right]\hat{U}_j + \hat{D} \\
    \hat{U}_{y,i}(k) &= \hat{Q}_y\left[\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\right]\hat{U}_j + \hat{D}
\end{align*}
\]
\[
\begin{align*}
    M_{11} &= \left(I - \hat{L}_x + \hat{\mathcal{C}}_x\hat{\mathcal{L}}_x\hat{\mathcal{C}}_x\right)\hat{P}_1 \\
    M_{12} &= \hat{\mathcal{C}}_x\hat{\mathcal{L}}_x\hat{\mathcal{C}}_x\hat{P}_2 \\
    M_{21} &= \hat{\mathcal{C}}_x\hat{\mathcal{L}}_x\hat{\mathcal{C}}_x\hat{P}_1 \\
    M_{22} &= \left(I - \hat{L}_y + \hat{\mathcal{C}}_y\hat{\mathcal{L}}_y\hat{\mathcal{C}}_y \right)\hat{P}_1
\end{align*}
\]  
(16)
where
Using the lifted format, one can see that the control input converges in the iteration domain as long as
\[
\|\lambda (\hat{Q} \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix})\| < 1 \tag{19}
\]
where \(i\) is defined on the interval \(i \in [1, n]\). The spectral radius condition, \((\lambda (\hat{Q}M) < 1)\), satisfies the stability criteria, but does not ensure monotonicity in the iteration domain [14]. A sufficient condition for monotonic convergence is given by:
\[
\|\hat{Q} \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\| < 1 \tag{20}
\]
In this equation, \(\|\cdot\|\) is the induced norm of the matrix. Satisfying (20) with the induced 2-norm ensures monotonicity and stability of the given system.

VI. DESIGN AND OPTIMIZATION OF AN LTV Q-FILTER

In this section we present a design methodology for the LTV Q-filter which is used in conjunction with the combined learning controller to generate an LTV CCILC [15]. This controller will then be applied to a Cartesian robotic test platform in Section VII. While the design results in this section are specific to the robotic test platform, the methodology is general enough to be applied to other motion control systems.

As stated previously, this work focuses on a high-performance design for the Q-filter. We assume that the learning functions have been designed using one of the generally accepted design methods mentioned in Section III.

We design the bandwidth \(\Omega(k)\) of the Q-filter to have roughly the same profile as the initial tracking errors of the multiaxis system, \(e_{xi}(k)\) and \(e_{yi}(k)\). For example, if we consider the raster trajectory shown in Fig. 1, representative initial tracking errors for the robotic system used in Section VII are given in Fig. 3.

The authors choose to use the same type of Gaussian filter as is used for the LTV Q-filter, with the exception of being time-invariant. The bandwidth of \(F(q)\) depends on the particular application/trajectory and in this case is chosen to be 10 Hz to smooth the signal, while still permitting distinct peaks.
Fig. 4. Parameterized profile.

To optimally select the parameters $a_1$, $b_1$, $a_2$, $b_2$, and $c$, we minimize a cost function with two competing objectives. The first objective is to obtain low converged contour error which is measured by the root-mean-square (RMS) of the converged contour error:

$$J_1(a_1, b_1, a_2, b_2, c) = \frac{\|e\|}{\sqrt{N}}$$

(22)

where $e_j$ is the converged value of the contour error defined loosely as $e_j = \lim_{j \to \infty} e_j$, as long as $e_j$ is stable and converges as $j \to \infty$. The second objective is fast convergence, which is defined as the number of iterations of learning required to converge within 1% of the converged contour error.

$$J_2(a_1, b_1, a_2, b_2, c) = \min_j \left\{ J^* : \frac{\|e_j - e_j^*\|}{\|e_j\|} \leq 0.01, \forall j \geq j^* \right\}$$

(23)

The combined cost function is given by

$$J(a_1, b_1, c) = J_1(a_1, b_1, c) + \eta \cdot J_2(a_1, b_1, c), \quad i = 1, 2$$

(24)

where $\eta$ is a design parameter.

To calculate the costs, $J_1$ and $J_2$, for a given set of variables, we first generate the LTV Q-filter using the smooth frequency profile $\Omega'$ with (7) and (8). The learning process is then simulated until the contour error converges. Convergence is assumed when the error change, $\|e_{j+1} - e_j\|$, is less than $10^{-8}$, which typically occurs between 10-500 iterations for the system given in Section VII. The converged error is taken as the final value of the contour error for the simulation and $J_1$ and $J_2$ are calculated from the simulation results using (24) and (25), respectively.

Equation (24) is typically not convex; therefore we use a brute force approach for searching different combinations of variables. Choosing a moderate baseline frequency, $c$, based on known performance and convergence properties of LTI Q-filter designs, a broad search pattern is selected for the remaining $a$ and $b$ variables.

As a final step, in keeping with previous design strategies for CCILC [7], we impose the condition of monotonicity. In order to satisfy this requirement, we check monotonicity of each LTV Q-filter combination using (20). The results of the optimization search are plotted in Fig. 5.

![Fig. 5. Numerical optimization results showing the tradeoff between converged error and learning time.](image)

As can be seen in Fig. 5, monotonicity is calculated for each data point and indicated by color variation on the plot. For this particular example, only monotonic LTI combined CCILC & ILC designs were considered, while the optimized combinations of the $a$, $b$, and $c$ variables produced both monotonic and non-monotonic LTV combined CCILC & ILC controllers.

For this work, monotonically convergent controllers which fell along the $\eta = 0.1$ line were chosen for comparison purposes. These are indicated by the letters A and B in Fig. 5. Table I shows a summary of the selected LTI and LTV controllers. A comparison of the $\Omega$ profiles for the different type of filters is shown in Fig. 6. As can be seen in Table I, the LTV filter resulted in a 14% improvement in the cost function (24) over the LTI filter for the combined system.

| TABLE I COMPARISON OF LTI AND LTV Q-FILTERS |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| $J_1$ | $J_2$ | $J$ | $a_1$ | $b_1$ | $a_2$ | $b_2$ | $c$ |
| LTI combined | 1.57 | 15 | 3.07 | - | - | - | 28 |
| LTV combined | 1.33 | 13 | 2.63 | 20 | 20 | 5 | 40 |
| Improvement | 15% | 13% | 14% | 14% |

Fig. 6 shows that the lower converged error and learning time is achieved by implementing higher bandwidth peaks at the locations where the trajectory rapidly changes and lower bandwidth elsewhere.
VII. APPLICATION: CARTESIAN ROBOTIC TESTBED

To demonstrate the effectiveness of this technique, an LTV CCILC controller is applied to a Cartesian robotic testbed system. The system consists of stacked x, y and z axes all mounted orthogonally to one another. Linear models of the x- and y-axes and proportional-integral-derivative (PID) feedback controllers for the Cartesian robot are given in the appendix.

The individual axis learning functions are designed using model inversion of the open-loop stabilized systems. While nominal plant models are used, the resonances vary depending on the location of the axes within the frame of the robotic testbed. Due to these inconsistencies, the resonances have been eliminated from the learning functions. The individual learning functions are given in the appendix.

The cross-coupled learning function selected for this system is a simple PD-type function of the form

\[ L_x(q) = k_{x_1} q + k_{x_2} (q-1). \]  (25)

PD parameters, \( k_{x_1} = 0.25 \) and \( k_{x_2} = 22 \), are tuned to provide satisfactory contour learning behavior, as well as satisfy the monotonic convergence condition (20). The reference command, shown in Fig. 1, is applied to the testbed as \( y_d \) and \( x_d \), respectively. Optimal LTI and LTV FIR Gaussian Q-filters for \( \eta = 0.1 \) are obtained using the design procedure described in Section VI. Implementing the LTI combined CCILC & ILC and LTV combined CCILC & ILC controllers on the robotic testbed, performance improvements resulting from the use of the LTV design can be seen in Fig. 7.

Fig. 7 illustrates that the LTV combined controller converges fairly quickly, while the LTI combined controller requires a much longer learning time. An overview of the cost function results for the experimental testing is presented in Table II. The LTI combined learning controller converged to a neighborhood of 2.92 \( \mu \)m RMS contour error in 60 iterations, while the LTV combined controller converged to a neighborhood of 0.97 \( \mu \)m RMS contour error in approximately 20 iterations.

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>COMPARISON OF LTI AND LTV Q-FILTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>J ( \mu )m</td>
<td>J ( \mu )m</td>
</tr>
<tr>
<td>LTI combined</td>
<td>2.92</td>
</tr>
<tr>
<td>LTV combined</td>
<td>0.97</td>
</tr>
<tr>
<td>Improvement</td>
<td>67%</td>
</tr>
</tbody>
</table>

Analogous to the simulation results, the experimental results indicate that the lowest cost function was achieved by the LTV combined learning controller with a 67% improvement over the LTI combined learning controller. The increase in the contour error resulting from unmodelled dynamics provides more opportunity for enhanced performance results from the LTV learning controller in the experimental testing. In many applications, one may have a more accurate plant model, in which case the simulation and experimental results would be more closely aligned. However, numerical, simulation and experimental results all indicate that an LTV Q-filter can improve the performance of the combined learning controller over an optimal LTI combined learning controller.

VIII. CONCLUSION AND FUTURE WORK

This paper has presented the basic algorithm and design procedure for an LTV Q-filter to be used in conjunction with individual axis ILC and CCILC. An LTV Q-filter provides a means of increasing the bandwidth of the Q-filter and implementing high-frequency control at precise locations within a trajectory. This is particularly useful in trajectories such as a raster scanning trajectory, where the direction of the reference trajectory changes rapidly from one scanning line to the next, requiring short sections of high frequency control. Stability and monotonic convergence analysis for the LTV combined controller (LTV CCILC) was presented and used, along with a two-
part cost function of converged contour error and learning time, to select optimal LTI and LTV Q-filters. The procedure and optimally designed controllers were implemented on a robotic testbed system. Simulation and experimental results demonstrated that the LTV combined learning controller resulted in a lower cost function than the LTI combined controller.

### APPENDIX CARTESIAN ROBOT MODEL

Dynamic models for the robotic system are of the form [16]

\[
G(z) = \frac{k(z + \alpha_1)(z^2 - \alpha_2 z + \alpha_3)(z^2 - \alpha_4 z + \alpha_5)}{(z - \beta_1)(z - 1)(z^2 - \beta_2 z + \beta_3)(z^2 - \beta_4 z + \beta_5)}. \tag{26}
\]

Numerical values for (26) are presented in Table III.

### TABLE III

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerator</td>
<td>(\alpha_1)</td>
</tr>
<tr>
<td></td>
<td>(\alpha_2)</td>
</tr>
<tr>
<td></td>
<td>(\alpha_3)</td>
</tr>
<tr>
<td></td>
<td>(\alpha_4)</td>
</tr>
<tr>
<td></td>
<td>(\alpha_5)</td>
</tr>
<tr>
<td>Denominator</td>
<td>(\beta_1)</td>
</tr>
<tr>
<td></td>
<td>(\beta_2)</td>
</tr>
<tr>
<td></td>
<td>(\beta_3)</td>
</tr>
<tr>
<td></td>
<td>(\beta_4)</td>
</tr>
<tr>
<td></td>
<td>(\beta_5)</td>
</tr>
<tr>
<td>Gain</td>
<td>(k)</td>
</tr>
<tr>
<td></td>
<td>(G_x)</td>
</tr>
<tr>
<td></td>
<td>(G_y)</td>
</tr>
<tr>
<td></td>
<td>(G_{\alpha})</td>
</tr>
<tr>
<td></td>
<td>(G_{\beta})</td>
</tr>
<tr>
<td></td>
<td>(G_{\delta})</td>
</tr>
</tbody>
</table>

PID feedback controllers used to stabilize the \(x\) - and \(y\)-axis plant models of (26) are given below

\[
C_{i,x}(z) = \frac{38(z^2 - 1.941 z + 0.9423)}{(z - 1)(z - 0.7408)}, \tag{27}
\]

\[
C_{i,y}(z) = \frac{36.5(z^2 - 1.949 z + 0.9506)}{(z - 1)(z - 0.7408)}. \tag{28}
\]

The open-loop stable plant model used in (3) and all subsequent equations is of the form

\[
P_j(z) = G_j(z) \frac{1}{G_i(z)C_{i,j}(z)}, \quad j = 1, 2, i = x, y. \tag{29}
\]

The individual axis learning functions, designed as modified inverse plants, are given as

\[
L_x(z) = \frac{34.1(z - 0.9235)(z - 0.9666)(z^2 - 1.815 z + 0.8303)}{(z - 1)(z - 0.8187)(z - 0.7408)}, \tag{30}
\]

\[
L_y(z) = \frac{11.43(z^2 - 1.939 z + 0.9409)(z^2 - 1.732 z + 0.815)}{(z - 1)(z - 0.8187)(z - 0.7408)}. \tag{31}
\]

### Reference List


