Predictive Control For Dynamic Systems To Track Unknown Input In The Presence Of Time Delay

2005

Yulan Li
University of Central Florida
Predictive Control for Dynamic Systems to Track Unknown Input in the Presence of Time Delay

by

Yulan Li
B.S. Central South University of Technology, 1994
M.S. Central South University of Technology, 1997

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the School of Electrical Engineering and Computer Science in the College of Engineering at the University of Central Florida Orlando, Florida

Fall Term
2005
ABSTRACT

This study investigated a tracking system to trace unknown signal in the presence of time delay. A predictive control method is proposed in order to compensate the time delay. Root locus method is applied when designing the controller, parameter setting is carried out through error and trail technique in w-plane. State space equation is derived for the system, with special state chose of tracking error. To analyze the asymptotic stability of the proposed predictive control system, the Lyapunov function is constructed. It is shown that the designed system is asymptotically stable when input signal is rather low frequency signal.

In order to illustrate the system performance, simulations are done based on the data profile technique. Signal profiles including acceleration profile, velocity profile, and trajectory profile are listed. Based on these profiles, simulations can be carried out and results can be taken as a good estimation for practical performance of the designed predictive control system.

Signal noise is quite a common phenomenon in practical control systems. Under the situation that the input signal is with measurement noise, low pass filter is designed to filter out the noise and keep the low frequency input signal. Two typical kinds of noise are specified, i.e. Gaussian noise and Pink noise. Simulations results are displayed to show that
the proposed predictive control with low-pass filter design can achieve better performance in the case of both kinds of noise.
# TABLE OF CONTENTS

LIST OF TABLES ................................................................. viii

LIST OF FIGURES ............................................................ ix

CHAPTER 1 INTRODUCTION ..................................................... 1

1.1 Background ................................................................. 1

1.2 Proposed Control ......................................................... 2

1.3 Simulations ................................................................. 3

CHAPTER 2 PREDICTIVE CONTROL USING ROOT LOCUS METHOD IN W-PLANE ......................................................... 4

2.1 Problem Statement ......................................................... 4

2.2 Predictive Control System ................................................ 5

2.3 Root Locus Method Application ........................................ 7

2.4 Parameter Tuning in W-Plane ........................................... 9

2.5 Tuning Outcomes and Simulation Results ............................ 10
LIST OF TABLES

2.1 Parameters Tuning Summary ........................................ 18

2.2 Parameters Tuning Summary For Profile Signal .................. 22
LIST OF FIGURES

2.1 Tracking system ............................................. 5
2.2 Predictive Control System .................................... 6
2.3 Equivalent System ............................................... 7
2.4 Flow Chart of the Tuning Procedure ............................. 11
2.5 Root locus plot (in the w-plane) ................................. 12
2.6 Tracking error .................................................... 12
2.7 Root locus plot for the second-round tuning of p (in the w-plane) ............................................. 13
2.8 Tracking error .................................................... 13
2.9 Tracking error under an alternative value of k .................. 14
2.10 Root locus plot upon tuning the zero of the controller (in the w-plane) ............................................. 15
2.11 Tracking error after tuning both zero and pole of the controller ............................................. 15
2.12 Tracking error under a larger gain k ................................ 16
2.13 Tracking error with smaller steady state error but increased settling time ............................................. 17
2.14 Acceleration profile ............................................. 19
4.10 Tracking error of predictive control for Gaussian noise . . . . . . . . . . . . . 42
4.11 Tracking error of predictive control with low-pass filter for Gaussian noise . 42
4.12 Tracking error of predictive control for pink noise . . . . . . . . . . . . . . . 43
4.13 Tracking error of predictive control with low-pass filter for pink noise . . . 43
CHAPTER 1

INTRODUCTION

In this thesis, a predictive control tracking problem is studied. The proposed solution to this problem has a potential to applications in practical output tracking systems with an input time delay.

1.1 Background

Time delay is common in many engineering systems. Delays often degrade the control and make the stabilization of the close loop system become more difficult. Considerable research has been devoted to the control of system with time delay \[1\][4][5][6][8][9][10][16][17].

For example, some classical approaches to dead time compensation include: (1) Smith predictor controller (SPC). The SPC was proposed by Smith et al.(1959). The controller incorporates a model of the system, thus to predict the system variables, and the controller may then be designed assuming that the system is delay-free. SPC works well for set-point changes, but the performance in regulating against disturbances is limited and can not stabilize an
unstable system[5]. (2) Finite spectrum assignment control (FSA). The FSA originated with
Manitius and Olbrot (1979). The FSA can arbitrarily assign the closed loop poles and there-
fore can be applied to poorly damped and unstable systems. It can be applicable if and only
if certain function space controllability is fulfilled[18]. (3) Model reduction control (MRC).
The MRC was introduced by Kwon and Pearson (1980). It introduce a linear transformation
to reduce the system into delay-free. MRC is widely used for system with an input delay
under the condition that the system matrix is stabilizable. (4) Time delay approximations.
This technique is applied mostly in the situation of linear systems with a single constant
delay.

1.2 Proposed Control

In this thesis, what we consider is a situation which is specified with: (1) An input time
delay produced in the process of measurement and transmission is considered. (2) The input
signal to be tracked is an unspecified low frequency signal other than a set-point signal.
Based on these two specifications, a predictive control is proposed. In this predictive control
design, in order to compensate the input time delay, a delay is introduced into the feed
back loop. Discrete-time approximation is used in controller design. Compare with other
predictive control techniques dealing with time delay, this is simple and direct to implement
under the situation as to track an unknown signal with input delay.

In order to compensate the time delay, predictive control is introduced as an effective and
direct means. Feed back delay compensator is added in control loop thus to compensate the input delay. Discrete-time approximation technique is applied when designing the controller. Root locus method is applied in controller design, with a trial and error solution to tune the parameters.

Stabilization of control systems with time delay has been studied extensively. In this thesis, a general format of the asymptotic stable problem of a time delay system is analyzed and through the Lyaponuv function construction, system stable condition is given with proof. The desired input signal is always with noise, under this situation, filter is designed according to the specification of the noise. To be practical, both Gaussian noise and pink noise is considered in this thesis.

1.3 Simulations

The performance of proposed predictive control with low pass filter is studied through simulations and profile modelling.

To analyze the signal with modelling, signal profiles are investigated, including the acceleration profiles, velocity profiles and trajectory profiles. These profiles can help to estimate the performance of the control design.
CHAPTER 2

PREDICTIVE CONTROL USING ROOT LOCUS

METHOD IN W-PLANE

In this chapter, a tracking system with time delay is described. A predictive control method is proposed according to presence of the time delay. The system design is carried out using the root locus method. To better analysis root locus plot in discrete time domain, \( w \)-transform is introduced. Through a means of trial and error, the controller parameters can be tuned so as to achieve better performance.

2.1 Problem Statement

As introduced in the previous chapter, the problem studied in this work is a tracking control problem with the presence of time delay. The purpose of the design is to track an unknown input signal, and let the controlled actuator follow the change of this input signal. In this process, there is a large time delay caused by the measurement and transmission of the input signal and this time delay is twice of the sampling period. The system diagram is shown in
Figure 2.1: Tracking system

The control design purpose is to predict and track the unknown input signal with a smallest tracking error and a small response time. Due to the large time lag caused by measurement and transmission time, a predictive control system is proposed in order to control as well as to compensate the time delay.

### 2.2 Predictive Control System

In the tracking system, the input signal we wish to track is captured by the measurement device, and the processing of the data and other factors contribute a time delay to the input signal. The design of our tracking control system is to produce a real-time estimate of instantaneous unknown input signal without any delay. Thus, the control to be designed is a predictive control. Consider the situation that this unknown signal is a relatively slow process. In comparison, the controlled plant for tracking is an actuator which is with much
faster response. Thus, if appropriate control structure is chosen, it is possible to make the actuator track the input signal real-time.

Below is the predictive control structure used to design the proposed tracking control, and it is two degree of freedom control structure with a feed forward controller to be designed and feedback compensator that matches the dynamics to be compensated for (i.e., the delay).

![Figure 2.2: Predictive Control System](image)

In Figure 2.1, $r$ is the input signal, $d(z)$ is the time delay, $e$ is the error fed into controller, $G_c(z)$ is the controller to be designed, $G_p(z)$ is the actuator dynamics, and $y$ is the actuator output. As mentioned earlier, the control design objective is to make the actuator output track the unknown input signal even though only a delayed version of the signal is available.

To see why the closed loop system in Figure 2.2 is chosen as the predictive control structure, we need convert Figure 2.2 to a standard tracking control system. To this end, it follows from the Figure 2.2 that

$$y = (rd - yd)G_cG_p \Rightarrow \frac{y}{e} = dG_cG_p$$  \hspace{1cm} (2.1)
The above equation means that the control structure in Figure 2.2 is conceptually equivalent to the following diagram, and the latter is the standard input-output tracking structure in which the time delay is now in the closed loop and can be compensated for through a control design.

![Figure 2.3: Equivalent System](image)

Having obtained Figure 2.3, we can now invoke one of the standard control design methods (here the classical control design is pursued) and proceed with design.

### 2.3 Root Locus Method Application

The root-locus method is a well-known tool for linear control system analysis and parameter setting. In general, the root-locus is formed by the geometric loci of the characteristic equation:

\[ 1 + dG_cG_p = 0 \]  \hspace{1cm} (2.2)

And, the corresponding open-loop transfer function is:

\[ G_l = dG_cG_p \]  \hspace{1cm} (2.3)
According to the root-locus design method, we have the following angle condition:

$$\angle G_l = -\pi$$  \hspace{1cm} (2.4)

And we have the magnitude condition:

$$|G_l| = 1$$  \hspace{1cm} (2.5)

Through solving these two conditions, controller parameters can be obtained. From the above discrete-time domain design, we know that, despite of the adequate choices of digital controller transfer function and the desired closed loop poles, the resulting closed loop system has another pair of complex conjugate poles whose values correspond to under-damped (a larger overshoot) and slowly convergent response. Since the classical control design does not place all the closed loop poles, we need to tune the digital controller so that the transient response of the system is improved and becomes comparable to that under continuous-time controller. We will use the root locus method and adjust the controller parameters so as to optimize the transient response. To this end, let us parameterize the controller as

$$G_l(z) = k\frac{(z + a)^2}{(z + p)^2}$$  \hspace{1cm} (2.6)

in which, adjustable control parameters are: (1) Open loop pole position p; (2) Open loop zero position a; (3) Gain k

In this case, the tuning problem is to determine optimal values for parameters 'a', 'p', and 'k' such that the closed loop system has better performance in both transient and steady state. In what follows, a tuning procedure is proposed, followed by the tuning results.
## 2.4 Parameter Tuning in W-Plane

The w-transform is developed for discrete systems so that most of design features in the s-plane for continuous systems are retained. Its essential idea of the method is to transform a discrete model \( G(z) \) by substituting \( z \) by a new variable \( w \) of the following bilinear mapping:

\[
w = \frac{2}{T} \frac{z - 1}{z + 1} \tag{2.7}
\]

The design is then carried out for \( G(w) \) as if the system were in the continuous domain. Once designed or analyzed, the resulting transfer function (either open loop transfer function or controller) can be transformed from the w-plane back to the z plane using the inverse of the above mapping, i.e.,

\[
z = \frac{1 + wT/2}{1 - wT/2} \tag{2.8}
\]

Hence, the controller to be designed has the following transfer function in the w-plane:

\[
G_c(w) = \frac{(1 - wT/2)^2 [1 + wT/2 + a(1 - wT/2)]^2}{(1 + wT/2)^2 [1 + wT/2 + p(1 - wT/2)]^2} \tag{2.9}
\]

Using the above transfer function, control parameters can be tuned using the root locus method. Below is the procedure of using w-plane root locus method to adjust design parameters in order to achieve better performance.

1) The startup setting is that the initial open loop poles of the controller are at \( p=-1 \). Use the root locus’ angle condition and calculate the open loop zero, 'a'.

2) Map the transfer function of the controller into the w-plane and plot the corresponding
3) According to the w-plane root locus, we can adjust \( a \) and \( p \) in order to get a larger open-loop gain \( k \) and better locations for the corresponding closed loop poles (other than the two dominant poles that have been placed). Keep \( a \) constant first, adjust \( p \) until the root locus branches have been shifted enough to the left half plane so that the system can be made stable with a larger open-loop gain and that better closed loop poles can be achieved. Once a good value of \( p \) is found, keep it constant and adjust \( a \) toward the same goals.

4) For the set of control parameters, simulate using simulink to validate the design. This tuning procedure is summarized by the flow chart in Figure 2.4.

2.5 Tuning Outcomes and Simulation Results

At the beginning, select and use the initial pole location \( p = -1 \). It follows from the angle condition that \( a = -0.4274 \). Now, map the open loop transfer function to the w-plane, and draw the corresponding root locus plot as in Figure 2.5.

Using the root locus plot in Figure 2.5, the designer can move the cursor and determine the open-loop gain for closed loop pole placement. It is apparent that any gain value larger than \( k = 1.02 \) will result in excessively large overshoot and settling time. At \( k = 1.02 \), using the Simulink model, system performance can be evaluated. As expected, the tracking error plot
Select S-domain
Closed loop pole

\[ z = e^{Ts} \]

Select \( p = 1 \)

Compute open loop zero ‘a’
(using the angle condition)

Open loop transfer function \( G_L(z) \)

Adjust ‘a’ and ‘p’ value

w-plane transfer function \( G_L(w) \)

Draw root locus of \( G_L(w) \)

Is the root locus optimal?

Select open loop gain ‘k’

Simulink simulation

Tracking error small enough?

Y

End

N
Figure 2.5: Root locus plot (in the w-plane)

in Figure 2.6 shows that, although the steady state error is good, settle time is too long. To improve the settle time, change the open-loop pole location to \( p = -0.98 \) and keep the value of 'a' the same as before (i.e., \( a = -0.4274 \)). The corresponding root locus plot is shown in Figure 2.7, and gain \( k \) can be chosen up to \( k = 1.08 \) and better closed loop pole locations (in
terms of overshoot and settling time) than before can be achieved.

![Root Locus Plot](image)

Figure 2.7: Root locus plot for the second-round tuning of p (in the w-plane)

If we select the same gain $k=1.02$ (so as to maintain the steady state performance). Its simulation shows the tracking error as in Figure 2.8. The result shows that, through the tuning of $p$, better settling time is achieved while the steady state performance is maintained.

![Tracking Error Plot](image)

Figure 2.8: Tracking error
Should a smaller gain $k=0.775$ is chosen with the tuned value of $p$, the tracking error as shown below converges faster while the magnitude of steady state error is slightly larger.

![Tracking error under an alternative value of $k$](image)

Figure 2.9: Tracking error under an alternative value of $k$

Next, we shall proceed with tuning of the zero of the controller. Let us keep $p=-0.98$ and change the value of 'a' to $a=-0.47$. The resulting root locus plot in w-plane is given below, and it shows the two root locus branches (closer to the imaginary axis) bend further into the left half plane. As a result, a larger open-loop gain or better settling time can be obtained.

Let us fix $k=0.973$. This choice is close to the original value of $k$, but the resulting closed loop poles are further into the left open-half plane and hence yield better transient performance.

The following simulation result from Simulink verifies this design choice.

Comparing Figures 2.11 and 2.6, we know that the proposed tuning procedure is quite effective. At this point, we can conclude that the proposed discrete domain design is successfully carried and that the tuning procedure is efficient.
Figure 2.10: Root locus plot upon tuning the zero of the controller (in the w-plane)

Figure 2.11: Tracking error after tuning both zero and pole of the controller
It should be noted that the basic structure of the proposed control (basically an approximate type-2 control with common poles and zeros) is instrumental in solving the tracking problem under study. The designer can choose to explore various options in the basic structure, and many of them have not been studied; for instance, the controller can have 2 different zeros. According to the root locus shown in Figure 2.10, a larger open loop gain $k=1.02$ can be selected. If so, the steady state tracking error (shown below) will be slightly improved to be within 0.02mm (as compared to that in Figure 2.11).

Should an even larger open loop gain of $k=1.18$ be selected, the resulting tracking error shown in Figure 2.13 indicates the tradeoff that any improvement of steady state response will result in an increase of settle time. Through such comparison, we know that, given the same selections of $p$ and $a$, the value of $k=1.02$ is fairly close to be optimal. Similarly, the design can explore the value combinations of '$a$' and '$p$' using the procedures outlined above. It turns out that, for the system under study, the set of poles and zeros given by $p=-0.98$
Figure 2.13: Tracking error with smaller steady state error but increased settling time
and $a=-0.47$ are approximately the best and $k=1.02$ is the most suitable gain corresponding
to the set. In Table 2.1, the tuning procedure is displayed with parameters and tracking
error results.

2.6 Data Profile Analysis and Simulations

As stated in previous section, for the tracking system studied in this work, a predictive
control design is proposed. To investigate its effectiveness, we can do some analysis on data
profiles. Although the reference signal is an unknown signal with random features, it has
some common specifications and can be simulated according to its specified profiles. In this
section, we carry out an analysis regarding to the data profiles, including the acceleration
<table>
<thead>
<tr>
<th>Gain</th>
<th>Zero</th>
<th>Pole</th>
<th>std(e)</th>
<th>max(e)</th>
<th>min(e)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.02</td>
<td>-0.4274</td>
<td>-1</td>
<td>0.0283</td>
<td>0.1029</td>
<td>-0.1122</td>
</tr>
<tr>
<td>1.02</td>
<td>-0.4274</td>
<td>-0.98</td>
<td>0.0204</td>
<td>0.1029</td>
<td>-0.0939</td>
</tr>
<tr>
<td>0.775</td>
<td>-0.4274</td>
<td>-0.98</td>
<td>0.0209</td>
<td>0.1129</td>
<td>-0.0601</td>
</tr>
<tr>
<td>1.08</td>
<td>-0.4274</td>
<td>-0.98</td>
<td>0.0247</td>
<td>0.1129</td>
<td>-0.0749</td>
</tr>
<tr>
<td>1.02</td>
<td>-0.47</td>
<td>-0.98</td>
<td>0.0176</td>
<td>0.1029</td>
<td>-0.0673</td>
</tr>
<tr>
<td>1.02</td>
<td>-0.5</td>
<td>-0.98</td>
<td>0.0183</td>
<td>0.1029</td>
<td>-0.0490</td>
</tr>
<tr>
<td>1.02</td>
<td>-0.6</td>
<td>-0.98</td>
<td>0.0258</td>
<td>0.1029</td>
<td>-0.0342</td>
</tr>
</tbody>
</table>
profile, velocity profile and trajectory profile of the input signal.

According to real time data observations, we can define the input signal with following acceleration profile:

\[
\begin{align*}
    a(t) &= a_{\text{max}}, \quad t \in [t_0, t_0 + T_1] \\
    a(t) &= 0, \quad t \in [t_0 + T_1, t_0 + T_1 + T_2] \\
    a(t) &= -a_{\text{max}}, \quad t \in [t_0 + T_1 + T_2, t_0 + 2T_1 + T_2] \\
    a(t) &= 0, \quad t \in [t_0 + 2T_1 + T_2, t_0 + T_1 + T_2 + T_3]
\end{align*}
\] (2.10)

The above 4 intervals constitute a half of the cycle. Then, change the sign of the acceleration to get the full cycle. For example, let us set \( T_1 = T_2 = T_3 = 1\sec \), and set \( a = 0.5 \) the acceleration profile is:

![Acceleration profile](image)

Figure 2.14: Acceleration profile
If we do integration on the acceleration profile, the velocity profile will be:

\[
\begin{align*}
    v(t) &= a_{\text{max}}(t - t_0), \quad t \in [t_0, t_0 + T_1] \\
    v(t) &= a_{\text{max}}T_1, \quad t \in [t_0 + T_1, t_0 + T_1 + T_2] \\
    v(t) &= a_{\text{max}}(t_0 + 2T_1 + T_2 - t), \quad t \in [t_0 + T_1 + T_2, t_0 + 2T_1 + T_2] \\
    a(t) &= 0, \quad t \in [t_0 + 2T_1 + T_2, t_0 + T_1 + T_2 + T_3]
\end{align*}
\]  

(2.11)

For example, let us set \(T_1 = T_2 = T_3 = 1\, \text{sec}\), and set \(a = 0.5\) the velocity profile is as in Figure 2.15.

![Velocity profile](image)

**Figure 2.15: Velocity profile**

If we further do integration on the velocity profile, the trajectory profile will be:

\[
\begin{align*}
    p(t) &= p(t_0) + \frac{1}{2}a_{\text{max}}(t - t_0)^2, \quad t \in [t_0, t_0 + T_1] \\
    p(t) &= p(t_0 + T_1) + a_{\text{max}}T_1(t - t_0 - T_1), \quad t \in [t_0 + T_1, t_0 + T_1 + T_2] \\
    p(t) &= p(t_0 + T_1 + T_2) - \frac{1}{2}a_{\text{max}}(t_0 + 2T_1 + T_2 - t)^2 + \frac{1}{2}a_{\text{max}}T_1^2, \quad t \in [t_0 + T_1 + T_2, t_0 + 2T_1 + T_2] \\
    p(t) &= p(t_0 + 2T_1 + T_2), \quad t \in [t_0 + 2T_1 + T_2, t_0 + T_1 + T_2 + T_3]
\end{align*}
\]  

(2.12)
For example, let us set $T_1 = T_2 = T_3 = 1\sec$, and set $a = 0.5$ the trajectory profile is as shown in Figure 2.16.

![Trajectory profile](image)

Figure 2.16: Trajectory profile

Using the same procedure discussed in the previous section, we can tune the parameters and set the controller parameter with best performance. As shown in Table 2.2, the best tracking performance can be achieved by set gain $k=0.63$ and zero$=-0.57$, with the pole$=-0.98$. The only difference with the tuning regarding to sine wave signal is that we need to consider the positive and negative maximum tracking error so as to select the best performance parameters.

In Figure 2.17 the tracking error is shown for trajectory as in Figure 2.16. For another example, if in the case of $T_1 = 1\sec, T_2 = T_3 = 0$, and $a = 0.25$, the trajectory profile is as in Figure 2.18, and the tracking error will be as in Figure 2.19.

Based on these profiles, we can do simulations and can estimate the tracking performance for real time data.
Table 2.2: Parameters Tuning Summary For Profile Signal

<table>
<thead>
<tr>
<th>Gain</th>
<th>Zero</th>
<th>Pole</th>
<th>std(e)</th>
<th>max(e)</th>
<th>min(e)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.02</td>
<td>-0.4274</td>
<td>-0.98</td>
<td>0.0099</td>
<td>0.0306</td>
<td>-0.0294</td>
</tr>
<tr>
<td>1.08</td>
<td>-0.4274</td>
<td>-0.98</td>
<td>0.0111</td>
<td>0.0289</td>
<td>-0.0261</td>
</tr>
<tr>
<td>0.775</td>
<td>-0.47</td>
<td>-0.98</td>
<td>0.0049</td>
<td>0.0181</td>
<td>-0.0163</td>
</tr>
<tr>
<td>0.6</td>
<td>-0.47</td>
<td>-0.98</td>
<td>0.0049</td>
<td>0.016</td>
<td>-0.0137</td>
</tr>
<tr>
<td>0.63</td>
<td>-0.47</td>
<td>-0.98</td>
<td>0.0048</td>
<td>0.0163</td>
<td>-0.014</td>
</tr>
<tr>
<td>0.63</td>
<td>-0.57</td>
<td>-0.98</td>
<td>0.0052</td>
<td>0.0134</td>
<td>-0.010</td>
</tr>
<tr>
<td>0.63</td>
<td>-0.45</td>
<td>-0.98</td>
<td>0.0050</td>
<td>0.0171</td>
<td>-0.015</td>
</tr>
<tr>
<td>0.63</td>
<td>-0.6</td>
<td>-0.98</td>
<td>0.0056</td>
<td>0.0135</td>
<td>-0.0096</td>
</tr>
<tr>
<td>0.43</td>
<td>-0.65</td>
<td>-0.98</td>
<td>0.0095</td>
<td>0.0219</td>
<td>-0.0144</td>
</tr>
<tr>
<td>1.05</td>
<td>-0.7</td>
<td>-0.98</td>
<td>0.006</td>
<td>0.0211</td>
<td>-0.0170</td>
</tr>
<tr>
<td>1.05</td>
<td>-0.57</td>
<td>-0.98</td>
<td>0.0046</td>
<td>0.0191</td>
<td>-0.0171</td>
</tr>
<tr>
<td>0.7</td>
<td>-0.6</td>
<td>-0.98</td>
<td>0.0052</td>
<td>0.0139</td>
<td>-0.0103</td>
</tr>
<tr>
<td>0.74</td>
<td>-0.68</td>
<td>-0.98</td>
<td>0.0068</td>
<td>0.0168</td>
<td>-0.0116</td>
</tr>
</tbody>
</table>
Figure 2.17: Tracking error

Figure 2.18: Trajectory profile
Figure 2.19: Tracking error
CHAPTER 3

LYAPUNOV STABILITY ANALYSIS

A very important research problem in control theory is that of controlling the output of the system so as to achieve asymptotic tracking of prescribed trajectories.[1] In this thesis work, as discussed in the previous chapter, the design objective is to track an unknown input signal with the presence of time delay, and the design of the predictive control system is described in the previous chapter. In this chapter, the stability of the predictive control system needs to be analyzed. Lyapunov method is employed as the control stability design technique. By constructing a Lyapunov function, a condition of asymptotical stability is given, under which condition, the predictive control system is asymptotically stable. This conclusion is meaningful and can be applied to other time delay systems.

3.1 Problem Statement

Given the input delay system as in Figure 3.1:

where, delay is $e^{-Ts}$, and $T$ is a constant.
And where,

\[ G_p(s) = \frac{y(s)}{u(s)} = \frac{(\tau_1 s + 1)(\tau_2 s + 1)}{s^2} \] (3.1)

So we have

\[ \ddot{y} = \tau_1 \tau_2 \ddot{u} + (\tau_1 + \tau_2) \dot{u} + u \] (3.2)

As the purpose of the tracking is to make the system asymptotically stable and to have minimum tracking error, let \( e(t) = r(t) - y(t) \), Then

\[ \ddot{e} = \ddot{r} - \ddot{y} = \ddot{r} - \tau_1 \tau_2 \ddot{u} - (\tau_1 + \tau_2) \dot{u} - u \] (3.3)

Let state

\[ x_1 = e + \tau_1 \tau_2 u, \] (3.4)

And that is:

\[ x_1 = \dot{e} + \tau_1 \tau_2 \dot{u} \] (3.5)

\[ = x_2 - (\tau_1 + \tau_2)u \] (3.6)
From (3.1.7), we can have another state

\[ x_2 = \dot{e} + \tau_1 \tau_2 \dot{u} + (\tau_1 + \tau_2)u \]  
(3.7)

And let us define

\[ \dot{x}_2 = \dot{r} - u \]  
(3.8)

Therefore, we have

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} +
\begin{bmatrix}
-\tau_1 - \tau_2 \\
-1
\end{bmatrix}
\begin{bmatrix}
u(t - T) \\
u(t)
\end{bmatrix} +
\begin{bmatrix}
0 \\
1
\end{bmatrix}
\begin{bmatrix}
\dot{r}(t)
\end{bmatrix}
\]  
(3.9)

\[
e =
\begin{bmatrix}
1 & 0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} - \tau_1 \tau_2 u(t - T)
\]  
(3.10)

Above are the state space equations of the tracking system with time delay. Further analysis can be produced based on these equations.

### 3.2 Stability Analysis

The above system in 3.1.9 and 3.1.10 can be described as

\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Bu(t - T) + E\dot{r}(t), \\
e(t) &= Cx(t) + Du(t - T)
\end{align*}
\]  
(3.11)

(3.12)

where

\[
A =
\begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix}, \quad
B =
\begin{bmatrix}
-\tau_1 - \tau_2 \\
-1
\end{bmatrix}, \quad
C =
\begin{bmatrix}
1 & 0
\end{bmatrix},
\]
And
\[
E = \begin{bmatrix}
0 \\
1
\end{bmatrix}, \quad D = \begin{bmatrix}
-\tau_1 \tau_2 \\
0
\end{bmatrix}.
\]

In above, \(x(t)\) is the state, \(u(t)\) is the input, \(e(t)\) is the tracking error, \(A, B, C, D, \text{and} E\) are system matrices. \(T\) is the time delay. And \(\ddot{r}(t)\) is the double derivative of the reference signal, since it is an unknown bounded signal, it can be treated as a disturbance \(\xi(t)\).

Thus, we write (3.2.1) as:
\[
\dot{x}(t) = Ax(t) + Bu(t - T) + \xi(t), \quad (3.13)
\]

Consider a controller
\[
u(t) = -\gamma B^T P x(t) \quad (3.14)
\]
such that the tracking error \(e(t)\) will be asymptotically zero, where \(\gamma > 0\) and \(P > 0\) is the solution to the Riccati equation:
\[
A^T P + PA - \gamma P B B^T P + Q = 0; Q > 0 \quad (3.15)
\]

Substitute (3.2.4) to (3.2.3) derives:
\[
\dot{x}(t) = Ax(t) - B\gamma B^T P x(t - T) + \xi \quad (3.16)
\]

Construct a Lyapunov function as:
\[
V = x(t)^T P x(t) + \alpha \int_{t-T}^t x(s)^T x(s) ds + \gamma \int_{t-T}^t x^T(s) P B B^T P x(s) ds, \quad (3.17)
\]
where, $\alpha$ is real positive number. We need to figure out under what condition, the system is asymptotically stable.

The derivative of (3.2.7) is:

\[
\dot{V} = \dot{x}^T(t)Px(t) + x^T(t)P\dot{x}(t) + \alpha x^T(t)x(t) - \alpha x^T(t-T)x(t-T) \tag{3.18}
\]
\[
+ \gamma x^T(t)PBB^TPx(t) - \beta x^T(t-T)PBB^TPx(t-T)
\]
\[
= x^T(t)A^TPx(t) + z^T(t)PAx(t) - Z^T(t-T)PBB^T\gamma P x(t)
\]
\[
- x^T(t)P\gamma BB^TPx(t) + x^T(t)\alpha x(t) - x^T(t-T)\alpha x(t-T)
\]
\[
+ \beta x^T(t)PBB^TPx(t) - \gamma x^T(t-T)PBB^TPx(t-T)
\]
\[
+ \xi^T(t)Px(t) + x^T(t)P\xi(t)
\]
\[
= x^T(t)[A^TP + PA + \alpha]x(t)
\]
\[
- [x^T(t-T)PB - x^T(t)PB][\gamma B^TPx(t) - \gamma B^T P x(t-T)]
\]
\[
- x^T(t-T)\alpha x(t-T) - 2\gamma x^T(t-T)PB^TBPx(t-T) + 2x^T(t)P\xi(t)
\]
\[
\leq -x^T(t)Jx(t) + 2x^T(t)P\xi(t)
\]

If following condition is satisfied:

\[
\|\xi\| \leq \mu\|x\| \tag{3.19}
\]

We have

\[
\dot{V} \leq -\lambda_{\text{min}}(J)\|x(t)\|^2 + 2\lambda_{\text{max}}(P)\mu\|x(t)\|^2
\]

where,

\[
J = A^TP + PA + \alpha I \tag{3.20}
\]
Hence, system is asymptotically stable if

\[
\mu < \frac{\lambda_{\min}(J)}{2\lambda_{\max}(P)}, J > 0
\]  

(3.21)

When reference signal is a slow frequency signal, \( \ddot{r}(t) \) is bounded so that \( \mu \) is bounded and thus condition (3.2.11) is easy to be qualified.

### 3.3 Simulations

To verify the result, let’s consider a simple example for simulation: let \( \tau_1 \) and \( \tau_2 = 0 \). And let choose the generated acceleration profile data as \( \ddot{r}(t) \) as shown in below figure. (The generation of the trajectory profile will be stated in Chapter 5.) We obtain the tracking error and control input plot in Figure 3.3: From the tracking error plot, we can see the control result is rather satisfying. In another figure, the control input plot is shown for the purpose to know about the control input time series.

![Trajectory profile](image.png)

Figure 3.2: Trajectory input
Figure 3.3: Tracking error

Figure 3.4: Control input
CHAPTER 4

FILTER DESIGN

Signal noise is very common in the practical systems. In this thesis, the reference signal we want to track is an unknown signal, and due to the existence of measurement noise, it is necessary to have a filter to get rid of these noise before the predictive control loop. According to the specifications of the reference signal, its frequency belongs to low frequency range up to 0.5Hz. In this case, a low pass filter is considered in order to diminish the measurement noise. In this work, two major types of noise are considered: Gaussian noise and pink noise.

4.1 Gaussian Noise Overview

In practice, a control system must maintain its effectiveness and robustness in the presence of noises and disturbances. For the system under study, the measurement noise is the most obvious. Thus, performance of the proposed predictive control must be investigated with respect to the types of noises the system may have. In this section, the effect of Gaussian
random noise is studied (whereas another class of noises, 1/f noise, will be studied in the next section).

To study the effect of Gaussian noise, we begin with determining the likely values of statistics measures that character the level of noises in the system. It is well known that the probability of a Gaussian random variable $X$ falls in the interval $[m - a, m + a]$ is:

$$P[m - a \leq X \leq m + a] = \text{erf}\left[\frac{a}{\sigma_x}\right]$$  \hspace{1cm} (4.1)

where $\text{erf}(.)$ is a standard function in Matlab, $m_x$ is the mean, and $\sigma_x^2$ is the variance.

Let us suppose that signal to noise ratio at the input end of the system is 5. This means that

$$SNR = \frac{\text{Signal Amplitude}}{\text{Noise level}} = 5$$  \hspace{1cm} (4.2)

For example, if signal magnitude is 0.5, the noise level will be 0.1mm. Next, if the probability of $P[-0.1 \leq X \leq 0.1] = 95\%$, invoking command line ’erfinv(0.95)’ in Matlab yields: $\text{erfinv}(0.95) = 1.3859$. Therefore, $\sigma_x$ can be calculated as follows:

$$\frac{0.1}{\sqrt{2\sigma_x}} = 1.3859 \Rightarrow \sigma_x = 0.0510$$  \hspace{1cm} (4.3)

### 4.2 Pink Noise Overview

In the real world, there are many physical processes whose noises do not have uniform power density over all frequencies as Gaussian noise does. Instead, many of these noises such as
audio noise have the so-called "pink" distribution of power. "Pink" noise is also called 1/f noise, which has an even distribution of power if the frequency is mapped in a logarithmic scale. A straightforward example would be that there is as much noise power in the octave 200 to 400 Hz as there is in the octave 2,000 to 4,000 Hz. White noise has the same distribution of power for all frequencies, so there is the same amount of power between 0 and 500 Hz, 500 and 1,000 Hz or 20,000 and 20,500 Hz. Pink noise has the same distribution of power for each octave, so the power between 0.5 Hz and 1 Hz is the same as between 5,000 Hz and 10,000 Hz. Since power is proportional to amplitude squared, the energy per Hz will decline at higher frequencies at the rate of about -3dB per octave.

Generate pink noise from Gaussian noise Gaussian noise is generated to have a "white" spectrum. A spectral shaping filter can then be multiplied to the spectrum of a white noise to generate noises of any given desired frequency domain amplitude function. This makes it possible for us to approximately generate 1/f noises by implementing the spectral shaping filter. An approximation to the ideal pinking filter can be realized by a simple 3rd order filter whose three poles and three zeros are:

<table>
<thead>
<tr>
<th>pole</th>
<th>zero</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.99572754</td>
<td>0.98443604</td>
</tr>
<tr>
<td>0.94790649</td>
<td>0.83392334</td>
</tr>
<tr>
<td>0.53567505</td>
<td>0.07568359</td>
</tr>
</tbody>
</table>

Its response follows closely the ideal -3dB/octave curve to within +/-0.3dB over a 10 octave range from 0.0009 to 0.9 Nyquist frequency.
Frequency response of the 3rd-order pink noise filter is in Figure 4.1. The Simulink diagram to generate pink noise from white noise is in Figure 4.2.

And the generated pink noise and its power spectrum periodogram is as in Figure 4.3.

Figure 4.1: Frequency response of pink noise generator

Figure 4.2: Simulink model of pink noise generation
4.3 Filter Design

It is shown that the input signal has a maximum frequency of 0.5Hz, while the random signal has many high frequency components. A properly designed filter can be used to attenuate those frequencies higher than 0.5Hz. Our design of filters will be carried out by choosing a cutoff frequency of 1Hz.

It would desirable to have an ideal low pass filter (as shown in the following figure) whose amplitude is 1 for frequencies less than or equal to 1Hz and is zero elsewhere and whose phase is also zero up to the cutoff frequency. Such an ideal filter could complete erase all the high-frequency components from the noise while having little distortion on the original signal.

Of course, such an ideal filter is not physically or digitally achievable. Thus, any filter design is to find a realizable filter whose magnitude is close to that of the ideal filter and whose angle (phase lag) is small (but always nontrivial). In what follows, two types of filters are
designed and compared: Chebyshev, and Butterworth.

(1) Chebyshev Type I filter

The Chebyshev lowpass filter has an amplitude response chosen to maintain a maximum allowable attenuation in the passband while maximizing the attenuation in the stopband. The amplitude response of a Chebyshev filter is of the form:

\[ |H_c(f)| = \frac{1}{\sqrt{1 + \epsilon^2 C_n^2(f)}} \]  \hspace{1cm} (4.4)

The parameter \( \epsilon \) is specified by the maximum allowable attenuation in the passband, and \( C_n() \), known as a Chebyshev polynomial, is given by:

\[ C_n(f) = 2(f/f_c)C_{n-1}(f) - C_{n-2}(f), \quad n = 2, 3, \ldots \]  \hspace{1cm} (4.5)

where,

\[ C_1(f) = f/f_c, \quad C_0(f) = 1 \]  \hspace{1cm} (4.6)

In Matlab, command \([b,a] = \text{cheby1}(n,Rp,Wn)\) generates an nth-order Chebyshev lowpass digital filter with a normalized cutoff frequency of \( Wn \) and with \( Rp \) (in dB) of peak-to-peak
ripple in the passband. The command line is returned with the vector of filter coefficients of dimension (n+1). Given the return row vectors b and a of coefficients in descending powers of z, the filter has the following transfer function:

\[ H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + \cdots + b(n + 1)z^{-n}}{1 + a(2)z^{-1} + \cdots + a(n + 1)z^{-n}} \] (4.7)

Normalized cutoff frequency is the frequency at which the magnitude response of the filter is equal to -Rp dB. For cheby1 filter, the normalized cutoff frequency Wn is a number between 0 and 1, where 1 corresponds to the Nyquist frequency, that is, \( \pi \) radians per sample. Smaller values of passband ripple Rp lead to wider transition widths (shallower rolloff characteristics).

Given that data are sampled at 30 Hz, a 1st-order lowpass Chebyshev Type I filter with 0.5 dB of ripple in the passband and a cutoff frequency of 1 Hz is: \([b,a] = \text{cheby1}(1,0.05,1/15);\) that is:

\[ H(z) = 0.2313 + 0.2313z^{-1} \]

(4.8)

Frequency response of the filter is in Figure 4.5:

(2) Butterworth filter

The Butterworth filter is a filter design chosen to approximate a constant amplitude response in the passband with the cost of less stopband attenuation. An nth-order Butterworth filter is with transfer function:

\[ H_B(s) = \frac{\omega_3^n}{(s - s_1)(s - s_2) \cdots (s - s_n)} \] (4.9)

where, the poles are symmetrical to real axis and \( f_3 = \omega_3/2\pi \) is the 3 - dB cutoff frequency.

In Matlab, command \([b,a] = \text{butter}(n,Wn)\) renders an nth-order lowpass digital Butterworth
Figure 4.5: Frequency response of Cheby filter

filter with normalized cutoff frequency Wn. The return is row vectors b and a of filter coefficients, of length (n+1), and in descending powers of z. The cutoff frequency is defined to be the frequency at which the magnitude of the filter is of its DC gain. For a Butterworth filter, normalized cutoff frequency Wn must be a number between 0 and 1, where 1 corresponds to the Nyquist frequency (pi radians per sample). Based on this characteristic of Butterworth filter, it is necessary to shift its cutoff frequency from 1Hz to 2.5Hz in order to make its frequency response satisfy the condition.

Given that data are sampled at 30 Hz, a 1st-order lowpass Butterworth filter with a cutoff frequency of 2.5 Hz is: [b,a] = butter(1,2.5/15); i.e.,

$$H(z) = \frac{0.2113 + 0.2113z^{-1}}{1 - 0.5774z^{-1}} \quad (4.10)$$

Frequency response of the 1st-order lowpass Butterworth filter is given by Figure 4.6.
Figure 4.6: Frequency response of Butterworth filter

4.4 Predictive Control with Low-pass Filter

As stated in Chapter 1, in this thesis work, the system studied is a tracking system for unknown input signal with the presence of time delay and with the presence of the measurement noise. Therefore, the designed control is a system of predictive control with low pass filter, the aim is to first filter out the measurement noise, then track the reference signal under the predictive control. Figure 4.7 shows the system diagram. To illustrate the effect of adding low pass filter to control system, let’s take the trajectory profile data as shown in Figure 4.8
as the input signal: For this trajectory data, the tracking error of predictive control is as in Figure 4.8:

![Figure 4.8: Predictive control with low-pass filter](image)

If Gaussian noise is injected, the tracking error of predictive control will be as shown in Figure 4.9:

![Figure 4.9: Tracking error of predictive control for noise free case](image)

in Figure 4.10. With the designed Cheby filter added before the predictive control loop, the tracking error of predictive control with filter will be as in Figure 4.11. For the case of pink
Figure 4.10: Tracking error of predictive control for Gaussian noise

Figure 4.11: Tracking error of predictive control with low-pass filter for Gaussian noise
noise, if pink noise is injected, without filter, the tracking error of the predictive control is as in Figure 4.12. With the designed Cheby filter added before the predictive control loop, the tracking error of predictive control with filter will be as in Figure 4.13. From the above

Figure 4.12: Tracking error of predictive control for pink noise

Figure 4.13: Tracking error of predictive control with low-pass filter for pink noise
simulation results, it is clear that the predictive control with filter has better performance than the predictive control itself given the situation of noise injections.
CHAPTER 5

CONCLUSION

5.1 Summary

In this thesis, a tracking system with input time delay is studied. As discussed in Chapter 2, a predictive control method is proposed to track an unknown input signal with the compensation for the time delay. Root locus method is applied when designing the controller, parameter setting is carried out through error and trail technique in w-plane.

In order to illustrate the system performance, simulations are done based on the data profile technique. Signal profiles including acceleration profile, velocity profile, and trajectory profile are listed. Based on these profiles, simulations can be carried out and results can be taken as a good estimation for practical performance of the designed predictive control system.

In Chapter 3, state space equation is derived for the system, with special state chose of tracking error. To analyze the asymptotic stability of the proposed predictive control system, the Lyapunov function is constructed. It is shown that the designed system is asymptotically stable when input signal is rather low frequency signal.
Signal noise is quite a common phenomenon in practical control systems. Under the situation that the input signal is with measurement noise, low pass filter is designed to filter out the noise and keep the low frequency input signal. In Chapter 4, two typical kinds of noise is specified, i.e. Gaussian noise and Pink noise. And simulations results are displayed to show that the proposed predictive control with low-pass filter design can achieve better performance in the case of both kinds of noise.

5.2 Conclusion

It is shown in Chapter 2 that the proposed predictive control system is a direct and easy to implement means to cope with the problem of output tracking with input time delay. The effectiveness is shown through simulation results.

Under the assumption that input signal is a low frequency signal, the system performance will be easy to confined to asymptotically stable. Choosing state variable with tracking error is a effective way to characterize the stability of output tracking problem. To simplify the analysis, the double-derivative of input signal can be taken as a disturbance such that Lyapunov method can be applied.

The designed low pass filter including Chebyshev filter and Butterworth filter has a good effect in dealing with Gaussian noise and Pink noise. And together with the proposed predictive control, the tracking system for unknown input signal with time delay can achieve good performance.
Further work can focus on improvement with Particle filter or Extended Kalman filter as a way to handle the random noise.
APPENDIX A

M-FILES
A.1 Program 1. Predictive Control Parameter Calculation

% closed loop dominant poles: s=sigma+omega*j

clear all; close all;

sigma = -10*pi;
omega = 10*pi;
T=0.033; %

x=0.98; % open loop pole

z=exp(T*(sigma+omega*j)) % transfer from s-domain to z-domain

rz=real(z); % real part of z

iz=imag(z); % imaginary part of z

phase1=phase(z^2*(z-x)^2); phase2=(-pi+phase1)/2;

a=(iz-tan(phase2)*rz)/tan(phase2)

num=[1 a]; % from k(z+a);

num=conv(num,[1 a]) % from z+a

den=conv([1 -x],[1 -x]); % from 1/(z-1)^2

den=conv(den,[1 0 0]) % from 1/z^2
[numw,denw] = z2w(num,den,T) %transfer to w-plane

figure;rlocus(numw,denw); %draw root locus in w-plane
A.2 Program 2. Transfer From z-plane to w-plane

```
% Z2W.m

function [numw,denw] = z2w(numz,denz,T)

% [numw,denw] = z2w(numz,denz,T)

% THIS FUNCTION COMPUTES THE W-TRANSFORM OF A GIVEN DISCRETE
% SYSTEM IN TRANSFER FUNCTION FORM WITH SAMPLING PERIOD T.
% The numerator is padded with leading zeros, if neccessary,
% to make it the same length as the denominator.

[m,n] = size(numz); numz=[zeros(m,length(denz)-n) numz];

[phi,gamma,H,J] = tf2ss(numz,denz); [n,m] = size(phi); I = eye(n);

r = sqrt(T); Q = I + inv(I + phi)*(I - phi); A = 2/T*(I - Q); B =
1/r*Q*gamma; C = 1/r*H*Q; D = J - H*Q*gamma/2; [numw,denw] =
ss2tf(A,B,C,D,1);

return
```
A.3 Program 3. Acceleration Profile

%data source generation

clear all; close all;

fid = fopen('data1.txt','w')
t = 0:0.033:21; a = 0.51;

T_1 = 1; T_2 = 1; T_3 = 1; T1 = 0:0.033:T_1; T2 = 0:0.033:T_2; T3 = 0:0.033:T_3;

for k = 1:5

for j = 1:length(T1)
    if rem(k+1,2)==0
        fprintf(fid,'%6.3f %12.8f
', t((k-1)*(2*length(T1)+
        length(T2)+length(T3))+j),a);
    else
        fprintf(fid,'%6.3f %12.8f
', t((k-1)*(2*length(T1)+
        length(T2)+length(T3))+j),-a);
    end

end

for j = 1:length(T2)
    fprintf(fid,'%6.3f %12.8f
', t((k-1)*(2*length(T1)+
        length(T2)+length(T3))+j),a);

end

for j = 1:length(T3)
    fprintf(fid,'%6.3f %12.8f
', t((k-1)*(2*length(T1)+
        length(T2)+length(T3))+j),-a);

end
for j=1:length(T1)
    if rem(k+1,2)==0
        fprintf(fid,'%6.3f %12.8f
',t((k-1)*(2*length(T1)+
            length(T2)+length(T3))+j+length(T1)+length(T2)),a);
    else
        fprintf(fid,'%6.3f %12.8f
',t((k-1)*(2*length(T1)+
            length(T2)+length(T3))+j+length(T1)+length(T2)),a);
    end
end

for j=1:length(T3)
    fprintf(fid,'%6.3f %12.8f
',t((k-1)*(2*length(T1)+
        length(T2)+length(T3))+j+2*length(T1)+length(T2)),0);
end
end fclose(fid);
datasource=load('data1.txt'); pm1=datasource(:,2);

t1=datasource(:,1); p1=[t1 pm1];

figure; plot(t1,pm1);
A.4 Program 4. Velocity Profile

%data source generation

clear all; close all;

fid = fopen('data2.txt','w')
t = 0:0.033:21; a = 0.51;
T_1 = 1; T_2 = 1; T_3 = 1; T_1 = 0:0.033:T_1; T_2 = 0:0.033:T_2; T_3 = 0:0.033:T_3;
for k = 1:5

    for j = 1:length(T_1)
        if rem(k+1,2)==0
            fprintf(fid,'%6.3f %12.8f
', t((k-1)*(2*length(T_1)+length(T_2)+
            length(T_3))+j), a*t(j));
        else
            fprintf(fid,'%6.3f %12.8f
', t((k-1)*(2*length(T_1)+length(T_2)+
            length(T_3))+j), -a*t(j));
        end
    end

    for j = 1:length(T_2)
        if rem(k+1,2)==0
            fprintf(fid,'%6.3f %12.8f
', t((k-1)*(2*length(T_1)+length(T_2)+
            length(T_3))+j), a*t(j));
        else
            fprintf(fid,'%6.3f %12.8f
', t((k-1)*(2*length(T_1)+length(T_2)+
            length(T_3))+j), -a*t(j));
        end
    end

end
fprintf(fid,'%6.3f %12.8f
',t((k-1)*(2*length(T1)+length(T2)+
    length(T3))+j+length(T1)),a*T_1);
else
    fprintf(fid,'%6.3f %12.8f
',t((k-1)*(2*length(T1)+length(T2)+
    length(T3))+j+length(T1)),-a*T_1);end
end
for j=1:length(T1)
    if rem(k+1,2)==0
        fprintf(fid,'%6.3f %12.8f
',t((k-1)*(2*length(T1)+length(T2)+
    length(T3))+j+length(T1)+length(T2)),a*T_1-a*t(j));
    else
        fprintf(fid,'%6.3f %12.8f
',t((k-1)*(2*length(T1)+length(T2)+
    length(T3))+j+length(T1)+length(T2)),-a*T_1+a*t(j)); end
end
for j=1:length(T3)
o=j
    fprintf(fid,'%6.3f %12.8f
',t((k-1)*(2*length(T1)+length(T2)+
    length(T3))+j+2*length(T1)+length(T2)),0);
end

datasource=load('data2.txt'); pm1=datasource(:,2);
t1=datasource(:,1); p1=[t1 pm1];

figure; plot(t1,pm1);
A.5 Program 5. Trajectory Profile

% data source generation

clear all;% close all;

fid = fopen('data2.txt','w')
t = 0:0.033:80; a = 0.25;
T_1 = 1; T_2 = 0; T_3 = 0; T1 = 0:0.033:T_1; T2 = 0:0.033:T_2; T3 = 0:0.033:T_3;

x = zeros(length(t),1);

for k = 1:6

for j = 1:length(T1)
    if rem(k+1,2) == 0
        if (k-1)*(2*length(T1)+length(T2)+length(T3))==0
            x((k-1)*(2*length(T1)+length(T2)+length(T3))+j)=x((k-1)*(2*length(T1)+length(T2)+length(T3)))+0.5*a*t(j)^2;
        else
            x((k-1)*(2*length(T1)+length(T2)+length(T3))+j)=x((k-1)*(2*length(T1)+length(T2)+length(T3)))+0.5*a*t(j)^2;
        end
    end

fprintf(fid,'%6.3f%12.8f
',t((k-1)*(2*length(T1)+length(T2)+length(T3))+j),

58
\begin{verbatim}
for j=1:length(T2)
    if rem(k+1,2)==0
        x((k-1)*(2*length(T1)+length(T2)+length(T3))+j)+length(T1))=x((k-1)*
        (2*length(T1)+length(T2)+length(T3))+a*T_1*(t(j));
        fprintf(fid,'%6.3f %12.8f
',t((k-1)*(2*length(T1)+length(T2)+
        length(T3))+j),
        x((k-1)*(2*length(T1)+length(T2)+length(T3))+j));
    else
        x((k-1)*(2*length(T1)+length(T2)+length(T3))+j)=x((k-1)*
        (2*length(T1)+length(T2)+length(T3)))-0.5*a*t(j)^2;
        fprintf(fid,'%6.3f %12.8f
',t((k-1)*(2*length(T1)+length(T2)+
        length(T3))+j),
        x((k-1)*(2*length(T1)+length(T2)+length(T3))+j));
    end
end
\end{verbatim}
\begin{align*}
  j + \text{length}(T1) &= x((k-1) \cdot (2 \cdot \text{length}(T1) + \text{length}(T2) \\
  &+ \text{length}(T3) + \text{length}(T1)) - a \cdot T_1 \cdot (t(j)); \\
  \text{fprintf}(fid, '\%6.3f \%12.8f \n' , t((k-1) \cdot (2 \cdot \text{length}(T1) + \\
  \text{length}(T2) + \text{length}(T3)) + j + \text{length}(T1)), \\
  x((k-1) \cdot (2 \cdot \text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + j + \text{length}(T1))); & \text{end}
\end{align*}

for j=1:length(T1)

  if \text{rem}(k+1,2) == 0
    x((k-1) \cdot (2 \cdot \text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + \\
    j + \text{length}(T1) + \text{length}(T2)) = x((k-1) \cdot \\
    (2 \cdot \text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + \text{length}(T1) + \\
    \text{length}(T2)) - 0.5 \cdot a \cdot (T_1 - t(j))^2 + 0.5 \cdot a \cdot T_1^2; \\
    \text{fprintf}(fid, '\%6.3f \%12.8f \n' , t((k-1) \cdot (2 \cdot \text{length}(T1) + \\
    \text{length}(T2) + \text{length}(T3)) + j + \text{length}(T1) + \text{length}(T2)), \\
    x((k-1) \cdot (2 \cdot \text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + j + \\
    \text{length}(T1) + \text{length}(T2))); \\
  \text{else} \quad x((k-1) \cdot (2 \cdot \text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + j + \\
    \text{length}(T1) + \text{length}(T2)) = x((k-1) \cdot \\
    (2 \cdot \text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + \\
    \text{length}(T1) + \text{length}(T2)) = x((k-1) \cdot \\
    (2 \cdot \text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + \\
    \text{length}(T1) + \text{length}(T2)); & \text{end}
\end{align*}
\[ \text{length}(T1) + \text{length}(T2) + 0.5a(T_1 - t(j))^2 - 0.5aT_1^2; \]

\[
\text{fprintf}(\text{fid}, '\%6.3f\%12.8f\n', t((k-1) +
(2*\text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + j + \text{length}(T1) + \text{length}(T2)),
\text{x((k-1)*(2*\text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + j +
\text{length}(T1) + \text{length}(T2))); \end

\]

\[
\text{for j=1:length(T3)}
\]

\[
x((k-1)*(2*\text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + j + 2*\text{length}(T1) + \text{length}(T2)) = x((k-1) * (2*\text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + 2*\text{length}(T1) + \text{length}(T2));
\]

\[
\text{fprintf}(\text{fid}, '\%6.3f\%12.8f\n', t((k-1) * (2*\text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + j + 2*\text{length}(T1) + \text{length}(T2)), x((k-1) * (2*\text{length}(T1) + \text{length}(T2) + \text{length}(T3)) + j + 2*\text{length}(T1) + \text{length}(T2))); \end
\]

\[
\text{fclose}(\text{fid});
\]
datasource=load('data2.txt'); pm1=datasource(:,2);

t1=datasource(:,1); p=[t1 pm1];

figure; plot(t1,pm1);
A.6 Program 6. State Space Simulation Program 1

%Simulation for the state space algorithm

clear all; close all; a=load('data4.txt');

global u u2;

global a1; T=0.033; t=0:T:20; x0=[0.1 0.1]'; C=[1 0]; u=0; for
i=1:length(t)-1
    clear y; clear t1;
    y1(:,i)=x0;
    u1(i)=u;
    if i>=3
        u2=u1(i-2);
    else u2=u1(i);
    end
    a1=a(i);
    [t1, y]=ode23('sp', [t(i) t(i+1)], x0);
    x0=y(length(t1), :);
end

plot(t(1:length(t)-1),C*y1);

figure; plot(t(1:length(t)-3),u1(3:length(t)-1));
%Simulation for state space algorithms

function xdot=sp(t,x) global u; global u2; global a1;

A=[0 1;0 0]; gamma=280; B1=gamma*[0 0;0 1]; E=[0;1]; B=[0;-1];
Q=eye(2);

P = ARE(A, B1, Q) ;

u=-gamma*B'*P*x;

xdot=A*x+B*u2+E*a1;
LIST OF REFERENCES


