MAPPING CONTROLLERS FROM THE S-DOMAIN TO THE Z-DOMAIN USING MAGNITUDE INVARIANCE AND PHASE INVARIANCE METHODS

A Thesis by

Prathamesh R. Vadavkar

Bachelors of Electronics Engineering, Pune University, 2004

Submitted to the Department of Electrical Engineering and the faculty of the Graduate School of Wichita State University in partial fulfillment of the requirements of the degree of Masters of Science

December 2007
I have examined the final copy of this Thesis for form and content and recommend that it be accepted in partial fulfillment of the requirement for the degree of Master of Science with a major in Electrical and Computer Engineering.

John Watkins, Committee Chair

We have read this Thesis and recommend its acceptance:

Larry Paarmann, Committee Member

Brian Driessen, Committee Member
ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. John Watkins, for his thoughtful, patient guidance and continuous wholehearted support. I would also like to extend my gratitude to the thesis committee members, Dr. Paarmann and Dr. Driessen. Last but not the least, I take this opportunity to thank my family and all my friends who helped me in successful completion of this thesis.
ABSTRACT

Design by emulation has been widely used in the field of control systems. Design by emulation is a process where initially a continuous time controller is designed to achieve desired closed loop specifications. This continuous time controller is then mapped to a digital equivalent using a suitable mapping technique. Methods traditionally used for this mapping include forward rectangular rule, bilinear rule and zero-pole matching.

We are presenting a new approach for mapping a continuous time controller to a discrete time controller. This approach, unlike any of the traditional mapping method, produces a discrete time transfer function with a magnitude response or phase response nearly the same as its analog prototype. To achieve this objective we are using the Magnitude Invariance Method (MIM) and Phase Invariance Method (PIM) that were recently developed in the field of signal processing. The frequency responses and the step responses of the closed loop systems obtained using this approach are systematically investigated to evaluate the effectiveness of these mapping techniques.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>1.1. Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2. Thesis outline</td>
<td>2</td>
</tr>
<tr>
<td>2. REVIEW OF DIGITAL MAPPING TECHNIQUES</td>
<td></td>
</tr>
<tr>
<td>2.1. Backward difference method</td>
<td>3</td>
</tr>
<tr>
<td>2.2. Forward difference method</td>
<td>6</td>
</tr>
<tr>
<td>2.3. Bilinear transform method</td>
<td>8</td>
</tr>
<tr>
<td>2.4. Bilinear transform with pre-warping</td>
<td>10</td>
</tr>
<tr>
<td>2.5. Matched-Z</td>
<td>11</td>
</tr>
<tr>
<td>3. MAPPING VIA MAGNITUDE INVARIANCE AND PHASE INVARIANCE METHOD</td>
<td></td>
</tr>
<tr>
<td>3.1. Magnitude invariance principle</td>
<td>12</td>
</tr>
<tr>
<td>3.2. Phase invariance principle</td>
<td>15</td>
</tr>
<tr>
<td>3.3. Decorrelation by means of cepstral processing</td>
<td>17</td>
</tr>
<tr>
<td>3.4. Determination of digital controller parameters</td>
<td>21</td>
</tr>
<tr>
<td>4. CONTROLLER MAPPING EXAMPLES</td>
<td></td>
</tr>
<tr>
<td>4.1. PD Controller (high pass)</td>
<td>24</td>
</tr>
<tr>
<td>4.2. PI Controller (low pass)</td>
<td>31</td>
</tr>
<tr>
<td>4.3. Lead Controller (high pass)</td>
<td>36</td>
</tr>
<tr>
<td>5. CONCLUSIONS SUGGESTIONS FOR FUTURE WORK</td>
<td></td>
</tr>
<tr>
<td>5.1. Conclusions</td>
<td>44</td>
</tr>
<tr>
<td>5.2. Suggestions for future work</td>
<td>45</td>
</tr>
<tr>
<td>LIST OF REFERENCES</td>
<td>47</td>
</tr>
<tr>
<td>APPENDICES</td>
<td>48</td>
</tr>
<tr>
<td>A. Script file for example 4.1</td>
<td>49</td>
</tr>
<tr>
<td>B. Matlab function for MIM and PIM</td>
<td>51</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Backward difference method</td>
<td>4</td>
</tr>
<tr>
<td>2.</td>
<td>Map of the left-half of the s-plane to the z-plane by backward difference method</td>
<td>5</td>
</tr>
<tr>
<td>3.</td>
<td>Forward difference method</td>
<td>6</td>
</tr>
<tr>
<td>4.</td>
<td>Map of the left-half of the s-plane by to the z-plane by forward difference method</td>
<td>7</td>
</tr>
<tr>
<td>5.</td>
<td>Bilinear transform method</td>
<td>8</td>
</tr>
<tr>
<td>6.</td>
<td>Map of the left-half of the s-plane to the z-plane by bilinear transform method</td>
<td>9</td>
</tr>
<tr>
<td>7.</td>
<td>Homomorphic filtering</td>
<td>18</td>
</tr>
<tr>
<td>8.</td>
<td>Details of the characteristic system $D^*$</td>
<td>18</td>
</tr>
<tr>
<td>9.</td>
<td>Details of inverse characteristic system $D^{-1}$</td>
<td>19</td>
</tr>
<tr>
<td>10.</td>
<td>Block diagram of continuous-time control system</td>
<td>24</td>
</tr>
<tr>
<td>11.</td>
<td>Block diagram of a discrete-time control system</td>
<td>25</td>
</tr>
<tr>
<td>12.</td>
<td>Frequency response of PD controller (M=1,2,3)</td>
<td>26</td>
</tr>
<tr>
<td>13.</td>
<td>Frequency response comparison: MIM versus Tustin for PD controller</td>
<td>27</td>
</tr>
<tr>
<td>14.</td>
<td>Step response comparison: MIM versus Tustin for PD controller</td>
<td>28</td>
</tr>
<tr>
<td>15.</td>
<td>Phase comparison: MIM versus Tustin for PD controller</td>
<td>29</td>
</tr>
<tr>
<td>16.</td>
<td>Inter-sample response of the discrete-time PD controller obtained by MIM</td>
<td>29</td>
</tr>
<tr>
<td>17.</td>
<td>Inter-sample response of the PD controller obtained using Tustin</td>
<td>30</td>
</tr>
<tr>
<td>18.</td>
<td>Frequency response for PI controller (M=1,2,3)</td>
<td>32</td>
</tr>
</tbody>
</table>
LIST OF FIGURES (CONTINUED)

19. Frequency response comparison: MIM versus Tustin for PI controller…………….....33
20. Step response comparison: MIM versus Tustin for PI controller…………………………34
21. Inter-sample response of PI controller obtained using MIM…………………………….35
22. Inter-sample response of PI controller obtained using Tustin………………………...36
23. Frequency response for Lead controller (M=1,2,3)……………………………………………37
24. Phase response comparison: PIM versus Tustin for lead controller……………………...38
25. Step response comparison: PIM versus Tustin for lead controller………………………39
26. Phase comparison: PIM versus Tustin for PI controller…………………………………….40
27. Step Response comparison: PIM versus Tustin for PI controller…………………………41
28. Phase comparison for PI controller (M=3,8) with sampling time Ts=0.01………………...43
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Discrete-time PD controllers obtained using MIM and Tustin</td>
<td>26</td>
</tr>
<tr>
<td>2. Discrete-time PI controllers obtained using MIM and Tustin</td>
<td>31</td>
</tr>
<tr>
<td>3. Discrete-time lead controllers obtained using PIM and Tustin</td>
<td>36</td>
</tr>
<tr>
<td>4. Discrete-time PI controllers obtained using PIM and Tustin</td>
<td>41</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Use of digital computers for controlling physical systems has become more and more popular due to the many advantages of digital control. Because continuous time design tools are abundant and more common with many control engineers, design by emulation is often preferred over direct digital design.

Various methods have been suggested for mapping a controller from the s-domain (continuous-time) to z-domain (discrete-time). The most popular methods used for this type of mapping are [1] [2]: backward difference, forward difference, matched-z, impulse-invariance method and bilinear transform. However, when we map a controller from the s-domain to the z-domain the frequency response of the equivalent digital controller does not remain same as its analog prototype. This happens because the mapping techniques are non-linear and hence distort the shape of the frequency response.

In this thesis we make use of the Magnitude Invariance Method (MIM) [2] and the Phase Invariance Method (PIM) [3] to map a controller from the s-domain to z-domain. Thus the magnitude or phase of the frequency response of the discrete-time controller is nearly equal to that of the continuous-time controller. This condition does not hold for the traditional mapping techniques. This property becomes very helpful in certain cases where the magnitude response or phase response of the controller is an important selection criterion. The magnitude or phase response of the discrete-time controller can be directly related to that of its analog prototype.
1.2 THESIS OUTLINE

This thesis consists of five chapters. Chapter 2 gives an overview of the traditional mapping techniques such as backward difference method, forward difference method, matched-z, bilinear transform method and bilinear transform method with pre-warping. In Chapter 3, the principle of the new design technique is explained in detail along with an explanation about cepstral processing and determination of controller parameters. In Chapter 4, the closed loop step and frequency responses of various examples are systematically investigated. Chapter 5 states conclusions and recommendations for future work.
CHAPTER 2

REVIEW OF DIGITAL OF MAPPING TECHNIQUES

In this chapter the five most common mapping techniques, backward difference method, forward difference method, bilinear transform method, bilinear transform method with pre-warping and matched-z will be reviewed.

2.1 The Backward Difference Method [1]

The basic concept is to represent the given controller transfer function $H(s)$ as a differential equation and then to approximate it by a difference equation. For example, consider the following system,

$$H(s) = \frac{a}{s+a} \quad (2.1)$$

Its equivalent differential equation can be written as

$$\frac{du}{dt} + au = ae \quad (2.2)$$

The above equation can be written in integral form as,

$$u(t) = \int_{0}^{t} [-au(\tau) + ae(\tau)] d\tau \quad (2.3)$$

$$u(kT) = \int_{0}^{kT-T} [-au + ae] d\tau + \int_{kT-T}^{T} [-au + ae] d\tau \quad (2.4)$$

Hence,

$$u(kT) = u(kT-T) + \left\{ \text{area of } (-au + ae) \right\}_{\text{over } kT-T < \tau < kT} \quad (2.5)$$
The backward difference rule follows from taking the amplitude of the approximating rectangle to be the value looking backward from kT toward kT-T.

Thus, the equation for \( u(kT) \) according to backward difference becomes,

\[
\begin{align*}
\quad & u(kT) = u(kT - T) + T[-au(kT) + ae(kT)] \\
\therefore & u(kT) = \frac{u(kT - T)}{1 + aT} + \frac{aT}{1 + aT} e(kT)
\end{align*}
\]

Now, we take the z-transform of the above equation in order to obtain the transfer function found using the backward difference method,

\[
H(z) = \frac{a}{(z - 1) / Tz + a}
\]

By comparing H(s) and H(z), the relation between \( s \) and \( z \) can be noted as shown below,

\[
s \leftarrow \frac{z - 1}{Tz}
\]
The stable portion of the s-plane, i.e., the left half of the s-plane is mapped inside a circle in z-plane with a radius of ½ and centered at 1/2 as shown in the Figure 2.2.

Figure 2.2 Map of the left-half of the s-plane to the z-plane by backward difference method
2.2 Forward Difference method [1]

The basic concept of the forward difference method is very similar to backward difference method. In this method the area is approximated by looking forward from kT-T and taking the amplitude of the rectangle to be the value of the integrand at kT-T.

Thus the equation for \( u(kT) \) according to forward difference method becomes,

\[
\begin{align*}
  u(kT) &= u(kT-T) + T[-au(kT-T) + ae(kT-T)] \\
  &= (1-aT)u(kT-T) + aTe(kT-T)
\end{align*}
\]  

(2.10)

(2.11)

The transfer function for forward difference method, obtained by taking z-transform is as given below,

\[
H(z) = \frac{a}{(z-1)/T + a}
\]  

(2.12)

Thus the relation between \( s \) and \( z \) is as shown below,
Forward difference method may produce unstable poles as shown in Figure 2.4. This is one of the disadvantages of this method.

\[ s \leftarrow \frac{z - 1}{T} \quad (2.13) \]

Figure 2.4 Map of the left-half of the s-plane by forward difference method
2.3 Bilinear Transform Method [1]

This method is also known as Trapezoid substitution method or Tustin method. In this method the area is approximated to be that of the trapezoid formed by taking averages of the rectangles considered in forward and backward difference methods.

\[
\begin{align*}
\left( kT - T \right) & \quad kT
\end{align*}
\]

Figure 2.5 Bilinear transform Method

Thus the equation for \( u(kT) \), according to bilinear transform method is given by,

\[
u(kT) = u(kT - T) + \frac{T}{2} [-au(kT - T) + ae(kT - T) - au(kT) + ae(kT)]
\]

(2.14)

The transfer function from the bilinear transform method is,

\[
H(z) = \frac{a}{(2/T)((z-1)/(z+1)) + a}
\]

(2.15)

Thus the relation between \( s \) and \( z \) is given by,

\[
s \leftarrow \frac{2z - 1}{T(z + 1)}
\]

(2.16)
The stable portion of the s-plane i.e. the left half of the s-plane is mapped inside the unit circle in the z-plane as shown in the Figure 2.6,

Figure 2.6 Map of the left-half of the s-plane to the z-plane by bilinear transform method
2.4 The Bilinear Transform method with Pre-Warping [1]

The bilinear transform method has a frequency warping effect. This means there exist a non-linear relationship between the analog filter frequency $\omega_a$ and the digital filter frequency $\omega$. This effect can be negated by employing a frequency pre-warping technique. In frequency pre-warping, the analog filter frequency is set as,

$$\omega_a = \frac{2}{T} \tan \left( \omega \frac{T}{2} \right)$$  \hspace{1cm} (2.17)

The advantage of frequency pre-warping is that the magnitude frequency response of the digital controller can be matched to that of its analog equivalent for one particular frequency. In the above example the frequency at which the matching is achieved is $\omega_a$. After this step, the design steps for the normal bilinear transform method are followed.
2.5 Matched-Z method [1]

This method is also known as zero pole matching. As the name suggests the poles and zeros of the analog controller are mapped directly into poles and zeros in the z-plane. Consider the following transfer function for the analog controller,

\[ H(s) = \frac{\prod_{k=1}^{M} (s-z_k)}{\prod_{k=1}^{N} (s-p_k)} \]  \hspace{1cm} (2.18)

Then, the transfer function for the digital controller is given by,

\[ H(z) = \frac{\prod_{k=1}^{M} (1-e^{\frac{\alpha T}{s}} z^{-1})}{\prod_{k=1}^{N} (1-e^{\frac{\alpha T}{p_k}} z^{-1})} \]  \hspace{1cm} (2.19)

where T is the sampling time. To avoid the aliasing effect, the sampling time should be very small.
CHAPTER 3

MAPPING VIA MAGNITUDE INVARIANCE AND PHASE INVARIANCE METHOD

In this chapter the magnitude invariance method, phase invariance method and decorrelation using cepstral processing will be explained in detail. Also the method for sequential estimation of the digital controller parameters will be explained.

3.1 Magnitude Invariance principle [2]

This method was proposed in the field of signal processing by Paarmann [2]. This method is unique in a way that it produces a magnitude frequency response of the discrete-time rational transfer function that nearly follows the magnitude frequency of the corresponding continuous-time prototype. This method is denoted as the Magnitude Invariance Method (MIM) [2]. In this mapping technique, it is shown that the autocorrelation function of the unit sample response of the discrete-time system is samples of the autocorrelation function of the Dirac impulse response of the analog prototype convolved with a sinc function. MIM is equivalent to autocorrelation invariance if the magnitude frequency response for the continuous-time prototype, for normalized radian frequencies, is strictly bandlimited to less than $\pi$. MIM is a mapping such that

$$|H(e^{j\omega})| = |H_c(j\omega)|_{\Omega=\omega T} = |H_c(j\omega T)|, \quad |\omega| \leq \pi$$

and hence it is called the magnitude invariance method (MIM). $H$ is the discrete-time transfer function and $H_c$ is the continuous-time transfer function. This would be very advantageous since we will have a discrete-time controller that has a magnitude frequency response the same as that of its analog prototype. In this approach, the starting point is the magnitude-squared frequency response of the analog controller.
**Magnitude invariance and autocorrelation.**

The relationship between magnitude-squared frequency response and the autocorrelation function is well-known in the s-domain.

\[ r_c(\tau) = \frac{1}{2\pi T} \int_{-\infty}^{\infty} |H_c(j\omega/T)|^2 e^{j\omega\tau/T} d\omega, \]  \hspace{1cm} (3.2)

where \( r_c(\tau) \) is the continuous-time autocorrelation function, and \( \omega T \) is used in place of \( \Omega \) for better comparison with the discrete-time form.

In the z-domain,

\[ r[k] = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 e^{j\omega k} d\omega, \]  \hspace{1cm} (3.3)

where \( r[k] \) is a discrete-time autocorrelation function. It is assumed that \( H_c(s) \) and \( H(z) \) are both minimum-phase rational transfer function with real coefficients, and therefore, the magnitude squared frequency response has even symmetry, and \( r_c(\tau) \) and \( r[k] \) are both real and even. If (3.1) holds true, it can be seen that,

\[ r[k] = \frac{1}{2\pi} \int_{-\infty}^{\infty} |H_c(j\omega/T)|^2 W(\omega/T)e^{j\omega k/T} d\omega \tau = kT \]  \hspace{1cm} (3.4)

where \( W(\omega T) \) is a rectangular window,

\[ W(\omega/T) = \begin{cases} 1, & |\omega| < \pi \\ 0, & \text{else} \end{cases} \]

From (3.2)-(3.4) it is clear that

\[ r[k] = T \left. r_c(\tau) \ast w(\tau) \right|_{\tau=kT}, \]  \hspace{1cm} (3.5)
where, * denotes convolution and $w(\tau)$ is the inverse continuous-time Fourier transform of $W(\omega T)$, which can be expressed as follows,

$$w(\tau) = \frac{\sin(\pi \tau / T)}{\pi \tau / T} = \text{sinc}(\tau / T),$$

Form the above equation it is clear that if $|H_c(j\omega T)|^2 = 0 \forall |\omega| \geq \pi$, then

$$r[k] = Tr_c(\tau) | \tau = \kappa T,$$

that is $r[k]$ would be the scaled samples of $r_c(\tau)$, and MIM would be equivalent to an autocorrelation invariance method as stated earlier. The $T$ in above equations is used for frequency scaling. As the algorithm to accomplish MIM is executed in Matlab, $H[k]$, the Discrete Fourier Transform (DFT) will be used in place of $H(e^{j\omega})$, i.e., discretized $\omega$, will be used.
3.2 Phase Invariance principle [3]

This is an algorithmic method that preserves phase characteristics on the \( j\omega \) axis in the \( s \)-plane and on the unit circle in \( z \)-plane. That is, the phase response of the discrete-time controller nearly matches that of continuous-time controller. This method, developed by Paarmann and Atris, is denoted as the Phase Invariance Method (PIM) [3]. The Phase Invariance Method is based on the previously mentioned Magnitude Invariance Method. To achieve phase invariance, first the phase of the analog controller is considered and then Hilbert transform [3] is used to determine the magnitude response. It is important to note that \( \phi[k] = \phi(\Omega)|_{\Omega = \frac{k\pi}{N}} \), i.e., \( \Phi[k] \) are simple samples of the analog phase response (not principal phase) which forces the phase response of the discrete-time system to be the same as the analog phase. Then a discrete-time Hilbert transform is used to find the required \( |H[k]| \). This magnitude response is then treated as the magnitude response of the discrete-time controller. Thus the key concept here is the Hilbert transform. The Hilbert transform as explained in Oppenheim and Schaffer is used with minor modification. In Oppenheim and Schaffer, the desired magnitude is supplied to the Hilbert transform to determine the phase. In the Phase Invariance algorithm, the phase is provided and it is the magnitude that is desired. The algorithm for obtaining magnitude form the desired phase response can be summarized below:

Given the desired phase response \( \phi[k] \),

\[
\xi[n] = \text{IFFT} \{j\phi[k]\},
\]

\[
\gamma[n] = \xi[n] \cdot v_N[n], \quad \text{where}
\]

\[
v_N[n] = \begin{cases} 
0, & n = 0, N/2, \\
1, & n = 1, 2, \ldots, N/2 - 1, \\
-1, & n = N/2 + 1, \ldots, N - 1 
\end{cases}
\]
\[ \alpha[k] = \text{FFT}\{ \gamma[n] \}, \text{and finally} \]

\[ |H[k]| = \exp\{\alpha[k]\} \]

where FFT stands for the Fast Fourier Transform, IFFT stands for the Inverse Fast Fourier Transform and \(N\) is the length of the FFT. Once the magnitude is obtained, the autocorrelation function is determined from the magnitude-squared response. This is done exactly as explained in magnitude invariance principle.

Once the autocorrelation function is obtained, the next step in the algorithm (both MIM and PIM) is to obtain the impulse response by means of cepstral processing. Cepstral processing is explained in detail in Section 3.3.
3.3 Decorrelation by means of Cepstral processing [2]

Decorrelation by means of cepstral processing is well-known in the signal processing field. The decorrelation of the autocorrelation function obtained in MIM and by PIM can be achieved by similar means. By assuming causality, (3.1) can also be expressed as follows:

$$r[k] = \sum_{n=0}^{\infty} h[n]h[k+n].$$

(3.6)

Comparing (3.6) with the convolution sum it can be observed that the only difference is the +n in the argument as opposed to –n in the convolution sum. Hence cepstral processing can be used to obtain the impulse response $h[n]$ from the autocorrelation function $r[k]$. The inverse Fourier transform of $H(e^{j\omega})$ is $h[n]$. If $\tilde{h}[n]$ is defined as the inverse Fourier transform of $H^*(e^{j\omega})$, where $H^*$ denotes complex conjugate of $H$, then (3.6) can be rewritten as follows:

$$r[k] = h[k] * \tilde{h}[k],$$

which means, $r[k]$ is simply the convolution of $h[k]$ and $\tilde{h}[k]$

Consider $|H[m]|^2$ to be the frequency samples of $|H(e^{j\omega})|^2$. Then,

$$H[m]H^*[m] \Leftrightarrow r[k]$$

$$H[m]H^*[m] \Leftrightarrow h[n] * \tilde{h}[n]$$

$h[n] \Leftrightarrow H[m] \Leftrightarrow H[z]$ and,

$$\tilde{h}[n] \Leftrightarrow H^*[m] \Leftrightarrow H(1/z),$$

Then according to the fundamental property of Fourier and $z$ transforms,

$$\tilde{h}[n] = h[-n]$$
Since $h[n]$ is causal and minimum phase, $\bar{h}[n] = h[-n]$ is maximum phase. This now becomes the problem of separating the minimum phase $h[n]$ and maximum phase signal $\bar{h}[n]$. Since $r[n] = h[n] * \bar{h}[n]$, separation can be achieved by simple deconvolution.

Homomorphic filtering in Oppenheim and Schaffer [4] can be employed to obtain the deconvolution. Homomorphic filtering for separation of minimum and maximum phase parts is shown in Figure 3.1

![Homomorphic Filtering Diagram](image)

**Figure 3.1 Homomorphic Filtering** [3]

In Figure 3.1, $D_*$ is the characteristic system for convolution, $D_*^{-1}$ is the inverse of $D_*$, $\hat{r}[n]$ is the complex spectrum of $r[n]$, $l_{mn}[n]$ and $l_{mx}[n]$ are the minimum phase lifter and maximum phase lifter sequences respectively. Thus $h[n]$ is the minimum phase part of $r[n]$ and similarly, $h[-n]$ is the maximum phase part of $r[n]$. The calculation of $\hat{r}[n]$ is done as shown in the Figure 3.2.

![Detailed Diagram](image)

**Figure 3.2 Details of the characteristic system $D_*$** [3]
In this application $R[k]$ is real because, $r[n]$ has even symmetry. Hence $\hat{R}[k] = \ln(R[k])$ is also a real operation. Computing the inverse DFT in order to obtain $r[n]$ is not required since, $R[k] = |H[k]|^2$. Hence $R[k]$ becomes the starting point of the algorithm. Also since $R[k]$ is real and even, $\hat{R}[k]$ and $\hat{r}[n]$ are real and even. Thus in this application of homomorphic filtering phase unwrapping is not required. Thus the lifter $l_{mn}[n]$ is defined as shown below:

$$l_{mn}[n] = \begin{cases} 
0, & n < 0 \\
0.5, & n = 0 \\
1, & n > 0 
\end{cases}$$

The inverse characteristic system $D^{-1}$ is obtained as shown in Figure 3.3

![Figure 3.3 Details of inverse characteristic system $D^{-1}$][3]

In the decorrelation algorithm mentioned above, the lengths of DFTs $N_1$ should be selected such that the DFT can be computed efficiently. It is known that,

$$H(e^{j\omega}) = \sum_{n=0}^{\infty} h[n]e^{-j\omega n},$$

Let

$$\tilde{H}[k] = H(e^{j\omega}) |_{\omega = (2\pi/N_1)k} = \sum_{n=0}^{\infty} h[n]e^{-j(2\pi/N_1)nk},$$

$$k = 0, 1, \ldots, N_1 - 1$$

$\tilde{H}[k]$ therefore, are the Fourier series coefficients of the following:

$$\tilde{h}[n] = \sum_{r=-\infty}^{\infty} h[n + rN_1] = \frac{1}{N_1} \sum_{k=0}^{N_1-1} \tilde{H}[k]e^{j(2\pi/N_1)nk}$$
Thus the decorrelation algorithm mentioned above theoretically results in $\hat{h}[n]$ not $h[n]$. Thus to achieve theoretical precision it is required that $R[k] = |\hat{H}[k]|^2$ be used as the starting point of the algorithm. However if the length of DFT is selected large enough, $\hat{h}[n]$ will accurately represent $h[n]$. The amount of error that may still be present is largely case dependant.

Thus the algorithm can be summarized as follows:

\[
R[k] = |H[k]|^2
\]

\[
\hat{R}[k] = \ln(R[k])
\]

\[
\hat{r}[n] = DFT^{-1}(\hat{R}[k])
\]

\[
\hat{r}_{mn}[n] = \hat{r}[n] \times l_{mn}[n]
\]

\[
\hat{R}_{mn}[k] = DFT(\hat{r}_{mn}[n])
\]

\[
R_{mn}[k] = \exp(\hat{R}_{mn}[k])
\]

\[
h[n] = DFT^{-1}(R_{mn}[k])
\]

Now the only step that remains is to get $H(z)$ from $h(n)$. This step is explained in next section.
3.4 Determination of Digital Controller Parameters [2] [5]

As the impulse response $h[n]$ is causal and minimum phase, all poles of the transfer function in the z-domain will be inside the unit circle. Similarly all zeros will either be on the unit circle or inside the unit circle. Since the zeros at $\infty$ in s-plane will be mapped to origin in the z-plane, the number of poles and zeros will be equal in number. Considering the above properties, a general expression for the transfer function is as shown in (3.7)

$$H(z) = \sum_{i=0}^{M} b_i z^{-i}$$

$$H(z) = \frac{\sum_{i=0}^{M} b_i z^{-i}}{1 + \sum_{i=1}^{M} a_i z^{-i}}$$

(3.7)

$H(z)$ may be obtained by applying the z-transform of $h[n]$. In that case, the following will result:

$$\beta_0 + \beta_1 z^{-1} + \beta_2 z^{-2} + \ldots$$

where the $\beta$ values are the unit sample values, i.e. $\beta_0 = h[0], \beta_1 = h[1], \ldots$. Since it is assumed that $h[n]$ was obtained from the analog prototype transfer function, $h[n]$ must be capable of being modeled as a rational transfer function with the same number of zeros and poles as that of the analog prototype. However, higher order discrete-time transfer function may be required because of the periodic property of $H(e^{j \omega})$. Similarly a higher order transfer function may be required to cope with the cusp of the magnitude response that may occur at $\pm \pi$.

Thus now the problem of determining the transfer function of (3.7), becomes the problem of determining $a_i$ and $b_i$ parameters of (3.7) from the $\beta$ parameters. This can be achieved by a modification of the work of D. Graupe and D.J.Krause [5].

$$\beta_0 + \beta_1 z^{-1} + \ldots = \frac{b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}}{1 + a_1 z^{-1} + \ldots + a_M z^{-M}}$$

(3.8)
By cross multiplying the terms in (3.8) we get,

\[ \beta_0 + (\beta_1 + \beta_0 a_1)z^{-1} + (\beta_2 + \beta_1 a_1 + \beta_0 a_2)z^{-2} + \ldots = b_0 + b_1 z^{-1} + \ldots + b_M z^{-M} \]

By equating the powers of \( z \), we get,

\[ b_0 = \beta_0 \]
\[ b_1 = \beta_1 + \beta_0 + a_1 \]
\[ b_2 = \beta_2 + \beta_1 a_1 + \beta_0 a_2 \]
\[ \ldots \]
\[ b_M = \beta_M + \beta_{M-1} a_1 + \ldots + \beta_0 a_M \]

\[ 0 = \beta_{M+1} + \beta_M a_1 + \ldots + \beta_1 a_M \]
\[ \ldots \]
\[ 0 = \beta_{2M} + \beta_{2M-1} a_1 + \ldots + \beta_M a_M \]

From the above equations, matrices can be formed as shown in (3.9) and (3.10)

\[
\begin{bmatrix}
    a_1 \\
    a_2 \\
    \vdots \\
    a_M
\end{bmatrix} = 
\begin{bmatrix}
    \beta_M & \beta_{M-1} & \ldots & \beta_1 \\
    \beta_{M+1} & \beta_M & \ldots & \beta_2 \\
    \vdots & \vdots & \ddots & \vdots \\
    \beta_{2M+1} & \beta_{2M-2} & \ldots & \beta_M
\end{bmatrix}^{-1}
\begin{bmatrix}
    \beta_{M+1} \\
    \beta_{M+2} \\
    \vdots \\
    \beta_M
\end{bmatrix}
\]

\[(3.9)\]
This is a deterministic process and accurate transfer function parameters can be obtained, because \( h[n] \) is known to be represented by a finite order transfer function. If an \( M^{th} \) order controller is desired, the above matrices will be \( M \) by \( M \) and the first \( 2M+1 \) values of \( h[n] \) would be required.
CHAPTER 4
CONTROLLER MAPPING EXAMPLES

This chapter deals with controller mapping examples. Examples for different type of controllers like, Proportional-Derivative (PD), Lead controller (Lead), and Proportional Integral (PI) are provided. These controllers are mapped using the MIM and PIM algorithms and the results are compared with Tustin. Parameters like order of the controller (M) and the sampling time, Ts, are varied to investigate MIM and PIM algorithms. All the results would be compared against the Tustin method because, amongst all the traditional mapping techniques, Tustin often produces the best results.

4.1 PD controller (High Pass)

Figure 4.1 Block diagram of continuous-time control system

Figure 4.1 shows a block diagram of a continuous-time control system. The plant transfer function $G(s)$ in this system is $G(s) = \frac{54}{s^3 + 12s^2 + 27s}$ [1]. The controller $D(s)$ in Figure 4.1 is a PD type controller, $D(s) = 0.5(s + 4)$ [1], is designed such that the closed loop system has a
overshoot of less than or equal to 4.32\% and a settling time of less than or equal to 1.33 sec. This continuous time controller $D(s)$ is mapped using MIM or PIM to obtain the digital control system as shown in Figure 4.2.

![Figure 4.2 Block diagram of a discrete-time control system](image)

The controller $D(z)$ in Figure 4.2 is obtained by mapping the controller $D(s)$ in Figure 4.1 using MIM. The various values obtained for $D(z)$ are summarized in Table 1.
Table 1 Discrete-time PD controller obtained using MIM and Tustin

<table>
<thead>
<tr>
<th>Order (M)</th>
<th>MIM</th>
<th>Tustin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(D(z) = \frac{6.506z - 4.059}{z + 0.255})</td>
<td>(D(z) = \frac{12z - 8}{z + 1})</td>
</tr>
<tr>
<td>2</td>
<td>(D(z) = \frac{6.976z^2 - 0.952z - 2.47}{z^2 + 0.7109z + 0.066})</td>
<td>NA</td>
</tr>
<tr>
<td>3</td>
<td>(D(z) = \frac{7z^3 + 2.664z^2 - 3.384z - 1.035}{z^3 + 1.228z^2 + 0.375z + 0.019})</td>
<td>NA</td>
</tr>
</tbody>
</table>

Figure 4.3 Frequency response of PD controller (M=1, 2, 3)

Figure 4.3 shows that a digital controller obtained from MIM, having an order 3, matches the magnitude more closely than a controller having an order 1. Thus it can be noted that if the
order of the controller is increased, the magnitude of the analog controller is better matched. The sampling time of 0.1 seconds is used in above example.

Figure 4.4 Frequency Response comparison: MIM versus Tustin for PD controller

It is observed from Figure 4.4 that the controller obtained using MIM with an order of 3 clearly matches the magnitude of the continuous-time controller than the controller obtained using the Tustin method.
Figure 4.5 Step response comparison: MIM versus Tustin for PD controller

Figure 4.5 shows the closed-loop step responses for the analog, MIM, and Tustin controllers. It is seen that the controller obtained using Tustin has lower overshoot than the controller obtained using MIM. This happens because, though MIM does better in terms matching the magnitude, Tustin does better in terms of matching the phase. This is shown below in Figure 4.6.
Figure 4.6 Phase comparison: MIM versus Tustin for PD controller

Figure 4.7 Inter-sample response of the discrete-time PD controller obtained by MIM
Figures 4.7 and 4.8 show the inter-sample response of the controller obtained using MIM and Tustin respectively. It can be seen from the above figures that the control signal for MIM oscillates less as compared to the control signal for Tustin. This happens to be one of the advantages of MIM.
4.2 PI Controller [6]:

Let $G(s)$ in Figure 4.1 be $\frac{25}{s^2 + 9s + 40}$. The controller $D(s)$ in this example is a PI controller such that $D(s) = \frac{10}{s(s + 4)}$. This continuous-time controller is then mapped using MIM to obtain a discrete-time controller as shown in Figure 4.2 Table 2 summarizes different transfer functions obtained for $D(z)$ by MIM and Tustin.

<table>
<thead>
<tr>
<th>Order(M)</th>
<th>MIM</th>
<th>Tustin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$D(z) = \frac{2.028z + 0.546}{z - 0.897}$</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>$D(z) = \frac{1.919z^2 + 1.548z + 0.218}{z^2 - 0.359z - 0.493}$</td>
<td>$D(z) = \frac{0.952z^2 + 1.587z + 0.635}{z^2 - 0.571z - 0.302}$</td>
</tr>
<tr>
<td>3</td>
<td>$D(z) = \frac{1.916z^3 + 2.788z^2 + 1.144z + 0.119}{z^3 + 0.289z^2 - 0.766z + 0.285}$</td>
<td>NA</td>
</tr>
</tbody>
</table>
Frequency response for PI controller (M=1,2,3)

It can be seen from Figure 4.9 that if the order of the controller is increased, the magnitude response of the analog controller is matched with increased accuracy. A sampling time of T=1 seconds is used in this example.
Frequency response comparison: MIM versus Tustin for PI controller

As seen in Figure 4.10, the controller obtained using MIM with an order of M=3 matches the magnitude of the analog controller better than the controller obtained using Tustin.
Step response comparison: MIM versus Tustin for PI controller

Figure 4.11 shows the closed-loop step responses for the analog, MIM, and Tustin controllers. It can be seen that in this example MIM clearly does better than the Tustin because it has a much lower overshoot than Tustin. Thus in certain examples the controller obtained using MIM does better even in terms of producing the step response closer to that of the analog closed loop system. Figures 4.12 and 4.13 show the inter-sample response for MIM and Tustin respectively. Again it is clear from the figures that the control signal from MIM oscillates less than the control signal from Tustin.
Figure 4.12 Inter-sample response of PI controller obtained using MIM
Now, similarly, PIM results will be investigated against Tustin method in next parts.

4.3 Lead Example [1]

Here the antenna angle tracker example form [1] is considered. The plant transfer function in Figure 4.1 is $G(s) = \frac{1}{10s^2 + s}$. A lead controller $D(s) = \frac{10s + 1}{s + 1}$ is designed to have a peak overshoot less than or equal to 16% and to have a settling time less than or equal to 10 seconds. This continuous-time controller $D(s)$ is then mapped using PIM to have a discrete-time control system as shown in Figure 4.2. Table 3 below, summarizes different transfer functions obtained using MIM and Tustin. A sampling time of $T = 1$ second was used.

Figure 4.13 Inter-sample response of PI controller obtained using Tustin
Table 3 Discrete-time Lead controller obtained using PIM and Tustin

<table>
<thead>
<tr>
<th>Order</th>
<th>PIM</th>
<th>Tustin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$D(z) = \frac{3.503z - 2.531}{z - 0.028}$</td>
<td>$D(z) = \frac{7z - 6.333}{z - 0.333}$</td>
</tr>
<tr>
<td>2</td>
<td>$D(z) = \frac{6.921z^2 - 1.779z - 3.957}{z^2 + 0.437z - 0.252}$</td>
<td>NA</td>
</tr>
<tr>
<td>3</td>
<td>$D(z) = \frac{7.345z^3 + 1.996z^2 - 5.743z - 1.867}{z^3 + 0.966z^2 - 0.950z - 0.140}$</td>
<td>NA</td>
</tr>
</tbody>
</table>

Figure 4.14 Frequency response for Lead controller (M=1, 2, 3)
It is observed in Figure 4.14 that, as the order of the controller is increased, the phase of the analog controller is matched more accurately. The controller obtained using PIM with an order of 3 does better in terms of the matching the phase than Tustin. This is shown in Figure 4.15.

![Phase Diagram](image)

**Figure 4.15 Phase response comparison: PIM versus Tustin for lead controller**

It can be seen from Figure 4.16, that the controller obtained using PIM has a lower overshoot than the controller obtained using Tustin. This may be because the controller obtained using PIM does better in terms of matching the phase of the analog controller.
Figure 4.16 Step response comparison: PIM versus Tustin for lead controller

The PI controller mentioned earlier is matched using PIM. Figure 4.17 and Figure 4.18 shows its performance in terms of phase matching and step response respectively.
Figure 4.17 Phase comparison: PIM versus Tustin for PI controller
Figure 4.18 Step Response comparison: PIM versus Tustin for PI controller

It is observed that the controller obtained using PIM matches the phase of the analog controller better and has a lower overshoot than the controller obtained using Tustin. The discrete-time transfer functions obtained using PIM are summarized in Table 4 below:
Table 4 Discrete-time PI controller obtained using PIM and Tustin

<table>
<thead>
<tr>
<th>Order</th>
<th>MIM</th>
<th>Tustin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(D(z) = \frac{0.689z + 0.839}{z - 0.939})</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>(D(z) = \frac{0.9805z^2 + 1.421z + 0.413}{z^2 - 0.707z - 0.181})</td>
<td>(D(z) = \frac{0.952z^2 + 1.587z + 0.635}{z^2 - 0.571z - 0.301})</td>
</tr>
<tr>
<td>3</td>
<td>(D(z) = \frac{0.989z^3 + 1.905z^2 + 1.044z + 0.132}{z^3 - 0.229z^2 - 0.571z - 0.033})</td>
<td>NA</td>
</tr>
</tbody>
</table>

Figure 4.19 shows the phase response of analog controller, a controller obtained using PIM with an order of \(M=3\), and a controller obtained using PIM with an order of \(M=8\). Both the controllers are mapped with a sampling time \(T=0.01\) seconds. It can be seen from the figure that the controller obtained using PIM with an order of \(M=3\) can no longer match the phase of the analog controller. Thus as the sampling time is decreased the order of the controller needs to be increased in order to match the phase of the analog controller.
Phase comparison for PI controller (M=3, 8) with sampling time Ts=0.01
CHAPTER 5

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

5.1 Conclusions

Two methods have been presented that map a controller from the s-domain to the z-domain such that they produce a digital controller with either the same magnitude or phase response as that of its analog prototype. It is noted that in some cases the order of the digital controller obtained is required to be more than its analog equivalent. This happens because of the periodic property of $H(e^{j\omega})$ and also because of the cusp in the magnitude observed around $\pm \pi$. Similarly, higher order digital controllers may be required if the sampling time is decreased. This method is not limited for strictly proper systems. As shown in one of the examples (PD), it can be implemented on improper systems too. As seen in the example for a PD type controller, the magnitude invariance method does better in terms of matching the magnitude of the analog controller but has more overshoot as than Tustin. In the PI example, it is observed that the controller obtained using MIM does better in terms of both matching the magnitude of the analog controller as well as step response performance. In the lead controller and PI controller examples it is seen that the control signal from the controller obtained using MIM is less oscillating than the control signal from the controller obtained using Tustin. As seen in both the lead controller and PI controller examples, the controller obtained using PIM matches the phase of the analog controller and also has lower overshoot as seen in the closed loop step response. Thus it can be concluded that both MIM and PIM gives better results in terms of magnitude and phase matching, respectively than the traditional mapping techniques. In certain cases, it is seen that MIM and PIM does better in terms of the step response as well.
5.2 Suggestions for future work

For future work, improvement in the suggested method can be achieved by improving phase of the digital controller obtained using MIM and by improving the magnitude of the digital controller obtained using PIM. Since it is seen in the examples that both MIM and PIM does better in the case of PI controller (low pass), another approach that can be used is transforming the PD or Lead controller (high pass), to its analog prototype and then mapping it using MIM or PIM. This would require high pass to low pass and low pass to high pass transformations. A detailed analysis of the computational errors present if any, involving in the algorithmic procedure can be carried out. A more accurate procedure for determining the controller transfer function may be established.
REFERENCES
LIST OF REFERENCES


This is a script file for generating Figure 4.12 - Figure 4.15 for PD type controller explained in Section 4.1.

%%% PD example %%%
clear all;
clc;
a1=[0 54];  % plant zeroes
b1=[1 12 27 0];  % plant poles
G=tf(a1,b1);  % plant

% Designing controller to meet following specifications
% 1. P.O < 4.32%
% 2. Ts = 1.33 sec

%%% Analog Controller
a2=[0.5 2];  % controller zeroes
b2=[0 1];  % controller poles
Ds=tf(a2,b2);  % controller
[Dz1] = c2dn(Ds,1,'mim',1,8192*128);
[a2,b2,T] = tfdata(Dz1);
[Dz2] = c2dn(Ds,1,'mim',2,8192*128);
[Dz3] = c2dn(Ds,1,'mim',3,8192*128);
figure(1);
bode(Ds,'r-',Dz1,'k-',Dz2,'k--',Dz3,'k-o');
legend('analog','Md=1','Md=2','Md=3');
grid on;
DzT=c2d(Ds,T,'tustin');  % Discrete controller obtained using Tustin
%DzZ=c2d(Ds,T,'zoh');

Gz=c2d(G,T);  % Discretized plant
Gcl=(G*Ds)/(1+(G*Ds));
Gcl=minreal(Gc1);
Gzcl=(Gz*Dz3)/(1+(Gz*Dz3));
GzclT=(Gz*DzT)/(1+(Gz*DzT));
%GzclT=(Gz*DzZ)/(1+(Gz*DzZ))
figure(2);
step(Gcl,Gzcl,GzclT);
legend('analog','MIM','tustin');
grid on;
figure(3);
bode(Ds,'r-',Dz3,'k-o',DzT,'k-x');
legend('analog','MIM','Tustin');
grid on;
APPENDIX B
Matlab Function for MIM and PIM

This is function which maps an analog controller to discrete controller using Magnitude Invariance Method (MIM) or Phase Invariance Method (PIM)

```
function [Dz] = c2dn(Ds,Ts,method,Md,NA);
%[Dz] = c2dn(Ds,Ts,method,Md,NA) maps the analog controller to digital
% using MIM or PIM.
% Dz is the digital equivalent of the controller.
% Ds is the analog controller to be mapped.
% Ts is the required sampling time.
% Method indicates MIM or PIM.
% Md is the order of the filter.
% NA is no of samples.

% Written by Prathamesh Vadhavkar and Dr. John Watkins.
% 10/07/2007

% First two input arguments are required; the other three have default
% values

[a2,b2] = tfdata(Ds,'v');
l=length(b2);
md=(l-1);

if nargin < 5, NA = 4096; end
if nargin < 4, Md = md; end
if nargin < 3, method = 'mim'; end
if nargin < 2, Ts = 1; end

% Scaling
T = Ts;   % Sampling time

k = length(a2);
l = length(b2);
n=k-1;
for x=1:k
    t1(x) = T^(x-1);
end
for x=1:l
    t2(x) = T^(x-1);
end
for x = 1:k
    a2(x) = a2(x)/(t1(k-(x-1)));
end
for x = 1:l
    b2(x) = b2(x)/(t2(l-(x-1)));
end
```
```matlab
switch lower(method);
    case 'mim'
        NA = NA;
        NA2 = NA/2;
        MD = Md;           % order of the controller
        w = 0:pi/NA2:pi*(NA2-1)/NA2;    % w = 0 to pi
        R1 = freqs(a2,b2,w);
        R1 = abs(R1).^2;
        w = pi:-pi/NA2:pi/NA2;     % w = pi to 0
        R2 = freqs(a2,b2,w);
        R2 = abs(R2).^2;
        R = [R1,R2];            % magnitude squared response
    end

switch lower(method);
    case 'pim'
        NA=NA;
        NA2=NA/2;
        MD=Md;
        w = 0:pi/NA2:pi*(NA2-1)/NA2;
        HW = freqs(a2,b2,w);
        MagH = abs(HW);
        PH = angle(HW);
        MagHF = MagH(1:NA2);
        PhaseH = PH;
        PhaseH2 = -fliplr(PhaseH);
        PhaseHF = PhaseH2(1:NA2);
        Mag = [MagH MagHF];
        Phase = [PhaseH PhaseHF];
        PhaseJ = Phase*i;
        beta = ifft(PhaseJ,NA);
        %
        ll = 0:1:NA2-1;
        v1 = ll.*0;
        v2 = exp(v1);
        v2(1) = 0;
        v3 = -v2;
        vN = [v2 v3];
        %
        gamma = beta.*vN;
        alpha = fft(gamma,NA);
        MagD = abs(exp(alpha));
        R = MagD.^2;
    end

    % Homomorphic filtering
    lm1 = 0.5;
    de = 1:NA2-1;
    lm2 = de.^0;
    de = 1:NA2;
    lm3 = de*0;
    lmn = [lm1,lm2,lm3];
```
%% Obtaining Impulse response using Cepstral Processing
h = log(R);  % R^(k)
h = ifft(h);  % complex spectrum r^(n)
h = h.*lmn';  % minimum phase sequence r^mn
h = fft(h);  % R^mn(k)
h = exp(h);  % Rmn
h = ifft(h);  % h(n)
h = real(h);

% Set up matrices and vectors:

%  
%  for y = 1:2*MD  
%     for k = 1:MD  
%        betaM1(y,k) = h(MD+1+y-k);  
%     end  
%  end  
%  for y = 1:MD  
%     for k = 1:MD  
%        if k>y  
%           betaM2(y,k) = 0.0;  
%        end  
%        if k<=y  
%           betaM2(y,k) = h(y+1-k);  
%        end  
%     end  
%  end  

betaV1 = h(MD+2:3*MD+1);
betaV2 = h(2:MD+1);

% Compute the filter coefficients:
%  
% a3 = (conj(betaM1))';
a4 = a3 * betaM1;
a5 = (conj(betaM1))' * betaV1;
a1 = - a4 \ a5;
b1 = betaV2 + (betaM2 * a1);
bE(1) = h(1);
aE(1) = 1.0;
for z = 1:MD;  
    bE(z+1) = b1(z);  
    aE(z+1) = a1(z);  
end  

Dz=tf(bE,aE,T);  % Discrete controller obtained using MIM/PIM
Dz=(Dz/(dcgain(Dz)))*dcgain(Ds);