HIGH PERFORMANCE CONTROL THEORY, DESIGN, AND APPLICATIONS

By

CHRISTOPHER S. PEEK

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2008

ACKNOWLEDGMENTS

I thank Professor Denis Gillet, Piyaway Kaekward, and Samuel Depraz of Ecole Polytechnique Federale de Laussanne (EPFL) for their collaboration in the virtual control lab and swarm control work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

HIGH PERFORMANCE CONTROL THEORY, DESIGN, AND APPLICATIONS

By

Christopher S. Peek

May 2008

Chair: Oscar D. Crisalle
Major: Chemical Engineering

Online virtual labs offer potential for students to participate in laboratory practice without access to physical laboratory equipment. Existing online labs show a tradeoff between extensibility for different systems and interactivity with the user. A Virtual Control Lab that overcomes this hurdle has been developed using LabView. A modular structure that allows a developer to simulate a new system is presented to the user by a system of panels and tabs which provides the user a variety of system analysis and real-time simulation data. Classical PID controllers cannot track a ramp change in a set point or disturbance for most systems. In order to achieve offset-free behavior a controller with an additional integral is required. A double integral controller which could be practically implemented as two standard PI controllers connected in series is suggested. Tuning correlations are developed to minimize the integral of the time-weighted absolute error for tracking of a set point ramp. Model Predictive Control is a common control strategy for industrial multivariable systems; however, it suffers from many of the same offset related problems as proportional control. MPC methods are described which use integral states and input-velocity weights to track steps, and in some cases ramps, without offset, while avoiding the complexity of previous tracking methods based on targets and disturbance estimation. A controller is also proposed to use the property of flatness for control of a swarm of wheeled mobile robots. The controller is augmentented with formation tracking and collision avoidance routines.

CHAPTER 1
INTRODUCTION

Controls education is improved by the use of a virtual control laboratory. The laboratory allows access to simulated plants and controllers over a web interface for analysis and simulation of common control problems. The lab makes use of LabVIEW software and the new simulation and control design toolkits for it to offer substantial advantages over existing simulation tools including a good balance between modularity for the development of new simulations using the lab framework and simulation interactivity for the end user. This implementation also provides potential for switching between simulation and remote control of real experimental apparatus. The value of such a laboratory is primarily pedagogical, for teaching principles of process control, but it also has potential as a test rig for new control designs or lab scale apparatus due to its extensible nature.

The tendency of many controllers to drive the system under control to final values that are not exactly those desired is known as offset. The offset-free tracking of ramps in set points can be addressed via a simple double integral controller. While the benefits of multiple integrators are well known, a simple implementation of this idea and tuning guidelines are offered, which can transform the known theory into a practical and effective control scheme. Steady state offset is also considered in the more advanced, multivariable, Model Predictive Control control strategy. Several offset-free methods are proposed which avoid the complexity of implementation and calculation that are characteristic of those already available in the literature.

The problem of controlling a swarm of wheeled mobile robots is addressed by extending a flatness-based control strategy previously developed for a single robot. The control strategy examined provides methods for moving the robots in formation and for avoiding collisions and restoring the formation in the event that the formation is disrupted.

CHAPTER 2
VIRTUAL CONTROL LABORATORY

2.1 Introduction

2.1.1 Background

Engineering educators are increasingly interested in developing tools that allow students to acquire hands-on laboratory experience without necessarily providing physical access to a building that houses specific experimental equipment [1][2]. These tools often take the form of software environments commonly referred to as virtual or remote laboratories [1][3][4][5]. The software either allows a user to interact with an experimental setup located in another geographical location (i.e., a remote laboratory) or uses numerical simulation tools to emulate the behavior of experimental system (a virtual laboratory). The appeal of virtual and remote laboratory tools is largely due to the increasing demand for active learning and flexible education, and for the appeal of implementing techniques of learning via discovery.

Active learning seeks to provide students with opportunities to better integrate and reinforce knowledge presented in the classroom, as well as to acquire the practical know-how that is such an essential component in engineering education. Problem-based learning and hands-on laboratory activities are good examples of active-learning vehicles. In traditional educational settings, an effective active-learning environment calls for intense levels of interaction with experimental resources, and requires the coordination of the efforts and schedules of multiple parties, including numerous students and pedagogical support staff. As a consequence, the effort may be challenged by significant obstacles stemming from logistical, organizational, and cost concerns. On the other hand, such constraints can be overcome by making available flexible learning resources that meet the students' individual needs and schedules. In particular, virtual and remote laboratories offer substantial flexible education benefits since they can be made accessible to the students at any time and from any location, features that cannot be easily matched

by traditional learning environments [6]. Furthermore, virtual and remote laboratory resources can be combined with other flexible-education tools, such as lectures offered via digital streaming-video, to maximize the degree of flexibility of the learning environment provided to the learners. In addition, another substantial advantage of virtual and remote control laboratory tools is that they can be used to supplement traditional teaching methodologies. For example, they can be used during a traditional lecture to show the students how to apply concepts presented in class to a simulated or remote experimental system. This approach imbues a classical lecture with an active and flexible learning component, thus strengthening the pedagogical value of the lecture.

Another benefit of virtual and remote laboratories in education is that they promote discovery learning. In this approach students are given access to the laboratory with minimal instructions, and are allowed to explore the systems for themselves [7]. A virtual laboratory or a well-designed remote laboratory can offer students a chance to explore safely and easily the behavior of a system that may be physically inaccessible.

2.1.2 Problem Statement

Given the significant appeal of the engineering-education benefits realized by virtual and remote laboratories, it is not surprising that several systems are already available on the World Wide Web (cf. [3] [8][9][10] and [11], among others). An early vision for a virtual control laboratory architecture can be found in [12]. The available systems address the flexible hands-on learning paradigm at different functionality levels, and with different degrees of student interactivity. Some virtual laboratories, especially those developed using commercial simulation tools, can be easily customized to suit the user needs, but tend to show limited potential for interactivity. For example, it is quite common that a simulation run has to be stopped for the purpose of changing controller settings or process parameters. Software plug-ins or expensive specialized third-party software are also often required to run the simulations. Another class of Web-based laboratories attain a substantial degree of interactivity through the use of proprietary

software written by the laboratory developers in a formal programming language such as Java. As a consequence, these tools require knowledge of the specific language adopted, and attempts to customize the environment to address specific needs of the learner may involve substantial modifications of source code. Therefore, these laboratories are typically only developed for a specific and restricted set of simulated or physical systems. Another concern when using proprietary code is that instances of the same laboratory software developed for different physical systems often have substantially different interfaces, requiring a user to get re-familiarized with the tools made available for each task. In summary, the typical Web-based laboratories that are currently available seek to strike a reasonable trade off between user interactivity and usage versatility.

The objective of this paper is to define a general architecture for a Virtual Control Laboratory (VCL) and to demonstrate a realization of that architecture in terms of a concrete example. The VCL is a software environment specialized for control engineering education, capable of avoiding the trades-off present in existing Web-based laboratories, and offering a single framework that can be used to easily develop Web-based, interactive virtual and remote laboratories for the deployment in a wide variety of systems and scenarios. The tool should be able to provide educational benefits in both flexible and traditional education, beyond those attained by existing software controls laboratories. It should also facilitate the developer's task of generating new laboratory experiments through a modular design.

2.2 Architectural Requirements for a Virtual Control Laboratory

2.2.1 General Principles

The relative shortcomings of virtual control laboratories that have been proposed to date can be overcome by conceiving a VCL architecture that meets three general principles: interactive Web-access capability, modularity of design, and an intuitive and user-friendly interface.

We argue that a VCL tool should be capable of deployment in the World-Wide-Web environment using standard Web browsers. The alternative of using specialized client software renders the VCL less flexible, since the user cannot access the resources using only standard software available in most computers. Hence, Web access is the key ingredient in imbuing the VCL with flexible learning features, making it possible to accommodate each student's personal schedule. Therefore, the software development environment should support easy publication of the VCL on the Web. Furthermore, the Web VCL tool should be highly interactive, allowing the student to change selected parameters and immediately observe the effect of the changes. Such interactive capability is crucial for the success of the VCL because the tool seeks to emulate experimentation activities that would traditionally be conducted in a physical laboratory . Another crucial requirement for the VCL envisioned is that the source code should be highly modular [13]. An instructor should be able to use the VCL framework to quickly and easily develop a simulation and analysis environment for any plant or controller that is of interest. This modularity is realized by developing a framework where the interface used to set up the analysis and simulation for one physical system can be easily modified to set up the analysis and simulation for a different system.

A third requirement for the VCL is that is should have a user-friendly interface. The student must find it easy to learn how to use the Web-base laboratory if the tool is expected to have substantial pedagogical value [14]. An excessive number of instructions can interfere with the discovery-learning process. It is therefore essential that the interface be intuitive. It may be desirable to make the interface resemble that of other software and hardware interfaces that are already familiar to the user, to simplify and accelerate the learning process. Another principle regarding the intuitiveness of the interface is that the same information should be deliberately presented in several different formats, for instance via graphical animation, through displays on slider bars, and also in a chart. Consistent

use of color is recommended to help the user quickly identify different representations of the same data.

2.2.2 Specific Architectural Requirements

Specific architectural requirements that are of central concern in the design of a VCL include the following: (1) the ability to define and modify the plant and the controller, (2) easy transfer from open-loop to closed-loop operation by switching the controller from manual to automatic mode, (3) the inclusion of advanced analysis tools, (4) the ability to support control synthesis tasks, (5) the ability to produce open-loop and closed-loop simulation of time-domain responses, (6) the inclusion of versatile signal generators to produce standard input signals, (7) the inclusion of animation to project a visual perception of the evolution of the plant, (8) the ability to incorporate the dynamics of sensors and actuators, as needed, and (9) embedded documentation and help-guides, and (10) reporting capabilities.

As a tool specialized for control engineering, the VCL must include two key components, namely a physical system of interest (the plant), and a controller. Examples of plants typically considered for pedagogical purposes in standard textbooks include a DC motor, an inverted pendulum, a helicopter, and a chemical reactor, among others. The plant model may contain significant nonlinear dynamics. On the other hand, unless the VCL is specifically designed to illustrate nonlinear control theory, from an educational perspective it is particularly important to include a linearized version of the plant model to enable the students to develop experience with the systematic and well-understood tools of linear control analysis and design. In addition, the VCL should allow the user to change the value of selected physical parameters of the plant, such as the inertia of a DC motor or the heat-transfer area in a chemical reactor, thus allowing for the investigation of the different dynamic-response regimes.

Examples of controllers include the ubiquitous proportional-integral-derivative (PID) control scheme, lead-lag control, on-off control, cascade control, and optimal control,

among others [15] [16] [17] [18][19] [20] [21] [22]. The user should be able to change key parameters of the controller. For example, to adjust the gain of a PID controller for the purpose of carrying out tuning experiments. When pedagogically suitable, the user should be able to easily select from a list of relevant control schemes. For example, the user may choose to select a single PID controller and proceed to evaluate its performance, and then select a dual-PID cascade scheme for evaluation and subsequent comparison with the single-PID choice. In the case of multiloop control schemes, the VCL should easily allow the user to modify the selection of appropriate input-output pairings. The VCL should make available to the user concise and factual information about each control scheme available.

Another required feature of a VCL is that it should enable the user to set the controller in manual mode, allowing the user to test the response of the plant to diagnostic input signals, such as a step forcing or a sinusoidal excitation. This situation is known as open-loop control because the output of the process is effectively disconnected from the controller. Alternatively, the user may select a specific control structure for deployment, in which case the plant is manipulated in automatic mode by the controller, defining a closed-loop control configuration. A key requirement is that the VCL should clearly and unambiguously show to the user that the system is in either open-loop or closed-loop mode. This can be achieved by displaying a message in all the relevant locations where it is important to distinguish between the two modes. Another solution, which is attractive from a pedagogical viewpoint, would be to present the user with a specific graphical rendition of the relationship between the manual controller and the plant under open-loop operation, clearly showing that the output of the process is no longer available to the controller. The VCL would then switch to a different graphical representation during closed-loop operation, perhaps showing a signal-flow diagram that clearly connects the outputs of the plant to the automatic controller and the outputs of the controller to the input ports of the plant.

An important concept taught in control engineering is the process of analysis of both the open-loop and the closed-loop system. Under open-loop control the analysis studies consist of revealing all the features of the plant that are of dynamic relevance. For linear systems the analysis task often involves the determination of the plant poles and zeros, the characterization of the frequency response of the plant through magnitude and phase diagrams in the form of Bode or Nichols plots, as well as a characterization of the time-domain response of the plant to step or sinusoidal input signals implemented via the manual-mode controller. The VCL should update the poles, zeros, Bode plots, and the other analysis immediately after the user changes the values of the plant parameters. The open-loop analysis tools included in the VCL should allow the user to classify the plant as stable or unstable, to assess its level of damping, and to determine the value of its dominant time constants and its bandwidth, etc. Under closed-loop control the analysis task also consists of producing a map of poles and zeros, generating Bode or Nichols plots, and tracing time-domain response curves; however, in this case the results describe the entire loop created between the plant and the automatic controller rather than to only the plant. For example, a classical closed-loop analysis task is the determination of whether the controller succeeds in creating a loop that be is stable, an issue that can be assessed by inspecting the location of the closed-loop poles on the complex plane. The VCL must be able to update the map of the closed-loop poles immediately after the user makes changes to the parameters of the controller or after a new controller structure is selected. Obviously, other analytical results should also be updated after such changes occur in the controller or plant parameters.

Another key concept taught to control engineering students is the process of control synthesis, which typically consists of specifying an appropriate control structure, for instance a PID or a lead-lag scheme, as well as values of the parameters included in the structure, such as controller gains and tuning time-constants. Such specification must meet closed-loop requirements, such as location of closed-loop poles, degree of overshoot,

settling time of the time-domain response, etc. A requirement of the VCL architecture is the ability to quickly display the effect of any changes made to the controller and to quantify the resulting closed-loop performance, since this allows the student user to carry out synthesis work. Since the assessment of the controller performance in closed-loop mode is in fact an analysis task, the VCL should be able to deliver a desirable control synthesis environment whenever its analysis component meets the architectural specifications previously discussed. It is of critical importance that the VCL enable the student to carry out open-loop and closed-loop simulations of time-domain responses. Numerical simulation studies under open-loop mode permit the characterization of the plant dynamics through the observation of plots that document the response produced by standard input signals, such as step and sinusoidal excitations. In addition, the closed-loop simulation capabilities allow the student to observe the time-domain effect of analytical features, such as the effect of the closed-loop poles on the plant response, as well as to assess the time-domain performance of alternative control design choices. The VCL should therefore include software resources that solve differential equations using robust numerical techniques, including the ability to cope with situations where the equations are stiff.

The VCL must include a number of signal generators that can inject appropriate types of inputs at the user's discretion. These elements should be able to produce standard signal patterns, including constant, step, sinusoidal, ramp, and pulse signals of different amplitudes and frequencies. Making available a large palette of options increases the versatility of the VCL, given that certain problems require the use of less common signals, such as saw-tooth patterns, and other specialized input waveforms. In particular, the signal generators are used to define set-point signals that are fed to the controller under closed-loop control operation, to produce a user-specified manual signal emanating from the controller under open-loop operation, or to inject a process disturbance signal as needed to test a controller's ability to perform as a regulator.

A desirable feature in a VCL is the ability to present to the user the evolution of the plant variables via two-dimensional or three-dimensional graphical animation. For example, the motion of a robotic arm may be shown in a graphical form that gives the user the impression that he/she is observing the movement as it would occur in three dimensions, much as if the motion were presented to the observer through a video-camera image. As another example, the motion of the indication needle in a pressure gauge can be shown to reflect changes in gas pressure inside a chemical reactor. Although the essential elements of the control loop are the plant and the controller, the VCL should have enough flexibility to include other components that are sometimes the focus of significant pedagogical interest, such as the case of sensors, power converters, and actuators.

The VCL should include documentation that presents to the user a fundamental description of the plant and of the controller, including the dynamic model utilized for analysis, along with a clear description of the most important physical parameters. Even when the interface is highly intuitive, explicit instructions are necessary to ensure ease of use. This is especially true since the system being simulated or the controller type used may not be familiar to beginner learners who stand to gain the most educational value from the VCL tool. In order to accomplish this, help documents should be embedded directly in the virtual laboratory interface, making them conspicuous and easy to access. The documentation should be complete, though concise, and should be presented from a perspective that is useful for control engineering design and analysis. Inclusion in the VCL of all relevant plant and controller documentation as well as a users guide contributes to making the environment self-contained, and hence more effective from the pedagogical point of view.

2.3 Realization of a Virtual Control Laboratory

Taking into consideration the architectural principles and requirements given in Section 2.2, the authors have developed a VCL that can be accessed by students using a Web browser. Figure 2-1 shows a view of the proposed VCL inside a standard

Web-browser window, as it is presented to a student. In this case the plant studied is a DC motor that is shown as an animated graphics on the upper-left area the VCL interface window. On the lower-left rectangle of the window the user can define different physical parameters for the DC motor, and can also select a controller structure as well as specify all values of the controller parameters. Finally, the right-half of the screen shows five tabbed windows that contain various types of information, including plant and controller documentation, as well as the results of analysis and simulation studies.

2.3.1 Software Platform for Development

The example shown in Figure 2-1 is a realization of the Virtual Control Lab concept created using National Instruments' LabVIEW software [24]. This software offers many powerful resources for developing and maintaining a VCL, including ease of creating standard interface controls, and built-in Web publishing capability. In addition, the software provides a suite of resources useful for control-engineering education, such as the LabVIEW Simulation Module and the LabVIEW Control Design Toolkit, which automate many of the analysis and simulation tasks carried out by the VCL. These advantages of LabVIEW, combined with the option of programming using a graphical language, make developing, maintaining, and modifying the VCL a relatively simple and straightforward task. Development and implementation alternatives for concepts analogous to the VLC prototype proposed here are also possible using other programming languages, as described in [23]; however, in that approach advanced analysis tasks typically require calls to external software packages.

2.3.2 Interface Design Based on Three Specialized Panels

To provide the user with an organized and intuitive access, an interface consisting of the following three distinct areas is proposed: an Animation Panel, an Interaction Panel, and a Navigation Panel. Figure 2-1 shows an example of the interface, featuring the three-panel structure. The Animation Panel (located on the upper left region of the interface window) and the Interaction Panel (located on the lower left region) are

permanently displayed, allowing the observation of key features of the system and the controller on a continuous basis, emulating a perspective that the user would have when running the equivalent experiment in a physical laboratory. The Navigation Panel, the larger rectangular area located on the right of the interface, contains five tabs that activate windows that display different types of information.

Whenever possible, standard dialog buttons, input boxes, and tab panels that resemble those found in common home-computer software are used. Thus, users have to learn only the specifics of the laboratory experiment itself, avoiding the confusion that may be introduced by esoteric interface controls. The intuitive nature of the VCL interface is also enhanced by assigning each input and output a color, and whenever possible, labeling interface elements related to a given variable with the appropriate color.

2.3.3 Animation and Interaction Panels

The Animation and Interaction Panels are closely integrated with each other, and it is therefore convenient to discuss them jointly. The purpose of the Animation Panel is to give the user a visual representation of the plant, along with indicators that display two-dimensional or three-dimensional motion. The purpose of the Interaction Panel is to allow the user to define plant parameters, and to adjust control parameters as needed. Choices made by the user in the Interaction Panel (such as a change in the controller state from Manual to Auto) trigger a change in what is displayed in the Animation Panel. Hence, the two panels are highly coupled. Figure 2-2 shows an implementation of the Animation and Interaction Panels of the DC-Motor VCL.

The Interaction Panel in Figure 2-2a shows a situation where the controller is set in Manual mode (see the area labeled "Controller Parameters"). Hence, the system is in an open-loop configuration. The Animation Panel of Figure 2-2a shows a controller liked with an electrical wire to the input of a three-dimensional rendition of the DC motor. In turn, the motor has an electrical connection to two sensors, namely velocity and position indicators. The DC motor is represented using a geometrical perspective

consisting of a stationary cylindrical body (the stator) connected to a rotating disk (the rotor). An electrical wire connects the motor to the angular velocity indicator represented by a speedometer, and also to a round dial that displays the angular position. These two output indicators reflect the fact that in this DC motor control problem either the angular position θ or the angular velocity ω may serve as the process variable (PV), namely as the measured variable that is eventually to be maintained at a set point by the controller. Since the controller is set to Manual mode in the Interaction Panel, the system is configured in open-loop mode and consequently the Animation Panel shows that the input wire representing the process variable (PV) is disconnected from the indicators. When in Manual mode, the student user can change at will the output of the controller indicated in the panel as the manipulated variable (MV), using either the bias box in the Interaction Panel or using a signal generator located in the Navigation Panel that is discussed later in this section.

Figure 2-2b depicts a situation where the user has used the Interaction Panel to switch the controller to Auto mode. In this case the Interaction Panel shows that the student has specified the angular position θ as the chosen process variable (PV), as indicated in the box labeled "MV to PV pairing". The system is now in a closed-loop configuration where the angular position is the process variable used for feedback control. The structure of the loop is made obvious to the user in the Animation Panel, where the controller input wire labeled PV is connected to the angular position indicator. The wiring scheme defines a closed loop, hence serving as a visual aid to enhance the student's awareness of the fact that the controller is in the Auto mode of operation. Note the appearance of a white needle on the angular position indicator, representing the controller set point. Likewise, a white triangular marker is visible on the side of the vertical slide bar that represents the angular position, also indicating the value of the set point of the controller. Either the white needle or the white triangular marker can be moved to different positions using a computer mouse, thus introducing a set-point change. Set

point changes can also be specified using a signal generator located in the Simulation window of the Navigation panel, as discussed later. The Interaction Panel shows the values of the proportional-integral-derivative (PID) controller, including a gain $K_c = 1$, an integral time constant $\tau_i = 0.8$, and a derivative time constant $\tau_d = 1$. Immediately to the left of the boxes that specify the values of the integral and derivative time constants, the user finds square buttons drawn with a relief perspective, which can be pressed via a computer mouse click to activate the integral and derivative terms. The figure shows that the integral-action term is turned OFF (which has the effect of ignoring the numerically-specified value in the τ_i box), and it also shows that the derivative-action term is turned ON. In keeping with the objective of maximizing interactivity, the parameters shown in the Interaction Panel may be changed at any time, even while a simulation is running. Figure 2-2c shows an alternative automatic-control configuration specified by the user via the Interaction Panel. In this case the selected process variable is the angular velocity ω, as clearly shown in the Animation Panel where the speedometer is wired to the controller input port. Note that in this case the white needle now appears on the speedometer, identifying the value of the set point that the user has defined for the angular velocity. Likewise, a white triangular set-point marker appears on the vertical slider bar displays the value of the angular velocity. Hence, the Animation Panel is updated in an immediate fashion to reflect the configuration defined in the Interaction panel.

2.3.4 Navigation Panel and its Tabbed Windows

The Navigation Panel allows the user to access a variety of pedagogical resources through a series of five of tabbed windows bearing the following titles: Information, Plant, Controller, Analysis, and Simulation.

First, when the user selects the Information tab, a window opens up in the Navigation Panel displaying general information on how to utilize the VCL environment. This window serves as a users guide that is embedded in the VCL, that gives the student access to

information without leaving the interface. The text displayed can be scrolled down to reveal the entire contents of the document. The Information window succinctly describes the key features of the Animation, Interaction, and Navigation panels. In addition, the window defines for the student a number of control-engineering experiments that can be carried out with the VCL, clearly stating the objective of each experiment and suggesting a sequence of steps that could be followed to accomplish the stated goals. For example, one experiment has the objective of characterizing the open-loop step response of the DC motor. The text suggests to the user to proceed by putting the controller in Manual mode, exciting the plant with a step change in the manipulated variable, and then identifying key features of the response of the process variable observed in the Simulation window of the Navigation Panel.

Second, selection of the Plant tab brings up a window that displays a file containing information on the dynamics of the DC motor. The Navigation Panel of Figure 2-1 shows that the Plant window presents to the user two transfer functions that describe the relationship between the applied driving potential u (the manipulated variable) and each of the two variables that can be of interest, namely the angular position θ and angular velocity ω (the process variables). Using the scroll bar located on the side of the window the user can display the remainder of the text, which describes how the transfer-functions are derived from the mechanical and electrical differential equations that define the dynamics of the motor, and also shows how the modeling equations can be written in a standard state-space form.

Third, the Controller tab activates a window that displays a file with information about the different controllers that are available to the user. In the case of the DC-Motor VCL, the Controller window reviews the key principles of PID control theory, as shown in Figure 2-3. The document describes the differences between the Manual and Auto modes of the controller, introduces the classical transfer function of an ideal PID controller, describes a practical realizable version of the PID law, and discusses the effect of the

different tuning parameters of the controller. The window also shows the numerical coefficients of the numerator and denominator polynomials that define the transfer function of the controller (see the frame entitled "Current Controller"). These coefficients are immediately updated when the controller parameters are modified by the user in the Interaction Panel. In addition, the Controller window shows a map of the location of the poles and zeros of the controller transfer function.

Fourth, the Analysis tab displays a window that contains the classical analysis tools supported by control-engineering theory. As shown in Figure 2-4, this window contains a transfer function showing numerical values for all the coefficients of the numerator and denominator polynomials, a plot the time-domain response to a unit-step input excitation, a map showing the location of poles and zeros on the complex plane, and a Bode plot that provides frequency response information. When the controller is in Manual mode, the information displayed in the Analysis window refers only to the plant. In that case the system is in open-loop mode, and the DC motor is devoid of a feedback connection to the controller. Hence, the transfer function, step response curve, pole-zero map, and the Bode plot refer to the dynamics of the DC motor. On the other hand, when the controller is set to Auto mode, the information displayed in the Analysis window refers to the closed-loop transfer function relating the set point of the controller to the process variable (either θ or ω, as defined by the user in the Interaction Panel). Hence, in this case the transfer function, step-response curve, pole-zero map, and the Bode plot describe the dynamics of the closed-loop established between the DC motor and the PID controller. Hence, the information shown in the Analysis window is synchronized with the user's choice of Manual or Auto control modes in the Interaction Panel. In that sense, the Analysis window is a contextual window, showing different analysis results depending on whether the system is in open-loop or closed-loop. As shown in Figure 2-4, the Analysis window clearly displays a message indicating whether the system is in open-loop or closed-loop mode, hence ensuring that the user correctly interprets the contextual analytical results

presented. The maximum and minimum values for the horizontal and vertical axes of any graph in the window can be easily adjusted by simply clicking on the corresponding tick mark and typing a new value. This feature gives the user great flexibility in adjusting the scale of graphs as needed to reveal features of interest.

Finally, the Simulation tab of the DC-Motor VCL brings to the front of the Navigation Panel a window with a graph showing the results of a numerical simulation that describes the response of the angular position θ and angular velocity ω to a user-selected input. This is also a contextual window. More specifically, when the controller is in Manual mode, the graph describes the open-loop responses of the plant to a user-specified driving-voltage input u, in the absence of feedback control. In this case the user can change the driving voltage signal by selecting an excitation from the Signal Generator appearing in the Simulation window (see bottom half of Figure reffig:sim). The signal options available in the DC-Motor VCL include a constant or sinusoidal signal, as well as a saw-tooth and a pulsed waveform. An alternative scenario is when the controller is switched to Auto mode and the system is therefore configured in closed-loop mode. In this case the user can change the set point of the controller by selecting an excitation waveform from the signal generator. In contrast to the previous case, the values of the driving voltage are now dictated by the PID control law, reflecting the fact that the controller feedback path is active during closed-loop operation. The numerical response of the plant or of the closed loop to the selected excitation is computed using a fourth-order Runge-Kutta integration algorithm. All the signals of relevance are displayed in the graph using lines of different line types and colors. The maximum value of the time axis can be easily changed by selecting the corresponding tick mark and typing a new value. In addition, the slider bar located below the graph allows the user to display previous traces that are no longer visible in current plot scope (cf. Figure 5). The window also has a button labeled "Run" that starts the numerical simulation task when it is depressed. The button label then turns to "Stop", and serves to terminate the calculations whenever it is

pressed. A button entitled "Clear Chart" allows the user to erase all the traces. Finally, a button entitled "Export" allows the user to create an ASCII file that contains all the results of the simulation calculations, as well as a summary of the relevant parameters that describe the plant and the controller. As in the case of the Analysis window, a message is clearly displayed indicating whether the system is in open-loop or closed-loop mode, hence ensuring that the user can correctly interpret the simulation results presented. The simulation window allows the user to make changes in the Interaction Panel while a numerical simulation is taking place. All changes are immediately accepted by the simulation resources, without requiring the user to stop a run in progress.

The authors have found that the use of a tabbed-window format for the Navigation Panel provides an intuitive method for organizing different aspects of the VCL. On the other hand, the use of an excessive number of windows appears to be counterproductive, given that the user can be confused or overwhelmed by a large number of options. In our opinion, restricting the number of tabs to only five appears to be a compromise that successfully resolves the competing need for a simple, elegant interface, and the need for displaying large amounts of relevant analysis and simulation results as well as documentary information.

2.3.5 Publication and Deployment on the World Wide Web

LabVIEW has built-in Web publishing capabilities that in principle make the software developments Web-accessible via a simple command. This creates one hypertext instance that can be accessed by one student at the time using a Web browser. On the other hand, in a teaching environment it is necessary to permit multiple users to have full access to the VCL at the same time. This can be easily accomplished by creating a common gateway interface (CGI) front end that upon request produces replicates of the VCL and runs a copy for each connected user. To minimize the use of storage resources, the CGI must also remove any VCL copies that are no longer in use. An effective CGI can be created using LabVIEW, which is the alternative selected by the authors, or could be written

using traditional languages such as Perl. An extensive discussion of technical alternatives is presented in [24].

In order to make available on a Web browser window all the interactive features of DC-Motor VCL created by the authors, it is necessary to install the LabVIEW Run-Time Engine in the remote computer. This engine is available as a free download that in turn installs an appropriate browser plug-in that is of critical importance to ensure an interactive interface. Fortunately, when a student attempts to use the VCL for the first time a dialog message appears if the Run-Time Engine plug-in is not installed, and the user is prompted to download and install the required executable file. On the other hand, the need for a plug-in solution, common to many interactive virtual or remote laboratories, reduces the flexibility of the learning environment because the students are constrained to utilizing computers where they have the appropriate permission to install a browser plug-in. The Web-based DC-Motor VCL is designed to support a high level of interactivity with the student user. The user can change controller and plant parameters, select alternative process variables for control purposes, establish a open-loop or closed-loop mode of operation, change the ranges of all the horizontal and vertical axis on the analysis plots, launch simulation studies, and change set points. In this fashion, the proposed tool satisfies a key structural requirement, namely, the attainment of a highly interactive interface via a Web-browser window.

2.3.6 Modularity Other Software Design Details

LabVIEW routines and their interfaces are stored in files referred to as virtual instruments (VIs). The actual functions that define the plant and controller are stored in sub-virtual-instruments (subVIs) in LabVIEW. The DC-Motor VCL is conceived in a highly modular fashion designed to enable an instructor to reuse the software to design another VCL. The instructor would proceed to first change the icons in the Animation Panel to reflect the physics of the new plant. Second, a state-space realization of the transfer function describing the relationship between the input variable and the

manipulated variable must be coded into the hidden LabVIEW wiring diagram associated with the VI. Third, the input boxes used to define plant parameters in the Interaction Panel VI are wired to their corresponding entries in the transfer function, so that any parameter changes are immediately reflected in the simulation. Fourth, updated files are embedded in the Information, Plant and Controller windows of the Navigation Panel VI, respectively describing the new VCL experiment and its plant. A subVI is used to define information common to multiple VIs, such as variable names, for example, so that the framework can automatically update all the interface elements in the Navigation Panel to reflect the current system without any effort on the part of the developer. The developer would change the minimum and maximum values of the percentage scale used in the graph of the Animation Panel. Other smaller tasks may be involved in the process, such as adding or eliminating new plant-parameter boxes in the Interaction Panel, and rearranging the buttons, boxes, and other icons to produce an aesthetically pleasing interface. Finally, the resulting VCL can be published for Web access. In this fashion, the resulting modified VCL and its original counterpart will have a common interface, each one customized in terms of animation and labels that reflect the physics of their respective plants.

The development takes advantage of the NI LabVIEW Simulation Module to calculate closed-loop time-domain responses. It is straightforward to configure the simulation to allow the user to implement changes to the plant or to the controller as a simulation is in progress, hence avoiding the need to restart a run whenever control or process changes are made. This behavior is obtained by conducting the simulation over a series of successive finite periods of durations larger than the integration step size used by the simulator. Data decimation used to report the results at pre-specified intervals, and an update of the plant and controller parameters is done at the end of each finite period so that user-defined changes can be quickly recognized by the simulator. We have found that a Runge-Kutta algorithm of order four with a fixed step size [25] is an adequate solver for the differential equations that describe the dynamics of the DC motor.

2.4 Pedagogical Scenarios

The VCL can be used in a number of pedagogical scenarios conceived to take advantage of the benefits of active learning, flexible learning, and learning via discovery.

A first scenario of interest is the use of the VCL during a traditional lecture to illustrate a particular control-engineering concept. For example, the instructor could use the VCL closed-loop capabilities to show the students how the offset characteristic of proportional-only controllers can be reduced by choosing larger absolute values for the controller gain. The instructor could then show how at the point of ultimate-gain the benefits of reduced offset begin to erode due to the onset of an undesirable persistently oscillatory response. The students would then observe animated evidence of how even larger values of the controller gain lead to instability. Other examples of concepts that can be illustrated is the effect of time constants on the plant time-domain response and on the frequency-domain bandwidth, the relationship between pole locations and stability, and the effect of open-loop zeros on the time-domain response of the plant, among many others.

The instructor can choose to present the VCL demonstration either before or after the control engineering concept in question is treated in the lecture. Presenting the VCL demonstration before the lecture discussion gives the students a strong motivational incentive to develop a curiosity for the problem, and helps them to identify the key technical problems that need to be addressed by the theory. The ensuing lecture discussion of the topic, presented on the blackboard or via other traditional delivery methods, such using viewgraph support, is likely to be more successful in keeping the students focused on the problem because they have already acquired relevant experience on the problem through the VCL. Alternatively, the instructor may choose to conduct a VCL demonstration after the topic has been presented on the blackboard. In this case the VCL provides an opportunity for reinforcing the knowledge gained, providing visual experience in the form of two-dimensional or three-dimensional animation and simulated

signal graphs, that help the students make cognitive links between associated concepts that often appear more abstract in nature when presented on the blackboard.

A second scenario of interest is the use of the VCL in the context of informal cooperative learning exercises in the classroom [26]. Using well-established techniques the instructor directs the students to organize themselves in small working groups, and defines the goals of a pre-planned exercise that needs to be solved cooperatively by all the students in the group [26] with the assistance of the Web-accessible VCL. Examples of exercise activities include tuning controllers, identifying the order of a plant from an open-loop step response, finding the ultimate gain of a proportional-only controller, etc. These activities fall under the categories of discovery and active learning.

A third scenario is the use of the VCL to support formal cooperative exercises [26], also known as group projects. In this case the students are given a long-term assignment that involves several intermediate steps, culminating with a comprehensive final report. In this case the VCL would be used to provide support for the completion of the intermediate steps, such as modeling the plant, characterizing its frequency-domain properties, comparing and contrasting the frequency-domain and time-domain performance of alternative control schemes, and validating the stability of proposed control design in terms of closed-loop pole maps. These activities also fall under the category of discovery and active learning.

A fourth scenario is the use of the VCL as a support tool for homework assignments. The flexible nature of the VCL and the comprehensive nature of its analysis and simulation resources can effectively support the instructor's plan for engaging the students in active learning while completing homework assignments. All aspects of modeling, analysis, design, and simulation can be addressed with the aid of the VCL as a supplemental learning resource. In this case, the VCL is used to support active learning.

Finally, from a pedagogical viewpoint the instructor may find it advantageous to expose students to new concepts in a sequential fashion. The modular structure of the

VCL framework allows the instructor to enable only selected parts of the software, preventing the students from experimenting prematurely with advanced features. For example, students could be presented with a VCL version where the controller is locked in Manual mode, and where the Simulation window is enabled. The student are then asked to excite the plant with diagnostic input signals, such as steps and sine waveforms, and use the resulting response to produce and validate a plant model. Once this task is completed, the instructor can release a version of the VCL identical to the latter except that the updated version allows the controller to be switched into Auto mode. The new exercise assigned could consist of tuning the controller by selecting appropriate parameters and assessing resulting the closed-loop response using the Simulation window.

2.5 Conclusions

The key elements required in an architectural description of a VCL have been identified as a conceptual guide for developing tools that successfully meet the needs of flexible and active learning. An example VCL specialized for the control of a DC motor meets most of the architectural requirements, realizing a high level of user interactivity while remaining a versatile modular platform that can be modified to accommodate other plants.

Although the DC-Motor VCL represents an effective and elegant realization of the proposed VCL paradigm, it does not necessarily satisfy all the desirable architectural features. One conceivable shortcoming is the need to install an appropriate browser plug-in their computers, which reduces the tool's ability to provide service to all Web-enabled client computers. This, however, is a common limitation of interactive Web-based systems, and the plug-in alternative selected in the DC-Motor VCL is a good compromise within the context of the currently available technology.

In contrast to the case of other virtual laboratories created using specialized programming languages such as Java, the DC-Motor VCL presents to the student a wide array of analysis and simulation options without making external calls to other

supporting pieces of software. In that sense, the VCL is a self-contained tool because it does not require that neither the student's machine nor the Web-server computer maintain external control-engineering support software. All the numerical simulation computations, analysis calculations, interface management, and Web-interactivity support tasks are carried out by a single software tool, namely LabVIEW.

Finally, even though developed on a proprietary software platform, the proposed DC-Motor VCL offers a very high level of user-interactivity while preserving the ability to allow the instructor to customize the base-case design to suit other physical plants. The modular structure of the VCL allows a developer to create new simulations that take advantage of the features and interface of the framework proposed. It is possible to model a wide range of plants without sacrificing any of the interactive features of the virtual control lab. The reusability of the DC-Motor VCL described here is fairly straightforward when the new plant of interest has only one manipulated variable and either one or two process variables. An extension to the case of multiple-input/multiple-output (MIMO) plants using an analogous procedure is easily envisioned provided that the instructor has available a base-case MIMO VCL that complies with the architectural constraints described in Section 2.2.

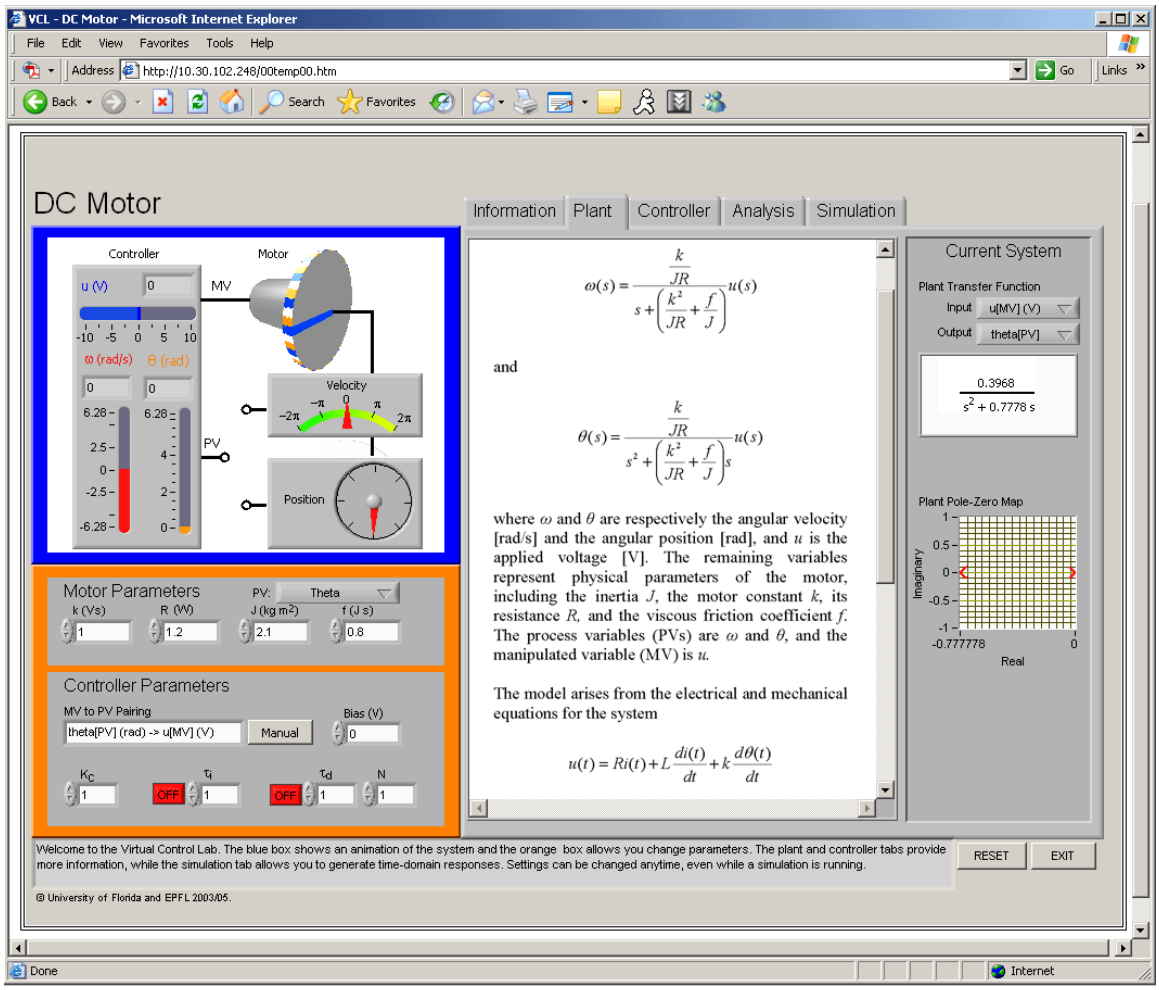

Figure 2-1. Realization of a DC Motor Virtual Control Lab accessed by students using a standard Web browser. The interface presents three distinct panels: (1) an Animation Panel (located on the upper left area) showing details of the DC motor under closed-loop control, (2) an Interaction Panel (lower left) that allows the user to define the parameters of the plant and of the controller, and (3) a Navigation Panel (right-half area) consisting of a series of five tabbed windows that contain different types of information as suggested by the tab labels.

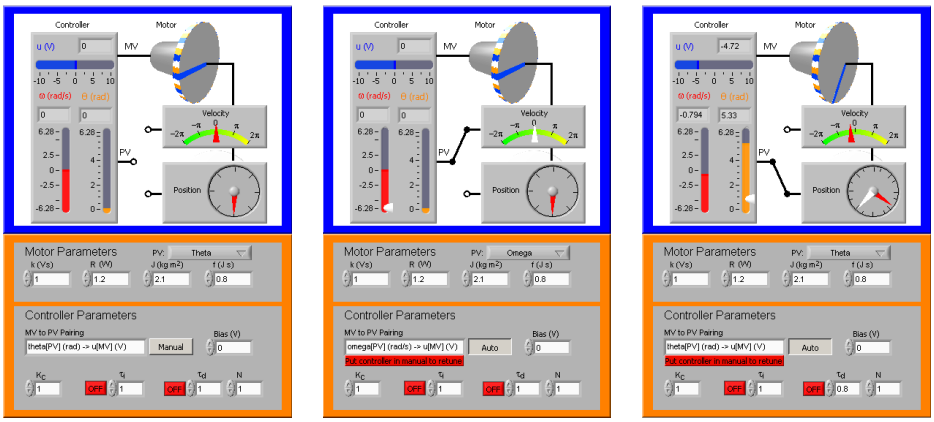

Figure 2-2. Animation panel and interaction panel showing (a) the controller set in Manual mode, (b) the controller set in Auto mode using the angular position as the process variable used as feedback, (c) the controller set in Auto mode using the angular velocity as the process variable used as feedback.

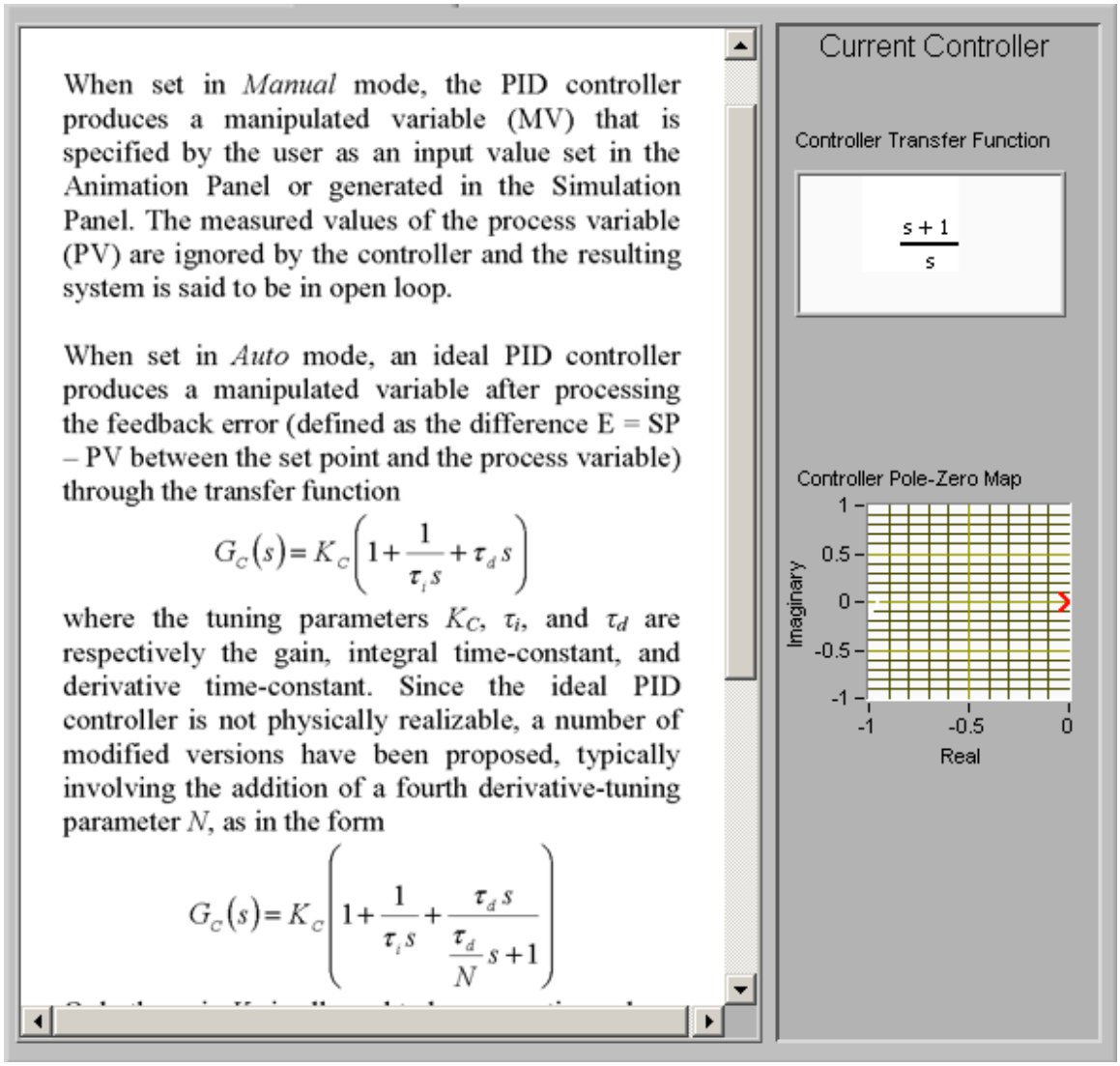

Figure 2-3. Navigation panel showing the controller window selection. An embedded PDF file describes the features of the controller, while the frame on the right-column shows the controller transfer function as well as the location of its poles and zeros.

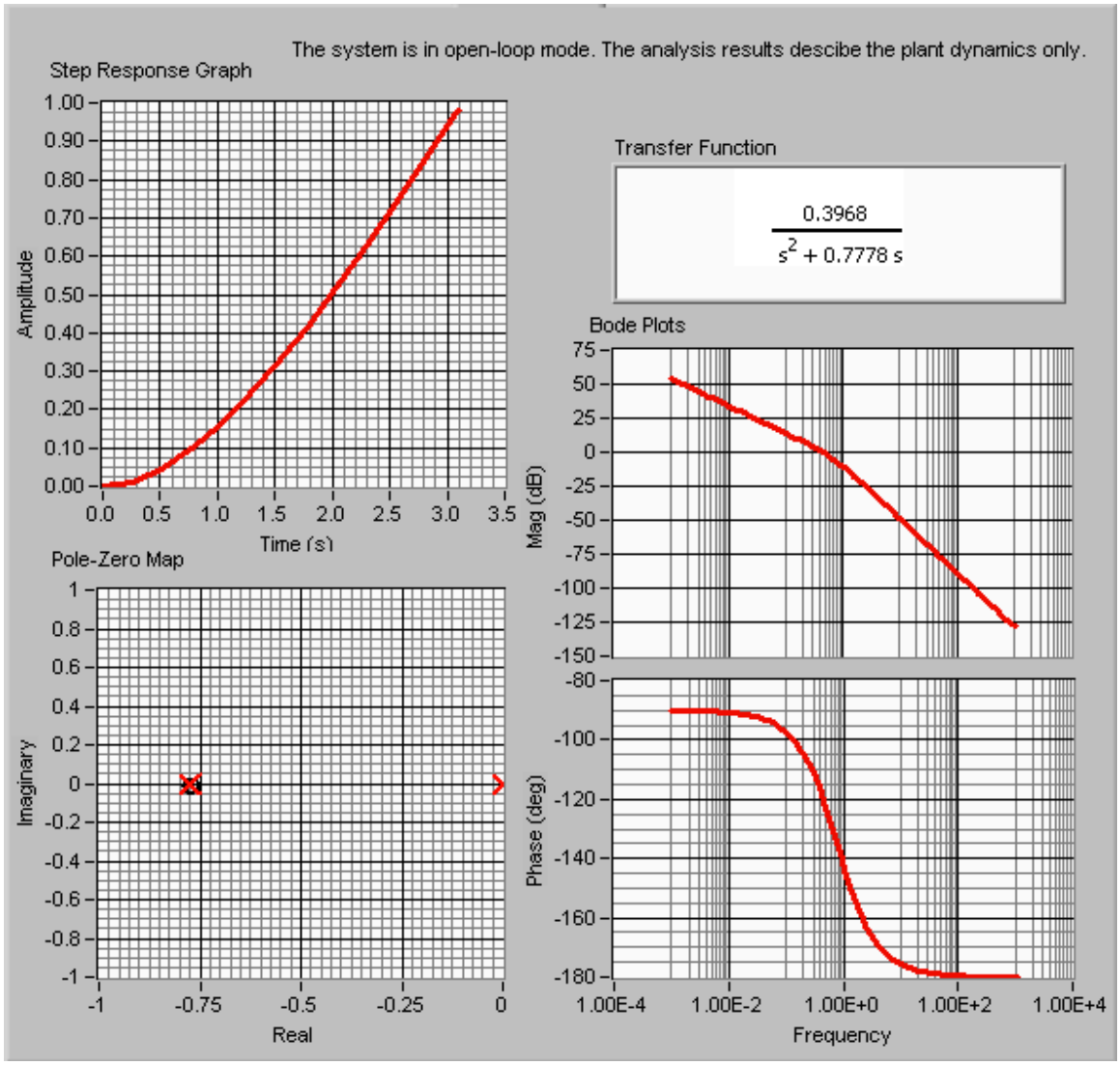

Figure 2-4. Navigation panel showing the analysis window selection. The results include the unit-step response, Bode plot, pole-zero map, and the transfer function of the plant given that the system is in open-loop mode, as indicated by the message across the top of the window. Equivalent results are presented by the Analysis window when the system is switched to closed-loop mode.

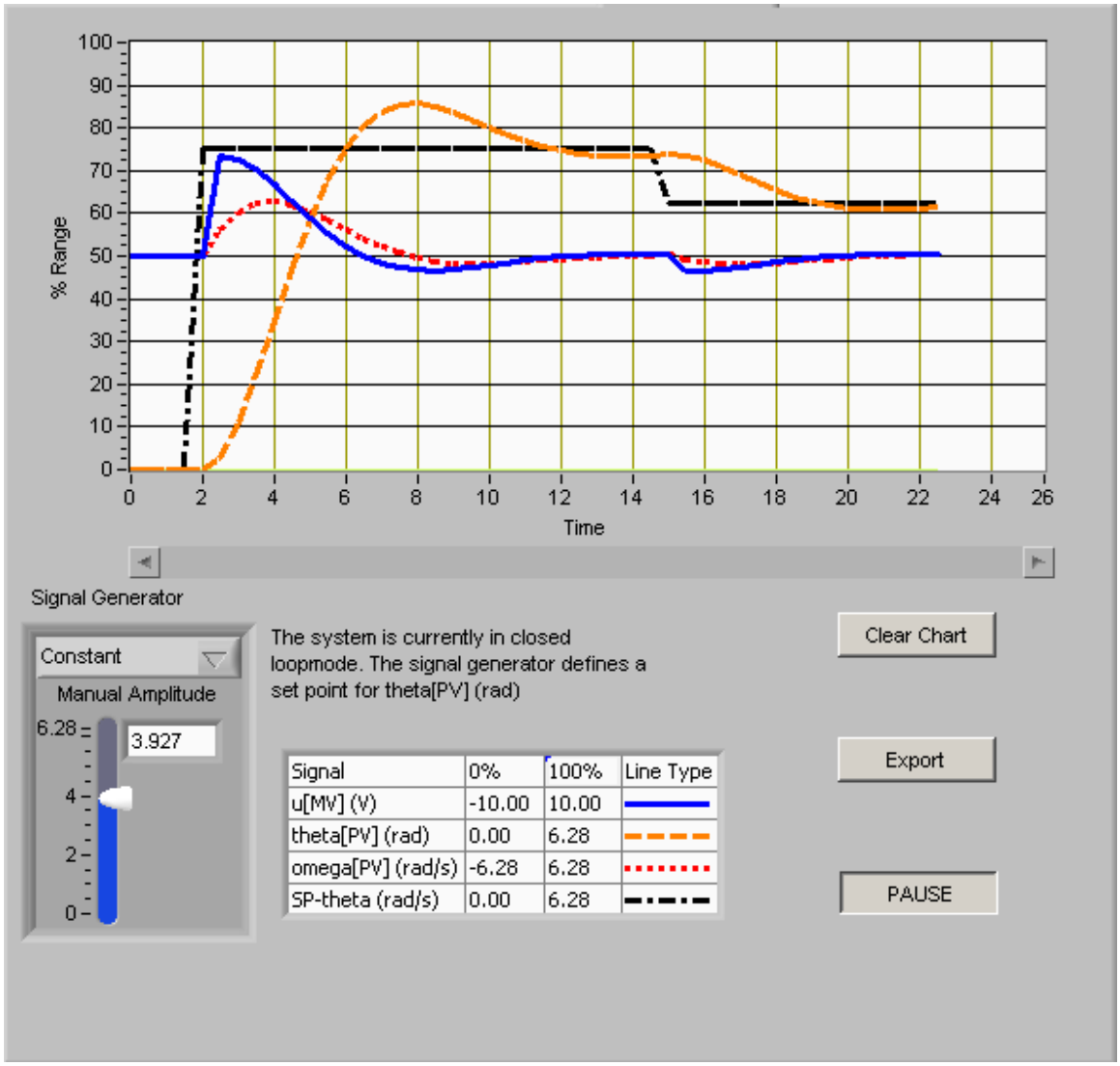

Figure 2-5. Navigation panel showing the simulation tab selection. The graph displays the traces of all the relevant variables, and the signal generator allows the introduction of set-point changes.

CHAPTER 3
DOUBLE INTEGRATOR CONTROL DESIGN

3.1 Introduction

3.1.1 Background

Most industrial applications operate under simple feedback controllers such as PID (Proportional-Integral-Derivative) loops. These controllers are often preferred over more advanced designs because they are inexpensive to implement and easily tuned. The integral action in PI and PID controllers allows them to track a step change in a set point or reject a step change in disturbance without causing steady-state offset. For a controller to be industrially useful, explicit, easily implementable and computationally inexpensive rules are required for tuning them. These rules often take the form of equations relating the parameters of the open loop system to the controller parameters that must be specified. Many of the most successful tuning relationships are developed by simulating a closed-loop system and seeking to minimize the error between the set point and the control variable [27][28].

Controllers with integral action are commonly used to compensate for disturbance or set-point changes that take the form of a step change. Unlike a pure proportional controller, a proportional plus integral (PI) controller can track the step change and eventually reduce the error to zero. Similarly, there is need for controllers that will track ramp type inputs. A PI controller produces steady-state offset for a ramp signal similarly to the way a proportional controller does for a step change. A controller with two integrators is required to reduce this error to zero. This phenomenon is well documented but few detailed studies exist [29][30]. A relevant study by Belanger and Luyben proposes a form of a proportional-integral-double integral (PI^2) controller [31]. Alvarez-Ramirez *et al.* further examined this controller and coined the nomenclature PI^2 [32].

Belanger and Luyben take an analytical approach to the tuning of PI^2 controllers, and relate the prescribed tuning parameters to the ultimate gain and period of the plant.

They assume that the plant model consists of only an integrator plus dead-time, which is approximately a standard first-order plus dead-time model only for large time constants. This simplifies the problem to a point where an analytic solution describing the optimal tuning is feasible. Furthermore, these authors restrict the two integral time-constants of the PI^2 controller to have values which cause repeated roots in the controller transfer function, minimizing any adverse impact on the closed-loop stability. These assumptions allow the parameters to be chosen as a function of the ultimate gain and period to achieve desired characteristics such as a specified closed-loop damping ratio. This study is based on a numerical optimization approach rather than an algebraic one, and uses a more general error metric than damping ratio. The effect of the restriction relating the two controller integral time constants can be examined since optimal tunings with and without the restriction can be calculated. Additionally, faster systems can be controlled using this approach since true first order responses are allowed by the chosen model form, in contrast to the more limited type of responses available when the model is restricted to consist of only one integrator.

This chapter proposes tuning relationships that, like existing PID tuning schemes, involve the parameters of a first-order plus dead-time (FODT) plant. Simulations of a closed loop system combining this FODT transfer function with one for a PI^2 controller yield error metrics which can be optimized to develop an appropriate tuning relationship. The remainder of this Section proposes three possible schemes for implementing a PI^2 controller and compares them to the scheme proposed by Belanger and Luyben. A method for tuning each of these schemes via the optimization of the integral of the time-weighted absolute error is described in Section 3.2. Section 3.3 presents the resulting optimal tuning relationships while Section 3.4 analyzes these results and compares them to tuning prescriptions presented in the literature.

3.1.2 Problem Statement

A PI^2 controller could be realized through various structures where the common feature is the double integrator. Transfer functions for three PI^2 are given in the following schemes:

$$Scheme1 \quad G(s) = K_c + K_{i1}\frac{1}{s} + K_{i2}\frac{1}{s^2} \tag{3–1}$$

$$Scheme2 \quad G(s) = K_c\left(1 + \frac{1}{\tau_{i1}s}\right)\left(1 + \frac{1}{\tau_{i2}s}\right) \tag{3–2}$$

$$Scheme3 \quad G(s) = K_c\left(1 + \frac{1}{\tau_{i1}s}\right)^2 \tag{3–3}$$

Scheme 1 is equivalent to the form used by Belanger and Luyben, but is written in terms of three separate gains rather than time constants [31]. This scheme is the most general of the three; however, from an implementation standpoint it would be useful to configure a PI^2 controller using two standard PI controllers in series. Scheme 2 shows such an approach. Since the gains of the two PI controllers are multiplicative the gain of the second controller in the series can be taken as unity without loss of generality, thus only one gain is used in Scheme 2. There is however a loss of generality in Scheme 2 relative to the form of Scheme 1. Any Scheme-2 controller can be easily represented as a Scheme-1 controller, but Scheme-1 controllers where $K_{i1}^2 < 4K_cK_{i2}$ do not have a Scheme-2 representation. This inequality constraint arises from the factorization of Scheme 1 and the desire to avoid complex values for the integral time constants in Scheme 2. It is also of interest to consider the special case of Scheme 2 where the two integral time constants are restricted to have the same value, as in Scheme 3. This is analogous to the restriction in time constants proposed by Belanger and Luyben and also simplifies substantially the tuning procedure. If Scheme 3 yields similar performance to Scheme 2, then it would be a more attractive option to adopt due to the simplicity of implementation and tuning. All three schemes are considered in this study.

3.2 Approach

For the purposes of this study these PI^2 control schemes are arranged with a plant in a standard feedback loop as shown in Figure 3-1 where r is the desired set point, u is the input prescribed by the controller and y is the plant output.

The performance of a controller can be measured in a variety of ways. A common approach is to use one of several metrics of the error of the system over time. In this study we adopt the integral of the time-weighted absolute error (ITAE) as defined by the expression

$$ITAE = \int_0^\infty t|e|dt \tag{3–4}$$

where $e = r - y$ is the closed loop feedback error.[28]. In this study, the transfer function G_c is a PI^2 control scheme and G_p is the first-order plus dead time plant transfer function

$$G_p(s) = \frac{K}{\tau s + 1} e^{-\theta s} \tag{3–5}$$

The ITAE is attractive as a performance measurement because it is weighted lightly at early times where error is inevitable since the system cannot respond instantly, and weighted more heavily at later times, where error could be indicative of instability or steady-state offset. Since the presence of an additional integrator tends to destabilize the closed-loop system, a reasonably conservative metric such as the ITAE is a reasonable choice for the double integrator problem and preferred to other standard metrics such as the integral of the average error and the integral of the square of the error. The ITAE approach is further attractive since optimization of this metric has been shown to be an effective method for developing tuning rules for traditional PID controllers. Simulations can be conducted to calculate the ITAE for a set of plant and controller parameters and the controller parameters can be optimized for a variety of plant parameter sets to yield tuning relationships.

A numerical simulation system is established using MATLAB. The simulation includes the first-order plus dead-time process, as in Equation (3–5), controlled by a PI^2

controller subjected to a ramp set point input. The three parameters of the process – gain, time constant, and delay – and the parameters of the controller can all be varied. The ITAE (3–4), formally an infinite integral, is truncated to

$$t_f = 15max(\tau, \theta) \tag{3–6}$$

This final time is longer than the time in which one would expect any effective controller to reduce the error to nearly zero. Thus the truncated ITAE is sufficient to identify poorly performing controllers and provides a good approximation of the true ITAE value for optimized controllers.

The process of setting the model parameters and optimizing the controller parameters is repeated for a variety of model parameters. A simplex algorithm is chosen for the optimization of parameters because of its ability to converge on accurate value in reasonable time and because it does not require gradients which are challenging to determine for the truncated ITAE. The results of this algorithm were confirmed for selected values of the parameters by performing an exhaustive numerical search. The data from the optimization is nondimentionalized and plotted versus the process parameters in a fashion similar to that adopted by existing tuning correlations for PID controllers. The general shape of the resulting curve suggests methods for fitting a linear approximation to the logarithm of the data. This is accomplished via a least-squares approach. Results from optimizations done on all the functional forms of the controller – namely Schemes 1, 2 and 3 – are obtained and examined independently of each other.

3.3 Optimal Tuning Relationships

Tuning relationships for each of the PI^2 schemes are independently developed using the simulation routine discussed above. The relationships apparent from the resulting data are well suited to being represented as exponential functions of nondimentionalized parameters. In a fashion similar to existing PI tuning relationships, the parameters of the controller are nondimentionalized using the plant gain and time constant, as appropriate

Table 3-1. Tuning relationships for all three schemes

Scheme	G_c PI^2 Transfer Function	Tuning Prescription
1	$G(s) = K_c + K_{i1}\frac{1}{s} + K_{i2}\frac{1}{s^2}$	$KK_c = 1.1616\frac{\theta}{\tau}^{-0.6880}$ if $\frac{\theta}{\tau} \leq 1$ $KK_c = 0.8133\frac{\theta}{\tau}^{-0.1894}$ if $\frac{\theta}{\tau} < 1$ $\tau KK_{i1} = 0.2892\frac{\theta}{\tau}^{-1.7942}$ $\tau^2 KK_{i2} = 1.2358\frac{\theta}{\tau}^{-1.0495}$
2	$G(s) = K_c\left(1 + \frac{1}{\tau_{i1}s}\right)\left(1 + \frac{1}{\tau_{i2}s}\right)$	$KK_c = 1.0083\frac{\theta}{\tau}^{-0.7854}$ if $\frac{\theta}{\tau} \leq 1$ $KK_c = 0.8480\frac{\theta}{\tau}^{-0.1600}$ if $\frac{\theta}{\tau} < 1$ $\frac{\tau}{\tau_{i1}} = 0.4880\frac{\theta}{\tau}^{-0.5501}$ $\frac{\tau}{\tau_{i2}} = 0.3805\frac{\theta}{\tau}^{-0.8150}$
3	$G(s) = K_c\left(1 + \frac{1}{\tau_{i1}s}\right)^2$	$KK_c = 1.0760\frac{\theta}{\tau}^{-0.7868}$ if $\frac{\theta}{\tau} \leq 1$ $KK_c = 0.8318\frac{\theta}{\tau}^{-0.0964}$ if $\frac{\theta}{\tau} < 1$ $\frac{\tau}{\tau_i} = 0.4317\frac{\theta}{\tau}^{-0.6820}$

[28][30]. The resulting relationships are given in Table 3-1. The same results are also examined graphically. For the purposes of clarity, only averaged data and the relationships described in Table 3-1 are shown in the plots.

Figure 3-2 shows the optimization results for Scheme 1. In a fashion analogous to that used for PID tuning, the optimal controller parameters are nondimentionalized using the plant parameters and plotted on a log-log scale. All three parameters are plotted against the ratio of the plant's dead time and lag time constant. For the two integral gains, this yields plots that can be reasonably approximated by a linear least-squares fit. This is convenient for use in actual tuning since it allows a simple tuning rule equation such as those used for PI controllers [28].

The proportional gain plot in Figure 3-2a shows a curve that does not lend itself to description via a simple mathematical form. Since this representation is extremely useful for tuning, the graph is broken into two parts each of which is subjected to separate linear least squares fitting procedures. These linear fits, represented by the dashed traces in Figure 3-2, are generated by the equations given in line one of Table 3-1.

Figure 3-3 shows plots similar to those in Figure 3-2 but corresponding to the optimal tuning for Scheme 2. It should be noted that in addition to suggesting a means of

physically implementing the controller, structuring the PI^2 controller as two separate PI controllers simplifies the nondimentionalization task. Rather than requiring a combination of several constants, the integral reset parameters can be nondimentionalized simply by using the plant lag time-constant. Again it is useful to split the proportional gain curve into two segments to reasonably model the data with a linear fit. For simplicity in applying the tuning rules, the same split point, $\frac{\theta}{\tau} = 1$, is used.

Figure 3-4 shows the optimal tunings for the Scheme-3 PI^2 controller. Only one plot is presented for the integral time constant, since this parameter is the same for both integral terms of Scheme 3. The nondimentionalization procedure and the splitting of the gain curve into two sections for regression purposes are identical to those performed on the previous figures.

3.4 Analysis and Discussion

3.4.1 Comparison of the PI^2 Schemes Considered

Closed-loop ramp responses are simulated to examine the effects of the controllers prescribed by the above relationships. Four plant models are used: (i) a plant with similar time constant and delay, (ii) a plant with a long delay, (iii) a plant with a short delay, and (iv) and a plant with a large delay and a large time constant. The recommended tuning for each of the controller forms is used, and the results are plotted in Figure 3-5.

Since all three schemes perform similarly well, the preferred scheme can be chosen based on practical implementation concerns rather than performance. Scheme 3 is preferred in the absence of difference in performance because it can be implemented using standard PI controllers and has two parameters that require tuning rather than three.

3.4.2 Comparison with Literature

Time responses are also calculated based on the controller scheme and tuning proposed by Belanger and Luyben [31]. Belanger and Luyben's control scheme assumes that the plant can be represented as an integrator plus a dead time. Thus a plant with a large time constant with respect to the dead time would satisfy the assumption, while a

plant with a smaller time constant would not. Thus, the Belanger-Luyben tunings would be expected to be less effective for the latter case.

Figure 3-6a shows a scenario where τ and θ are of similar magnitude. In this case both controllers successfully achieve the zero offset requirement at sufficiently long time. This is not surprising, as it can be shown that either form eventually eliminates offset provided the controller produces a stable closed loop. However the PI^2 controller proposed by Scheme 3 shows substantially less transient behavior and would be described as a better performing option by most metrics. To provide a fair comparison it is necessary to examine the case where the plant time constant is so large that the first order lag can be approximated by an integrator, as this is the case for which Belanger and Luyben's tunings are designed. Thus, a scenario with a large lag time constant and smaller gains and dead times is examined in Figure 3-6b. The two controllers produce similar responses. Either controller could be described as superior depending on your performance metric of choice, but both controllers perform adequately. Finally, the Figure 3-6c considers a scenario in the other extreme, where a large dead-time is used with a smaller lag. This is both a challenging control problem and well outside the realm for which Belanger and Luyben's tuning criteria was designed. Here the Scheme 3 controller shows a long transient behavior that is not ideal but probably unavoidable given the large dead-time. The Belanger-Luyben tuning prescription, however, shows unstable behavior, making it unsuitable for this plant.

3.4.3 Validation of the Optimization Results

Several approaches were taken to verify the validity of the results . All optimal tuning values produced by the numerical optimization data points are examined via Nyquist techniques to verify the stability of the closed loop. Furthermore contour plots constructed for selected values of the $\frac{\theta}{\tau}$ ratio to examine the convexity of the ITAE as a function of the controller parameters provide a supplemental validation tool. Figure 3-7 shows the ITAE as a function of the Scheme-3 parameters, K_c and τ_i for various values of the plant

parameters. Specifically, the plots show a K value of 1, a τ value of 10 and θ values of 2.5, 10 and 250. Similar graphs for varying K values are not shown since these serve only to scale the results, only changing the ratio of delay to time constant alters the shape of the contour. It should be noted that the values predicted by the correlations in table 3-1 are near the optimal values for moderate θ to τ ratios, and are also well away from the regions with steep gradients that characteristically indicate an approach to the stability boundary. While these plots do not constitute a proof of convexity of the systems in question, they do serve to lend credibility to the accuracy of the minima identified by the numerical optimization procedure.

3.5 Conclusions

A PI^2 double-integral controller implemented either as a general three term controller or as the more specialized but easily implementable two-PI-controllers-in-series scheme is shown to reduce steady state error to zero for step and ramp type set points and tuning correlations are developed for first order plus dead time systems. These correlations prescribe controller parameters that minimize the ITAE as an exponential function of nondimentionalized system parameters. Ramp-response simulations show that the performance of the there proposed control schemes are comparable, so Scheme 3 is the preferred option due to its simplicity of implementation and tuning. These tunings are also shown to be competitive with and in some cases superior to the analytically developed tuning relations proposed by Belanger and Luyben for similar PI2 controllers [31]. Stability of the systems recommended by the correlation has been established via Nyquist methods but studying their robustness with respect to both errors in the correlations and in the process parameters would prove to be interesting future work.

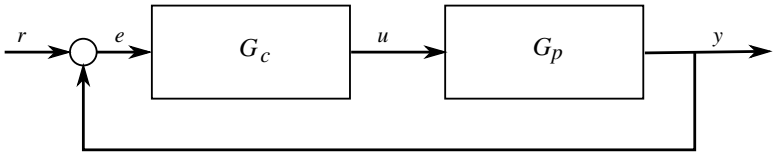

Figure 3-1. Closed loop control structure featuring a plant (Gp), a PI2 controller (Gc) and the set point (r), output (y), input (u) and feedback error (e) signals.

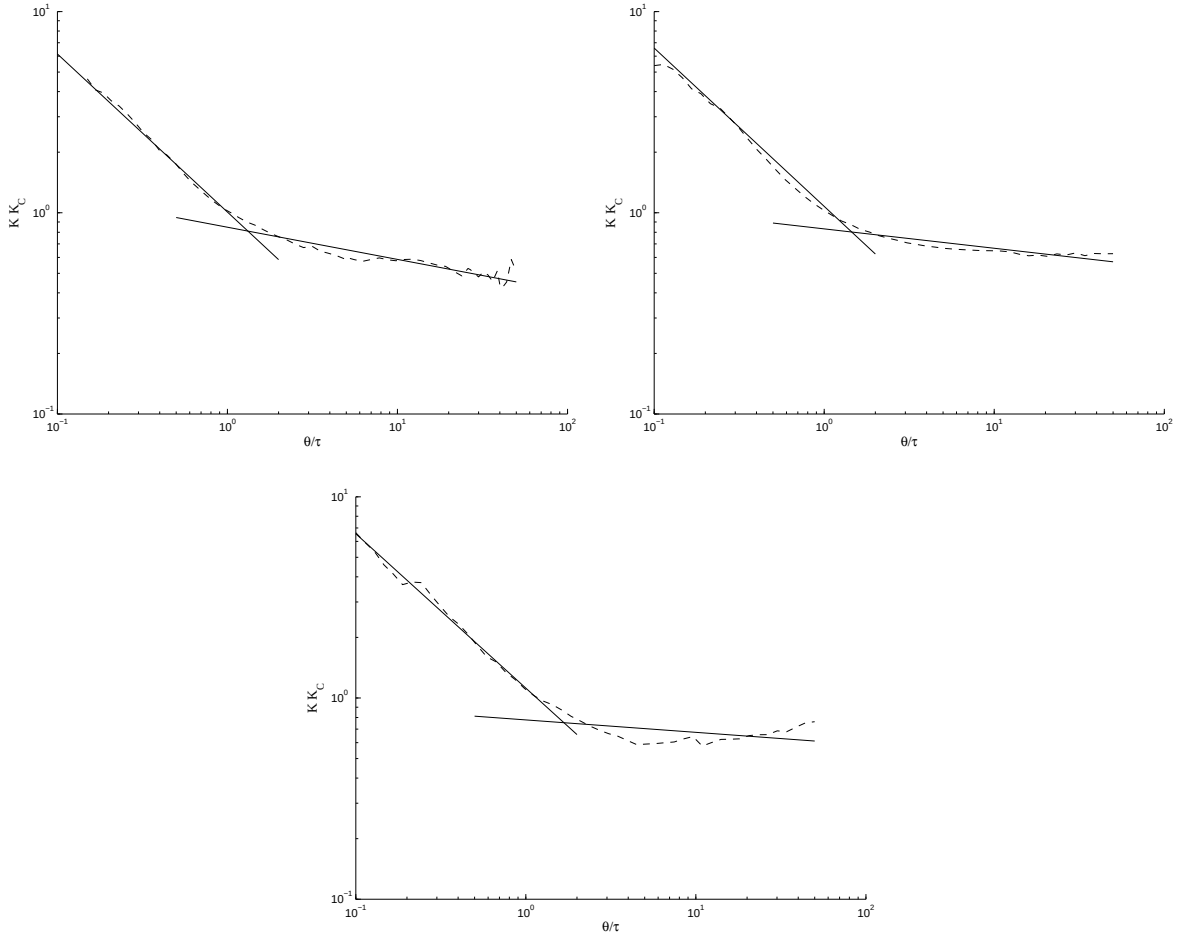

Figure 3-2. Scheme 1 optimal ITAE tuning relationship.

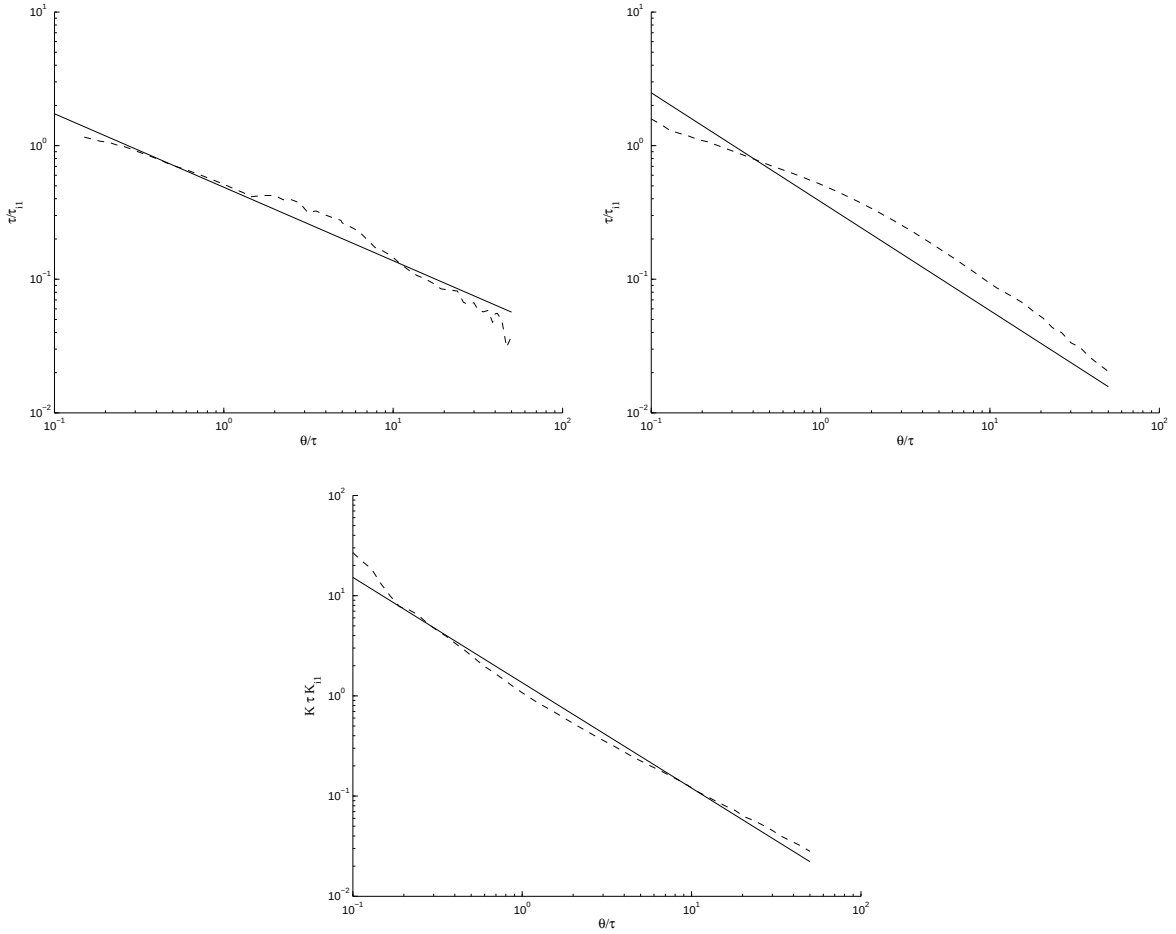

Figure 3-3. Scheme 2 optimal ITAE tuning relationship.

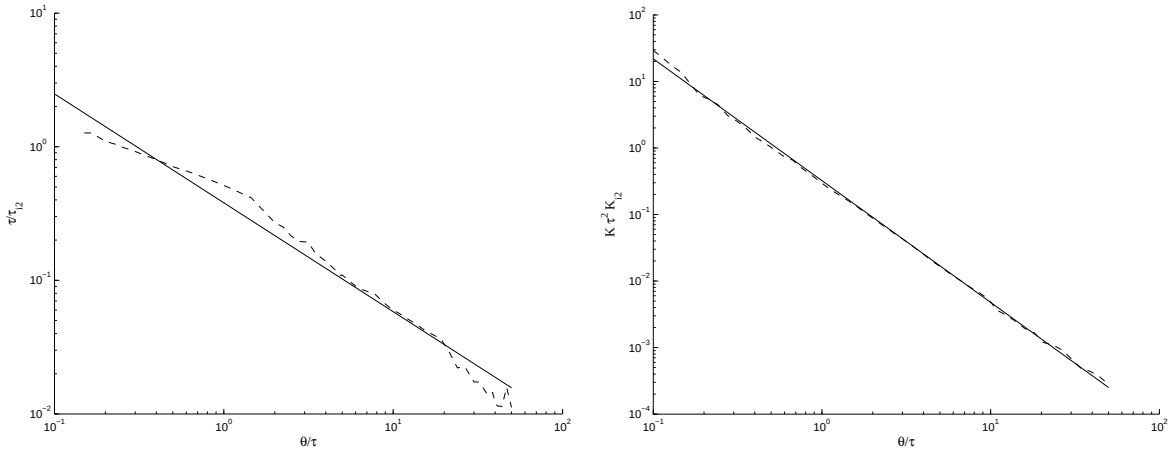

Figure 3-4. Scheme 3 optimal ITAE tuning relationship.

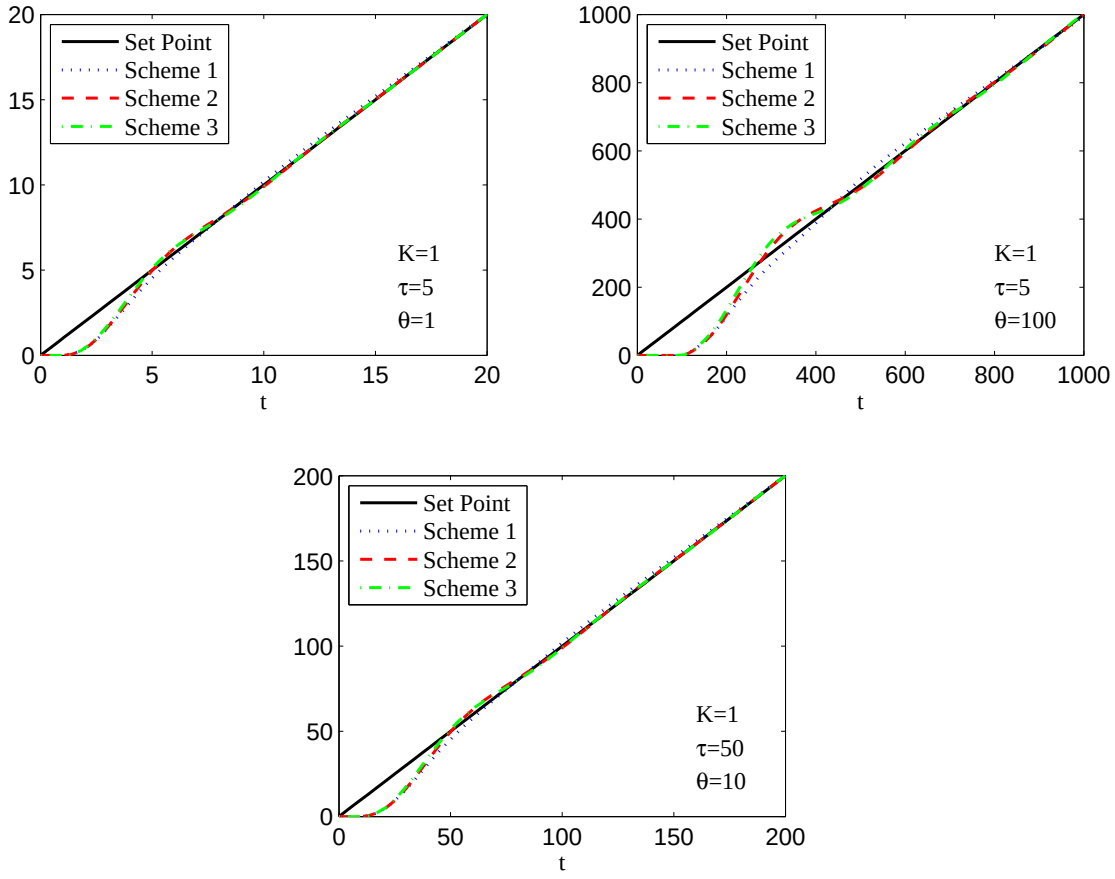

Figure 3-5. Time responses of the three PI2 schemes for various values of plant parameters. Tuning parameters are as prescribed by Table 3-1.

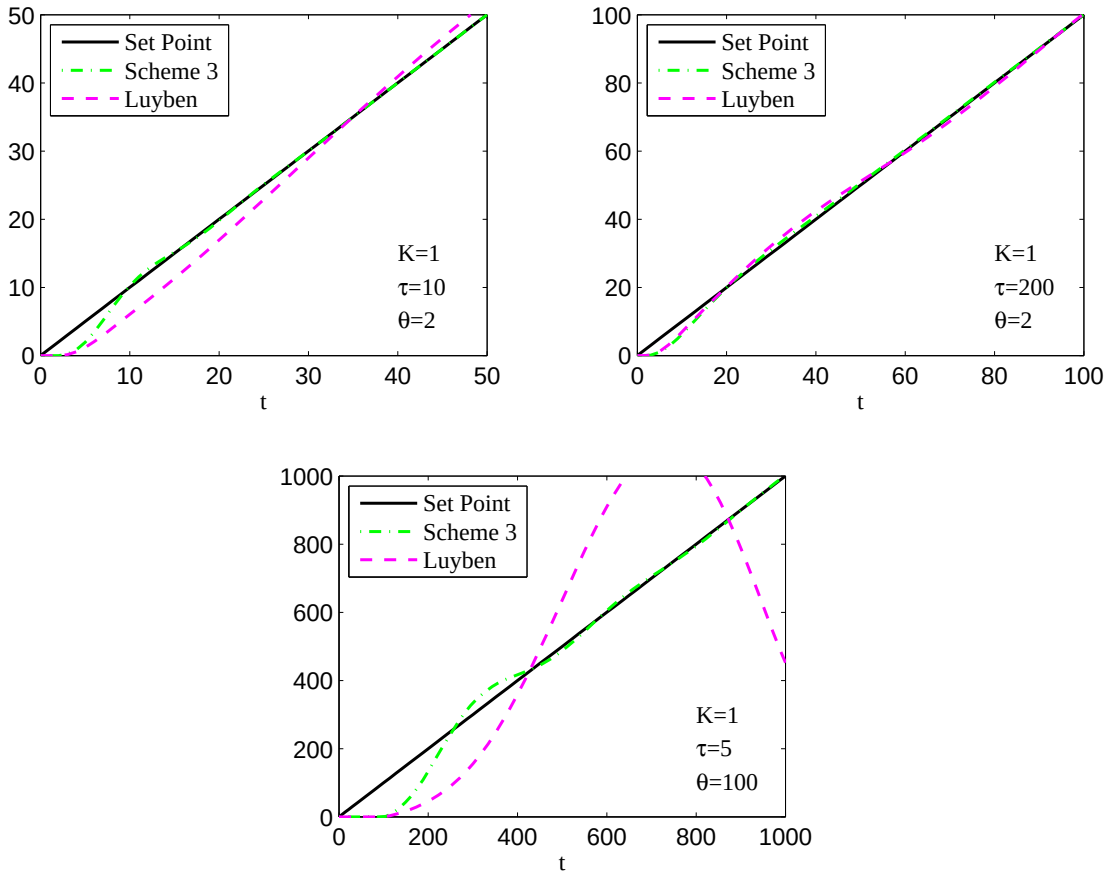

Figure 3-6. Comparison of Luyben's PI2 controller with scheme 2 proposed here.

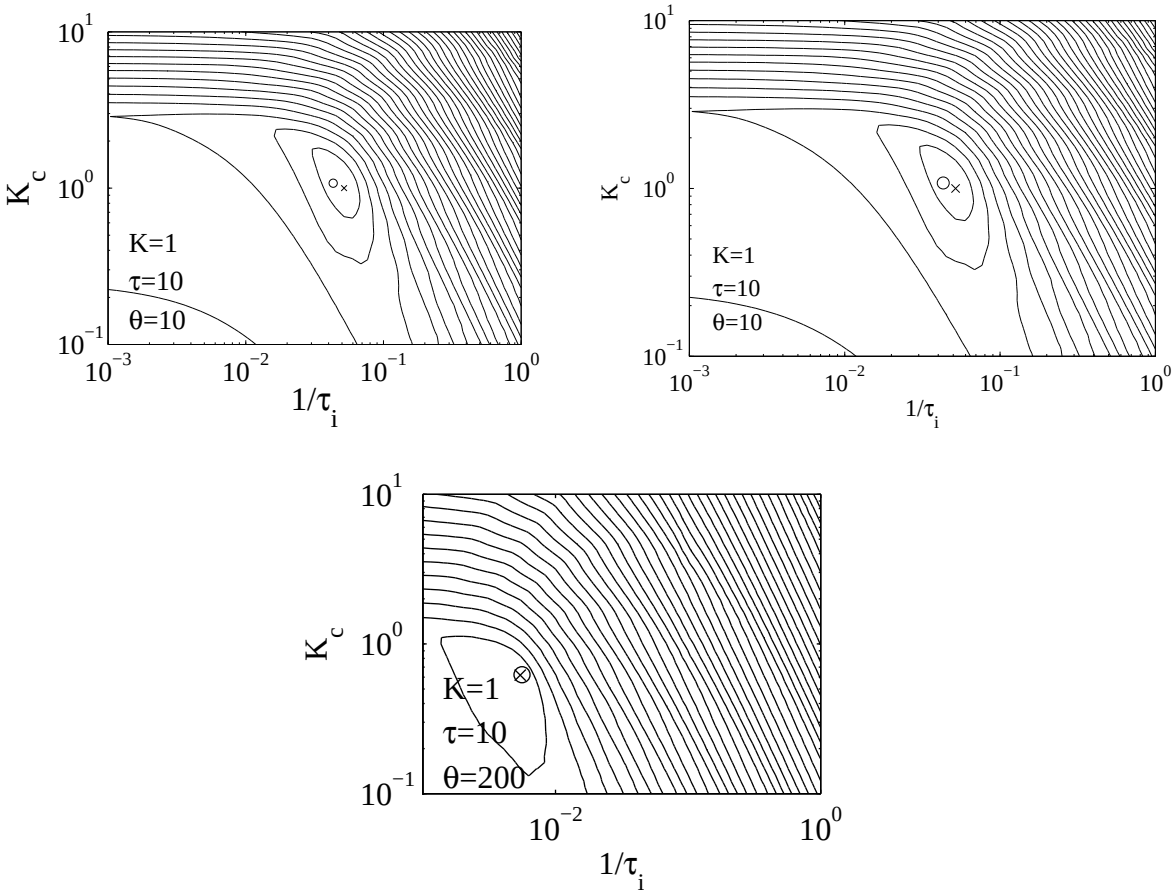

Figure 3-7. Contour plots for one time constant calculations.

CHAPTER 4
OFFSET FREE MODEL PREDICTIVE CONTROL

4.1 Introduction

Model predictive controllers (MPC) are in widespread use in industry to adjust the manipulated variables (also known as inputs) of systems of interest with the goal of ensuring good performance of a set of controlled variables (also known as outputs). The class of systems susceptible to MPC control is very large, and it includes chemical reactors, distillation columns, hydrocarbon crackers, entire refineries, as well as cement mills and ovens, robots, automobiles, aircraft, trains, amphibious vehicles, autonomous vehicles, gas turbines, and jet engines, among others. Additional systems of interest for which MPC can deliver appropriate automatic manipulations to ensure suitable functional performance include biological environments, encompassing, for example, bioreactors designed to produce insulin, and human organs such as the pancreas and its insulin and glucagons production processes, among many others. The systems of interest also include abstract human constructs that involve inputs and outputs that change with time, such as the stock market, financial models, manufacturing production models, scheduling networks for production and transportation, for example. These systems of interest are often referred to as the plant.

The popularity of MPC controllers stems from their ability to deliver good performance when deployed on complex systems featuring large numbers of inputs and outputs. Furthermore, the MPC controllers can satisfy constraints imposed by the user, such as ensuring that input values do not deviate from a range specified by maximum minimum bounds. The MPC technology has established itself as the primary means of controlling industrial systems with multiple inputs and multiple outputs.

In spite of its high-value capabilities, it is well known that special measures must be taken to ensure that the MPC design make the outputs achieve their specified set-point values at steady state without appreciable error. The error observed after all transient

responses settle into approximately constant patterns, known as steady-state offset, is often magnified by the presence of disturbances affecting the plant.

A comprehensive discussion of design measures proposed for obtaining offset-free performance at steady state for model predictive controllers is given in Muske and Badgwell [33] and in Pannocchia and Rawlings [34].

The design measures advocated in the references mentioned above are challenging to deploy in a systematic fashion because they require including in the model used for designing the MPC controller a dynamic representation of the disturbances that is often inconsistent with the structural form of the actual disturbances that act on the plant. Hence, the proposed disturbance representations are fictitious models adopted because in cases of interest they bring about the beneficial consequence of eliminating steady-state. Unfortunately, different fictitious representations can cause different performance qualities during the transient period that precedes the onset of steady state. There are no systematic guidelines on how to specify an optimal disturbance structure. These approaches are therefore not sufficiently systematic because the design process calls for significant investments of engineering effort to discriminate among many possible representations of disturbance dynamics.

In addition, the techniques in these references involve the inclusion of additional design variables known as state and input targets that are involved in the solution of a supplementary numerical optimization problem that must be solved on line at every instant that the controller needs to make an adjustment. The process of optimization increases the numerical cost of deploying the MPC algorithm, and introduces additional design complications because special cost matrices must be specified without the benefit of established guidelines. Furthermore, no systematic guidelines are given to assist in the specification of effective values for the input target variable, an impediment to ensuring adequate transient performance without investing a significant amount of added engineering effort. It is often observed that the behavior of the model predictive

controllers designed according to the guidelines given in these references behave in an unintuitive fashion, and thus the methodology lacks the robustness desirable in a mission-critical or high-performance control system. Performance degradation in the form of unacceptable steady-state offsets can also be observed when the operating conditions of the plant change, which may call for a significant investment in additional engineering resources to redesign the MPC structure to effectively address needs of new environmental conditions.

This chapter presents model predictive control algorithms that do not suffer from the shortcomings identified above. The model predictive controllers disclosed deliver offset-free performance, and can be deployed in a systematic fashion. Furthermore, the numerical computational burden required may be significantly lower than that of alternative approaches documented in the literature.

4.2 Problem Statement

This chapter presents four methods for designing and implementing Model Predictive Controllers (MPC) that deliver responses that are free of steady-state offset errors. The MPC methods disclosed are effective for eliminating offset, and hence ensure perfect set-point tracking at steady state, even when the plant being manipulated for the MPC controller is subject to the presence of unmeasured input disturbances and output disturbances that adopt constant values at steady state.

All MPC methods described in this chapter eliminate the need for including in the typical MPC performance index a number of computationally expensive and unintuitive concepts, including the steady-state targets required by previous MPC implementations. All MPC methods use a performance index that includes a set-point error tracking cost.

Figure 4-1 is a block diagram illustrating a system with a Model Predictive Controller (MPC). It includes and a plant and controller.The plant takes one or more inputs, u, and produces one or more outputs, y. The output and a desired value known as the set point are fed to the MPC Controller. At every sampling interval MPC Controller determines the

input by making use of a Model System, an Estimation System, a Prediction System, and an Optimization System.

A typical control performance specification is that the input prescribed by the MPC Controller produces a sequence of output values that are close to a specified sequence of set-point values. The steady-state offset, often referred to simply as offset, e_{ss}, is defined as the value adopted by the feedback error after a sufficiently long time to allow all transient changes in the output to disappear and allow the output to adopt a constant pattern. Mathematically, the offset is represented as a limit, namely the error as time tends to infinity, whereas in practice the concept of an infinitely long time index is replaced by a sufficiently long but finite time index. The offset is therefore a measure of how much the output differs from the specified set-point value after a relatively long period of operation. The MPC control design methodology discussed here ensures that the resulting controllers attain zero offset, hence forcing the output to match the set point at steady state even under conditions where there is a change in the set point or where Plant 105 is subject to the presence of disturbances that are not measured. Thus, the desirable feature delivered by the MPC designs disclosed here is the attainment of the condition

The Model System of an MPC Controller is a mathematical algorithm that describes the operation of the plant. A variety of linear and non-linear models can be used, but the focus of this chapter is on the linear MIMO state space plant model

$$x(k) = Ax(k-1) + Bu(k-1) + d(k) \tag{4–1}$$

$$y(k) = Cx(k) + p(k) \tag{4–2}$$

where x is a vector of states of the plant, u is the input from the controller, y is the output from the plant to the controller and d and p are state and output disturbances respectively. The sizes of these vectors are given by:

$$x \in \mathbb{R}^n, y \in \mathbb{R}^m, u \in \mathbb{R}^p, d \in \mathbb{R}^n, p \in \mathbb{R}^m \tag{4–3}$$

The Prediction System describes the anticipated future values of the variables associated with the Model System and the Plant. At sampling interval k the Prediction System can produce numerical estimates of the future values of outputs and states, and of current and future values of the input and input-increment vectors. These predictions can be represented compactly via the augmented vectors

$$x = \begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x(k+N_p) \end{bmatrix} \quad y = \begin{bmatrix} y(k+1) \\ y(k+2) \\ \vdots \\ y(k+N_p) \end{bmatrix} \quad u = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N_c) \end{bmatrix} \tag{4–4}$$

where

$$\Delta u = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_c) \end{bmatrix} \tag{4–5}$$

$$\Delta u(k) = u(k) - u(k-1) \tag{4–6}$$

The indices N_p and N_c are known as the prediction horizon and control horizon respectively. Therefore, the MPC controller adjusts a total of $N_c \times p$ manipulated variables, and the vector y contains a total of $N_p \times m$ rows. Furthermore, the first p rows of u contain the current input vector.

The Prediction System considered in this chapter operates by iteratively solving the model equations (4–1) and (4–2) for future sampling intervals. This process can be represented by the augmented predictor in the equations

$$x = A_a x(k) + B_a u + p \tag{4–7}$$

$$y = C_a x + d \tag{4–8}$$

where

$$A_a = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix} \tag{4–9}$$

$$B_a = \begin{bmatrix} B & 0 & \dots & 0 & 0 \\ AB & B & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A^{N_c-1}B & A^{N_c-2}B & \dots & AB & B \\ A^{N_c}B & A^{N_c-1}B & \dots & A^2B & \sum_{i=0}^{1} A^iB \\ A^{N_c+1}B & A^{N_c}B & \dots & A^3B & \sum_{i=0}^{2} A^iB \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A^{N_p-1}B & A^{N_p-2}B & \dots & A^{Np-Nc+1}B & \sum_{i=0}^{Np-Nc} A^iB \end{bmatrix} \tag{4–10}$$

$$C_a = \begin{bmatrix} C & 0 & \dots & 0 \\ 0 & C & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & C & \dots & C \end{bmatrix} \tag{4–11}$$

and where

$$d = \begin{bmatrix} d(k) \\ d(k+1) \\ \vdots \\ d(k+N_p) \end{bmatrix} \quad p = \begin{bmatrix} p(k) \\ p(k+1) \\ \vdots \\ p(k+N_p) \end{bmatrix} \tag{4–12}$$

The Estimation System of an MPC Controller is used to compute exact or approximate values of the state vector using available measurements and information provided by the Model System. More specifically, the Estimation System generates a sequence of estimated state values $\hat{x}$ that correspond exactly or approximately to the sequence of state values x produced by the plant. Exact values of disturbances may not be available. Thus, the

notation of $\hat{d}$ and $\hat{p}$ is introduced. Similar to $\hat{x}$, these vectors contain values that may be exact or estimated. Since unlike prior offset-free MPC implementations, those described here do not require disturbance estimation, another possible alternative is to set$\hat{d} = \hat{p} = 0$.

When the $\hat{d}$ and $\hat{p}$ vectors are not measured or identically zero, the sequence of estimated states can be calculated using an Estimation System designed using schemes for state estimation that are well known in the literature. For example the estimator:

$$\hat{x}(k) = A\hat{x}(k-1) + Bu(k) + L_x\left[y - C\hat{x}\right] \tag{4–13}$$

$$\hat{d}(k) = \hat{d}(k-1) + L_d\left[y - C\hat{x}\right] \tag{4–14}$$

$$\hat{p}(k) = \hat{p}(k-1) + L_p\left[y - C\hat{x}\right] \tag{4–15}$$

$$\tag{4–16}$$

where L_x, L_d, and L_p are the estimator gains, is well documented in the MPC literature. Details regarding estimation, including one particular alternative estimator are discussed in section (4.4).

The Optimization System of an MPC Controller finds the optimal value of the input vector u or the incremental input vector Δu, used as inputs to the Plant. Typically this is done by minimizing a performance index. This minimization is often performed under restrictions defined in a constraint set. Since the goal of this chapter is offset elimination, which cannot be achieved when constraints are active at steady state, unconstrained minimization is the focus of this discussion.

An MPC Controller uses a set of MPC Parameters that are defined by the user of the control system. Two central MPC Parameters are the prediction horizon N_p and the control horizon N_p , which can be specified to MPC Controller 110 as fixed values or as rules that describe time varying horizons. Other MPC Parameters include the values of cost matrices, constraints, and heuristic information, such as the variance of measurement and disturbance noise.

The following section examines four different predictive control methods that eliminate offset in the Plant. Method I uses an integral state vector in the Model System, Method II uses an input-velocity cost, Method III combines both an integral state vector and an input velocity, and finally Method IV uses a first integral state vector and a second integral state vector.

4.3 Offset-Free MPC Methods Proposed

The concepts of integral states and velocity weighting can be combined to yield several different MPC design methods. Perhaps the simplest approach in the addition of explicit integral states to the prediction.

4.3.1 Method I: Integral States

In Method I the integral state equation

$$z(k) = z(k-1) + r(k-1) - y(k-1) \tag{4–17}$$

is added to the plant model described by equations (4–1) and (4–2). This plant model is used to create the prediction system

$$\begin{aligned} x(k+j) &= Ax(k+j-1) + Bu(k+j-1) && (4–18)\\ y(k+j) &= Cx(k+j) && (4–19)\\ z(k+j) &= z(k+j-1) + r(k+j-1) - y(k+j-1) && (4–20) \end{aligned}$$

Defining the augmented vector z as

$$z = \begin{bmatrix} z(k+1) \\ z(k+2) \\ \vdots \\ z(k+N_p) \end{bmatrix} \tag{4–21}$$

results in a predictor that can be written in the form

$$y = C_I \hat{x}(k) + D_I u + F_I \hat{d} + E_I \hat{p} \tag{4–22}$$

$$z = M_I \left[z(k) + r(k) - y(k) \right] + N_I \left(r - y \right) \tag{4–23}$$

where

$$C_I = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix} \tag{4–24}$$

$$D_I = \begin{bmatrix} CB & 0 & \dots & 0 & 0 \\ CAB & CB & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{N_c-1}B & CA^{N_c-2}B & \dots & CAB & B \\ CA^{N_c}B & CA^{N_c-1}B & \dots & CA^2B & \sum_{i=0}^{1} CA^iB \\ CA^{N_c+1}B & CA^{N_c}B & \dots & CA^3B & \sum_{i=0}^{2} CA^iB \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & \dots & CA^{Np-Nc+1}B & \sum_{i=0}^{Np-Nc} CA^iB \end{bmatrix} \tag{4–25}$$

$$F_I = \begin{bmatrix} C \\ \sum_{i=0}^{1} CA^i \\ \vdots \\ \sum_{i=0}^{N_p-1} CA^i \end{bmatrix} \tag{4–26}$$

$$E_I = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} \tag{4–27}$$

$$M_I = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} \tag{4–28}$$

$$N_I = \begin{bmatrix} 0 & 0 & \dots & 0 \\ I & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ I & I & \dots & I \end{bmatrix} \tag{4–29}$$

The Optimization System minimizes the performance index

$$J = (r - y)^T Q_a (r - y) + u^T R_a u + z^T S_a z \tag{4–30}$$

which contains terms for set-point tracking, input action and integral states, where

$$r = \begin{bmatrix} r(k+1) \\ r(k+2) \\ \vdots \\ r(k+N_p) \end{bmatrix} \tag{4–31}$$

In the simplest configuration the weights are defined as

$$Q_a = diag\left(Q, Q, \ldots, Q, Q\right) \tag{4–32}$$

$$R_a = diag\left(R, R, \ldots, R, R(N_p - N_c)\right) \tag{4–33}$$

$$S_a = diag\left(S, S, \ldots, S, S\right) \tag{4–34}$$

where Q, R, and T are weighting matrices that define the relative costs associated with each output, input and integral, respectively. The minimum of this performance index occurs at $\nabla J_u = 0$.

Solving the minimization of the performance index for u yields the control law

$$\begin{aligned} u &= \left(K_I + K_{zI} N_I\right)\left[r - C_I \hat{x}(k)\right] - F_I \hat{d}(k) - E_I \hat{p}(k)\right] \\ &\quad + K_{zI} M_I \left[z(k) + r(k) - y(k)\right] \end{aligned} \tag{4–35}$$

$$K_I = \left[D_I^T \left(N_I^T S_a N_I + Q_a\right) D_I + R_a\right]^{-1} D_I^T Q_a \tag{4–36}$$

$$K_z I = \left[D_I^T \left(N_I^T S_a N_I + Q_a\right) D_I + R_a\right]^{-1} D_I^T N_I^T S_a \tag{4–37}$$

$$\tag{4–38}$$

At each sampling interval the MPC controller performs this calculation and implements the first p rows of u as $u(k)$. If the plant is controllable and the closed loop is stable, this MPC controller is sufficient to eliminate offset for disturbances or changes in set point that have a constant final value.

Another alternative for addressing the possibility of disturbance in the use of incremental equations. If the equations (4–1) and (4–2) are shifted backwards in time

by one interval and subtracted from their unshifted forms, the result is

$$\begin{aligned}\hat{x}(k+j) &= \hat{x}(k+j-1) + A\hat{x}(k+j-1) - A\hat{x}(k+j-2) \\ &\quad + B\Delta u(k+j-1) + \hat{d}(k+j) - \hat{d}(k+j-1) \end{aligned} \tag{4–39}$$

$$\begin{aligned}\hat{y}(k+j) &= \hat{y}(k+j-1) + C\hat{x}(k+j) - C\hat{x}(k+j-1) \\ &\quad + \hat{p}(k+j) - \hat{p}(k+j-1) \end{aligned} \tag{4–40}$$

Since the case where the disturbances reach constant values in time is of interest, the assumptions

$$\hat{p}(k+j) = \hat{p}(k+j-1) \tag{4–41}$$

$$\hat{d}(k+j) = \hat{d}(k+j-1) \tag{4–42}$$

$$\tag{4–43}$$

are relevant and reduce (4–40) and (4–41) to

$$\hat{x}(k+j) = \hat{x}(k+j-1) + A\hat{x}(k+j-1) - A\hat{x}(k+j-2) + B\Delta u(k+j-1) \tag{4–44}$$

$$\hat{y}(k+j) = \hat{y}(k+j-1) + C\hat{x}(k+j) - C\hat{x}(k+j-1) \tag{4–45}$$

An apparent alternative method of formulating the incremental output equation would be to use the equation for the current time interval, k, rather than the $k+j-1$ interval, yielding

$$\hat{y}(k+j) = \hat{y}(k-1) + C\hat{x}(k+j) - C\hat{x}(k-1) \tag{4–46}$$

However both of these methods yield the augmented prediction system in equations

$$y = C_{II}^{0}\hat{x}(k) - C_{II}\hat{x}(k-1) + D_{II}\Delta_0 u + E_I y(k) \tag{4–47}$$

$$z = M_I\left[z(k) + r(k) - y(k)\right] + N_I\left(r - y\right) \tag{4–48}$$

where

$$C_{II}^{0} = \begin{bmatrix} C\sum_{i=0}^{1} A^i \\ C\sum_{i=0}^{2} A^i \\ \vdots \\ C\sum_{i=0}^{N_p} A^i \end{bmatrix} \tag{4–49}$$

$$C_{II} = \begin{bmatrix} C\left(\sum_{i=0}^{1} A^i - I\right) \\ C\left(\sum_{i=0}^{2} A^i - I\right) \\ \vdots \\ C\left(\sum_{i=0}^{N_p} A^i - I\right) \end{bmatrix} \tag{4–50}$$

$$D_{II} = \begin{bmatrix} CB & 0 & \dots & 0 & 0 \\ C\sum_{i=0}^{1} A^i & CB & \dots & 0 & \\ \vdots & \vdots & \vdots & \vdots & \\ C\sum_{i=0}^{N_p-1} A^i & C\sum_{i=0}^{N_p-2} A^i & \dots & C\sum_{i=0}^{N_p-N_c+1} A^i & C\sum_{i=0}^{N_p-N_c} A^i \end{bmatrix} \tag{4–51}$$

$$\Delta_0 = \begin{bmatrix} I & 0 & 0 & \dots & 0 \\ -I & I & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & I \end{bmatrix} \tag{4–52}$$

$$\Delta_1 = \begin{bmatrix} I \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4–53}$$

(4–54)

This results in the control law

$$\begin{aligned} u &= (K_{I\Delta} + K_{zI\Delta} N_I) \left[r - C_{II} \left(\hat{x}(k) - \hat{x}(k-1) \right) + D_{II} \Delta_1 u \right] \\ &\quad + K_{zI\Delta} M_I \left[z(k) + r(k) - y(k) \right] \end{aligned} \tag{4–55}$$

$$\tag{4–56}$$

where

$$K_{I\Delta} = \left[(D_{II}\Delta_0)^T \left(N_I^T S_a N_I + Q_a \right) (D_{II}\Delta_0) + R_a \right]^{-1} (D_{II}\Delta_0)^T Q_a \tag{4–57}$$

$$K_{zI\Delta} = \left[(D_{II}\Delta_0)^T \left(N_I^T S_a N_I + Q_a \right) (D_{II}\Delta_0) + R_a \right]^{-1} (D_{II}\Delta_0)^T N_I^T S_a \tag{4–58}$$

$$\tag{4–59}$$

and where

$$\Delta_0 = \tag{4–60}$$

$$\Delta_1 = \tag{4–61}$$

These forms can also be combined, using an incremental state equation with a standard output equation or vice versa. The estimation and control equations for these variants and others that are discussed below are listed in Table 4-1.

4.3.2 Method II: Input-Velocity Cost

An MPC controller can also be configured to perform a predictive control method that utilizes an input-velocity cost. Since input velocities are the goal of the method, it is a natural choice to use an incremental model like the one described in equations (4–45) and (4–45) since they already describe the plant in terms of the input velocity Δu. The construction of the Estimation and Prediction systems proceeds as in Method I yielding the augmented prediction equation

$$y = C_{II} \left[\hat{x}(k) - \hat{x}(k-1) \right] + D_{II} \Delta u + E_I y(k) \tag{4–62}$$

The optimization system then minimizes the performance function

$$J = (r - y)^T Q_a (r - y) + \Delta u^T T_a \Delta u \tag{4–63}$$

which includes the key element of Method II, a penalty on the rate of change of inputs. Minimizing this equation produces the control law

$$\Delta u = K_{II} \left[C_{II} \left[\hat{x}(k) - \hat{x}(k-1) \right] + E_I y(k) \right] \tag{4–64}$$

where

$$K_{II} = \left[D_{II}^T Q_a D_{II} + T_a \right]^{-1} D_{II}^T Q_a \tag{4–65}$$

This approach is fundamentally similar to that discussed for Method I . This similarity might lead one to believe that the velocity-weighting method could be implemented based on non-incremental plant models using the Δ_0 matrix defined in (4–60). However, the simulation study presented in the next section shows that an incremental state equation is required for velocity-weighting to correctly eliminate offset. The use of a non-incremental output equation is still a viable strategy, which is offset-free in some cases.

4.3.3 Method III: Integral States and Velocity Control

An MPC controller can also perform a combination of Methods I and II. The Model, Estimation, and Prediction Systems proceed as described for Method II. If an incremental model is used, the resulting predictor is given by

$$y = C_{II} \left[\hat{x}(k) - \hat{x}(k-1) \right] + D_{II} \Delta u + E_I y(k) \tag{4–66}$$

$$z = M_I \left[z(k) + r(k) - y(k) \right] + N_I (r - y) \tag{4–67}$$

The optimization system uses the performance index

$$J = (r - y)^T Q_a (r - y) + \Delta u^T T_a \Delta u + z^T S_a z \tag{4–68}$$

which include terms for both input velocity and integral states. The minimization of this performance index yields the control law

$$\begin{aligned} u &= (K_{I\Delta} + K_{zI\Delta} N_I) \left[r - C_{II}^0 \hat{x}(k) + C_{II} \hat{x}(k-1) - E_I y(k) \right] \\ &\quad + K_{zI\Delta} M_I \left[z(k) + r(k) - y(k) \right] \end{aligned} \tag{4–69}$$

$$\tag{4–70}$$

where

$$K_{II} = \left[D_{II}^T \left(N_I^T S_a N_I + Q_a \right) D_{II} + R_a \right]^{-1} D_{II}^T Q_a \tag{4–71}$$

$$K_{zII} = \left[D_{II}^T \left(N_I^T S_a N_I + Q_a \right) D_{II} + R_a \right]^{-1} D_{II}^T N_I^T S_a \tag{4–72}$$

$$\tag{4–73}$$

These equations are based on the incremental model equations (4–45) and (4–45). If non-incremental state equations are used, the velocity control mechanism fails to add an integrator, but the integrating states continue to perform normally. In the following section this is demonstrated via the application of a ramp in set-point. If both integrating methods are effective, the controller is able to eliminate offset with respect to a ramp. If only one integrator is effective, the controller is capable of offset-free tracking for step set-point changes but not for ramps. Table 4-1 details which formulations of Method III track ramps and thus contain two integrators.

4.3.4 Method IV: Double Integral States

An alternative method of including two integrators in the MPC controller is two include two sets of explicit integral states. In this case the Model System is augmented to

the form given

$$
\begin{aligned}
x(k+j) &= Ax(k+j-1) + Bu(k+j-1) && (4\text{–}74)\\
z(k+j) &= z(k+j-1) + r(k+j-1) - y(k+j-1) && (4\text{–}75)\\
w(k+j) &= w(k+j-1) + z(k+j-1) && (4\text{–}76)\\
y(k+j) &= Cx(k+j) && (4\text{–}77)\\
& && (4\text{–}78)
\end{aligned}
$$

This leads to the predictor

$$
\begin{aligned}
y &= C_I\hat{x}(k) + D_I u + F_I\hat{d} + E_I\hat{p} && (4\text{–}79)\\
z &= M_I\left[z(k) + r(k) - y(k)\right] + N_I\left(r - y\right) && (4\text{–}80)\\
w &= M_I\left[w(k) + z(k)\right] + N_I w && (4\text{–}81)\\
& && (4\text{–}82)
\end{aligned}
$$

The performance index

$$
J = \left(r - y\right)^T Q_a \left(r - y\right) + u^T R_a u + z^T S_a z + w^T W_a w \qquad (4\text{–}83)
$$

includes terms that weight the double integral states. The minimization of this performance index yields the control law

$$
\begin{aligned}
u &= \left(K_I V + K_{zIV} N_I + K_w N_I^2\right)\left[r - C_I\hat{x}(k)\right] - F_I\hat{d}(k) - E_I\hat{p}(k)\right] && (4\text{–}84)\\
&\quad + \left(K_{zIV} + K_w\right) M_I\left[z(k) + r(k) - y(k)\right] + K_w\left[w(k) + z(k)\right] && \\
& && (4\text{–}85)
\end{aligned}
$$

where

$$K_{IV} = \left[D_I^T \left(N_I^{2^T} W_a N_I^2 + N_I^T S_a N_I + Q_a\right) D_I + R_a\right]^{-1} D_I^T Q_a \quad (4\text{–}86)$$

$$K_{zIV} = \left[D_I^T \left(N_I^{2^T} W_a N_I^2 + N_I^T S_a N_I + Q_a\right) D_I + R_a\right]^{-1} D_I^T N_I^T S_a \quad (4\text{–}87)$$

$$K_w = \left[D_I^T \left(N_I^{2^T} W_a N_I^2 + N_I^T S_a N_I + Q_a\right) D_I + R_a\right]^{-1} D_I^T N_I^{2^T} W_a \quad (4\text{–}88)$$

$$(4\text{–}89)$$

As in Method I, standard or incremental models can be used to describe the plant without an impact on the integrals or the steady state behavior of the system.

4.4 Estimators

Several options for the estimation system are available. If x, d or p are measurable, then they can be used directly. If d or p is not measurable, zero values can be used for $\hat{d}$ or $\hat{p}$. If this is not desirable, or if unmeasured states are unavoidable then an estimator is required. The most lcommon estimators are the ones shown in equations (4–14-4–16). This estimator works effectively for all methods, but the incremental models discussed above suggest the alternative estimator formulation:

$$\hat{x}(k) = (A + I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta u(k-1) \quad (4\text{–}90)$$

$$+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right] \quad (4\text{–}91)$$

$$\hat{d}(k) = \hat{d}(k-1) + L_d\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$$

$$\hat{p}(k) = \hat{p}(k-1) + L_p\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right] \quad (4\text{–}92)$$

$$(4\text{–}93)$$

If desired, the incremental model state estimator can be reduced to the standard form via the substitution

$$v(k) = \begin{bmatrix} \hat{x}(k) \\ \hat{x}(k-1) \end{bmatrix} \quad (4\text{–}94)$$

$$(4\text{–}95)$$

this yields the system

$$\begin{aligned} v(k+1) &= \left(\begin{bmatrix} A+I & -A \\ I & 0 \end{bmatrix} - K_v \begin{bmatrix} C & -C \end{bmatrix} \right) v(k) \\ &\quad + \begin{bmatrix} B \\ 0 \end{bmatrix} \Delta u(k) + K_v \left(y(k) - y(k-1) \right) \end{aligned} \tag{4–96}$$

allowing the gain K_v to be designed by the usual techniques, such as pole placement.

4.5 Simulation Study

The various methods for offset elimination are examined via numerical simulation study. Each is implemented for several different plant models. The results regarding steady state behavior are listed in table 4-1. The first column describes the method used. The second and third columns show whether standard or incremental models are used for the state and output equations respectively. The fourth column lists whether or not state and disturbance estimators are included. When they are not included, the simulations are performed under the assumption that x is directly measurable and $\hat{d}$ and $\hat{p}$ are both zero. The fifth and sixth columns show the final predictor and controller equations. The remaining columns show the steady state behavior. In the tracking column an "offset" entry means that the controller cannot track a step change in set point without offset, a "step" entry means the controller can track a step change without offset, and an entry of "ramp" means the controller can track steps or ramps in set point without steady state offset. The d and p columns list if the control configuration can reject constant final-value disturbances of the types described above as d and p.

The focus of this chapter has is steady state behavior, but for completeness transient behavior must be considered as well. To this end the effect of the controllers on a plant consisting of a model of a stirred tank reactor is examined. Pannocchia and Rawlings [34]

propose the reactor model

$$x(k+1) = \begin{bmatrix} 0.2511 & -3.368 \times 10^{-3} & -7.056 \times 10^{-4} \\ 11.06 & 0.3296 & -2.545 \\ 0 & 0 & 1 \end{bmatrix} x(k) \tag{4–97}$$

$$+ \begin{bmatrix} -5.426 \times 10^{-3} & 1.530 \times 10^{-5} \\ 1.297 & 0.1218 \\ 0 & -6.592 \times 10^{-2} \end{bmatrix} u(k) + d(k) \tag{4–98}$$

$$y(k+1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + p(k+1) \tag{4–99}$$

(4–100)

where the states x are the concentration of reactant, temperature, and liquid level in the reactor, the inputs u are the coolant temperature and the flow rate of material out of the reactor. For the MPC design the horizons are chosen to be $N_p = 12$ and $N_c = 3$. The effects of both set-point changes and disturbances on a variety of controllers is considered. Step changes in set point are applied to both of the controlled outputs and the disturbance

$$d(k) = \left[-1.762 \times 10^{-5} 7.784 \times 10^{-2} 6.592 \times 10^{-2}\right]^T \tag{4–101}$$

is introduced at time, $t = 175$; prior to that the disturbance is zero.

The reactor model is placed in a closed loop and subjected to changes in set point and disturbances. Figure 4.6 shows the response of this plant to a controller formulated according to Method I as described in row 1 of Table 4-1. Offset elimination in both outputs is successfully achieved. Using the weighting matrices

$$Q = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & \end{bmatrix}, R = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}, S = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 1 & \end{bmatrix} \tag{4–102}$$

the dynamic performance shown is archived, if specific performance criteria are desired the weights can be adjusted to archived these. As in the examples used by Pannocchia and Rawlings the second output is not controlled. To this end the weights on both it and its integral are set to zero.

Though state and disturbance estimation are not required for the methods listed here, it can still be worthwhile to demonstrate how to implement an observer. Thus the next simulation is performed using Method II with an incremental model estimator as described in row 23 of Table 4-1. The weighting matrices

$$Q = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & \end{bmatrix}, T = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \tag{4–103}$$

(4–104)

are used as well as the observer gain from equation (4–105).

$$L_x = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \tag{4–105}$$

The results are shown in Figure 4.6. Again the offset is completely eliminated and the dynamic performance can be further adjusted via tuning should a user so desire.

Controllers designed according to methods III and IV also have the ability to track ramps. To demonstrate this, consider the plant model

$$x(k+1) = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.6 \end{bmatrix} x(k) \tag{4–106}$$

$$+ \begin{bmatrix} 0.5 & 0 \\ 0 & 0.4 \\ 0.25 & 0 \\ 0 & 0.6 \end{bmatrix} u(k) + d(k) \tag{4–107}$$

$$y(k+1) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} + p(k+1) \tag{4–108}$$

$$\tag{4–109}$$

Figure 4.6 shows this plant used with a method III controller formulated according to row 34 of Table 4-1. This controller uses the weights

$$Q = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}, T = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}, S = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.3 \end{bmatrix} \tag{4–110}$$

and the horizons $N_p = 5$ and $N_c = 4$. Figure 4.6a demonstrates the response to a step in set point and in disturbance. Figure 4.6b shows the response to a ramp in set point. In either case the steady-state offset is eliminated. Figure 4.6 shows a similar set of responses for the Method IV controller described on row 39 of Table 4-1. This controller is tuned using the weights

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}, S = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, W = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \tag{4–111}$$

It is apparent from Figure 4.6 that the Method IV controller successfully eliminates offset from both step and ramp set-point changes as well as step changes in disturbance.

4.6 Conclusion

Four different methods for offset elimination in predictive control are suggested. Simulation results demonstrate that all are capable of eliminating steady state offset from step changes in input or constant final-value disturbances. The specific forms which are effective are outline in Table 4-1 which can be summarized by the following observations. Explicit integral states, as used in methods I, III and IV, introduce an integrator regardless of the model form used. Input-velocity weighting control, as used in methods II and III introduces an integrator in the tracking behavior only when paired with an incremental state equation, and ensures offset free performance in the disturbance rejection problem only when paired with an incremental output equation.

Table 4-1. Equations and simulation results for various MPC methods

Method	State	Output	Estimator	Tracking	Disturbance	Equations
1	I	Standard	Standard	None	Step	Both	$\hat{y}= C_I\hat{x}(k) + D_I u$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $u= \left(k_I^T + k_z^T N_I\right)\left[r - C_I\hat{x}(k)\right]$ $+k_z^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
2	I	Standard	Standard	Standard	Step	Both	$\hat{y}= C_I\hat{x}(k) + D_I u + F_I\hat{d}(k) + E_I\hat{p}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\hat{x}(k+1)= A\hat{x}(k) + Bu(k) + \hat{d}(k) + L_x\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $u= \left(k_I^T + k_z^T N_I\right)\left[r - C_I\hat{x}(k) - F_I\hat{d}(k) + E_I\hat{p}(k)\right]$ $+k_z^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
3	I	Standard	Standard	Incremental	Step	Both	$\hat{y}= C_I\hat{x}(k) + D_I u + F_I\hat{d}(k) + E_I\hat{p}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta u(k-1)$ $+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $u= \left(k_I^T + k_z^T N_I\right)\left[r - C_I\hat{x}(k) - F_I\hat{d}(k) + E_I\hat{p}(k)\right]$ $+k_z^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
4	I	Incremental	Standard	None	Step	Both	$\hat{y}= C_{II}^0\hat{x}(k) - C_{II}\hat{x}(k-1) - D_{II}\Delta_1 u(k-1) + D_{II}\Delta_0 u$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $u= \left(k_{I\Delta}^T + k_{z\Delta}^T N_I\right) \times$ $\left[r - C_{II}^0\hat{x}(k) + C_{II}\hat{x}(k-1) + D_{II}\Delta_1 u(k-1)\right]$ $+k_{z\Delta}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
5	I	Incremental	Standard	Standard	Step	Both	$\hat{y}= C_{II}^0\hat{x}(k) - C_{II}\hat{x}(k-1) - D_{II}\Delta_1 u(k-1) + D_{II}\Delta_0 u + E_I\hat{p}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\hat{x}(k+1)= A\hat{x}(k) + Bu(k) + \hat{d}(k) + L_x\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $u= \left(k_{I\Delta}^T + k_{z\Delta}^T N_I\right) \times$ $\left[r - C_{II}^0\hat{x}(k) + C_{II}\hat{x}(k-1) + D_{II}\Delta_1 u(k-1) - E_I\hat{p}(k)\right]$ $+k_{z\Delta}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
6	I	Incremental	Standard	Incremental	Step	Both	$\hat{y}= C_{II}^0\hat{x}(k) - C_{II}\hat{x}(k-1) - D_{II}\Delta_1 u(k-1) + D_{II}\Delta_0 u + E_I\hat{p}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta u(k-1)$ $+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $u= \left(k_{I\Delta}^T + k_{z\Delta}^T N_I\right) \times$ $\left[r - C_{II}^0\hat{x}(k) + C_{II}\hat{x}(k-1) + D_{II}\Delta_1 u(k-1) - E_I\hat{p}(k)\right]$ $+k_{z\Delta}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
7	I	Standard	Incremental	None	Step	Both	$\hat{y}= C_I\hat{x}(k) + D_Iu + E_Iy(k) - E_IC_Ix(k) + F_I\hat{d}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $u= \left(k_I^T + k_z^T N_I\right)\left[r - C_I\hat{x}(k) - E_Iy(k) + E_IC_Ix(k) - F_I\hat{d}(k)\right]$ $+k_z^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
8	I	Standard	Incremental	Standard	Step	Both	$\hat{y}= C_I\hat{x}(k) + D_Iu + E_Iy(k) - E_IC_Ix(k) + F_I\hat{d}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\hat{x}(k+1)= A\hat{x}(k) + Bu(k) + \hat{d}(k) + L_x\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $u= \left(k_I^T + k_z^T N_I\right)\left[r - C_I\hat{x}(k) - E_Iy(k) + E_IC_Ix(k) - F_I\hat{d}(k)\right]$ $+k_z^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
9	I	Standard	Incremental	Incremental	Step	Both	$\hat{y}= C_I\hat{x}(k) + D_Iu + E_Iy(k) - E_IC_Ix(k) + F_I\hat{d}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta u(k-1)$ $+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $u= \left(k_I^T + k_z^T N_I\right)\left[r - C_I\hat{x}(k) - E_Iy(k) + E_IC_Ix(k) - F_I\hat{d}(k)\right]$ $+k_z^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$

Table 4-1 Continued

Method		State	Output	Estimator	Tracking	Disturbance	Equations
10	I	Incremental	Incremental	None	Step	Both	$\hat{y}= C_{II}\left(\hat{x}(k)-\hat{x}(k-1)\right)-D_{II}\Delta_1 u(k-1)+D_{II}\Delta_0 u$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right]+N_I\left[r-\hat{y}\right]$ $u= \left(k_{I\Delta}^T+k_{z\Delta}^T N_I\right)\left[r-C_{II}\left(\hat{x}(k)+\hat{x}(k-1)\right)+D_{II}\Delta_1 u(k-1)\right]$ $+k_{z\Delta}^T M_I\left[z(k)+r(k)-\hat{y}(k)\right]$
11	I	Incremental	Incremental	Standard	Step	Both	$\hat{y}= C_{II}\left(\hat{x}(k)-\hat{x}(k-1)\right)-D_{II}\Delta_1 u(k-1)+D_{II}\Delta_0 u$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right]+N_I\left[r-\hat{y}\right]$ $\hat{x}(k+1)= A\hat{x}(k)+Bu(k)+\hat{d}(k)+L_x\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k)+L_d\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k)+L_p\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $u= \left(k_{I\Delta}^T+k_{z\Delta}^T N_I\right)\left[r-C_{II}\left(\hat{x}(k)+\hat{x}(k-1)\right)+D_{II}\Delta_1 u(k-1)\right]$ $+k_{z\Delta}^T M_I\left[z(k)+r(k)-\hat{y}(k)\right]$
12	I	Incremental	Incremental	Incremental	Step	Both	$\hat{y}= C_{II}\left(\hat{x}(k)-\hat{x}(k-1)\right)-D_{II}\Delta_1 u(k-1)+D_{II}\Delta_0 u$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right]+N_I\left[r-\hat{y}\right]$ $\hat{x}(k+1)= \left(A+I\right)\hat{x}(k)-A\hat{x}(k-1)+B\Delta u(k-1)$ $+L_x\left[y(k)-y(k-1)-C\hat{x}(k)+C\hat{x}(k-1)\right]$ $u= \left(k_{I\Delta}^T+k_{z\Delta}^T N_I\right)\left[r-C_{II}\left(\hat{x}(k)+\hat{x}(k-1)\right)+D_{II}\Delta_1 u(k-1)\right]$ $+k_{z\Delta}^T M_I\left[z(k)+r(k)-\hat{y}(k)\right]$
13	II	Standard	Standard	None	Offset	None	$\hat{y}= C_I\hat{x}(k)+D_I\Delta_0^{-1}\Delta u+D_I\Delta_2 u(k-1)$ $\Delta u= \left(k_{II\Delta}^T+k_{z\Delta}^T N_I\right)\left[r-C_{II}\hat{x}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
14	II	Standard	Standard	Standard	Offset	None	$\hat{y}= C_I\hat{x}(k) + D_I\Delta_0^{-1}\Delta u + D_I\Delta_2 u(k-1) + F_I\hat{d}(k) + E_I\hat{p}(k)$ $\hat{x}(k+1)= A\hat{x}(k) + Bu(k) + \hat{d}(k) + L_x\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\Delta u= \left(k_{II\Delta}^T + k_{z\Delta}^T N_I\right)\left[r - C_{II}\hat{x}(k) - F_{II}\hat{d}(k) + E_{II}\hat{p}(k)\right]$
15	II	Standard	Standard	Incremental	Offset	None	$\hat{y}= C_I\hat{x}(k) + D_I\Delta_0^{-1}\Delta u + D_I\Delta_2 u(k-1) + F_I\hat{d}(k) + E_I\hat{p}(k)$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta u(k-1)$ $+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\Delta u= \left(k_{II\Delta}^T + k_z^T N_I\right)\left[r - C_I\hat{x}(k) - F_I\hat{d}(k) + E_I\hat{p}(k)\right]$
16	II	Incremental	Standard	None	Step	d Only	$\hat{y}= C_I^0\hat{x}(k) - C_I\hat{x}(k-1) - D_I\Delta_1\Delta u(k-1) + D_I\Delta_0\Delta u$ $\Delta u= \left(k_{II}^T + k_{z\Delta}^T N_I\right)\left[r - C_{II}^0\hat{x}(k) + C_{II}\hat{x}(k-1)\right]$
17	II	Incremental	Standard	Standard	Step	Both	$\hat{y}= C_{II}^0\hat{x}(k) - C_{II}\hat{x}(k-1) + D_{II}\Delta_0\Delta u + E_I\hat{p}(k)$ $\hat{x}(k+1)= A\hat{x}(k) + B\Delta u(k) + \hat{d}(k) + L_x\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\Delta u= \left(k_{II}^T + k_{z\Delta}^T N_I\right)\left[r - C_{II}^0\hat{x}(k) + C_{II}\hat{x}(k-1) - E_I\hat{p}(k)\right]$

Table 4-1 Continued

	Method	State	Output	Estimator	Tracking	Disturbance	Equations
18	II	Incremental	Standard	Incremental	Step	Both	$\hat{y}= C_{II}^{0}\hat{x}(k) - C_{II}\hat{x}(k-1) + D_{II}\Delta_0\Delta u + E_I\hat{p}(k)$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta\Delta u(k-1)$ $+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\Delta u= \left(k_{II}^{T} + k_{z\Delta}^{T}N_I\right)\left[r - C_{II}^{0}\hat{x}(k) + C_{II}\hat{x}(k-1) - E_I\hat{p}(k)\right]$
19	II	Standard	Incremental	None	Offset	None	$\hat{y}= C_I\hat{x}(k) + D_I\Delta u + E_I y(k) - E_I C_I x(k) + F_I\hat{d}(k)$ $\Delta u= \left(k_{II\Delta}^{T} + k_{z}^{T}N_I\right)\left[r - C_I\hat{x}(k) - E_I y(k) + E_I C_I x(k) - F_I\hat{d}(k)\right]$ $+k_{z}^{T}M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
20	II	Standard	Incremental	Standard	Offset	None	$\hat{y}= C_I\hat{x}(k) + D_I\Delta u + E_I y(k) - E_I C_I x(k) + F_I\hat{d}(k)$ $\hat{x}(k+1)= A\hat{x}(k) + B\Delta u(k) + \hat{d}(k) + L_x\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\Delta u= \left(k_{II\Delta}^{T} + k_{z}^{T}N_I\right)\left[r - C_I\hat{x}(k) - E_I y(k) + E_I C_I x(k) - F_I\hat{d}(k)\right]$
21	II	Standard	Incremental	Incremental	Offset	None	$\hat{y}= C_I\hat{x}(k) + D_I\Delta u + E_I y(k) - E_I C_I x(k) + F_I\hat{d}(k)$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta\Delta u(k-1)$ $+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\Delta u= \left(k_{II\Delta}^{T} + k_{z}^{T}N_I\right)\left[r - C_I\hat{x}(k) - E_I y(k) + E_I C_I x(k) - F_I\hat{d}(k)\right]$
22	II	Incremental	Incremental	None	Step	Both	$\hat{y}= C_{II}\left(\hat{x}(k) - \hat{x}(k-1)\right) + D_{II}\Delta_0\Delta u$ $\Delta u= \left(k_{II}^{T} + k_{z\Delta}^{T}N_I\right)\left[r - C_{II}\left(\hat{x}(k) + \hat{x}(k-1)\right)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
23	II	Incremental	Incremental	Standard	Step	Both	$\hat{y}= C_{II}\left(\hat{x}(k)-\hat{x}(k-1)\right)+D_{II}\Delta_0\Delta u$ $\hat{x}(k+1)= A\hat{x}(k)+B\Delta u(k)+\hat{d}(k)+L_x\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k)+L_d\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k)+L_p\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\Delta u= \left(k_{II}^T+k_{z\Delta}^T N_I\right)\left[r-C_{II}\left(\hat{x}(k)+\hat{x}(k-1)\right)\right]$
24	II	Incremental	Incremental	Incremental	Step	Both	$\hat{y}= C_{II}\left(\hat{x}(k)-\hat{x}(k-1)\right)+D_{II}\Delta_0\Delta u$ $\hat{x}(k+1)= \left(A+I\right)\hat{x}(k)-A\hat{x}(k-1)+B\Delta\Delta u(k-1)$ $+L_x\left[y(k)-y(k-1)-C\hat{x}(k)+C\hat{x}(k-1)\right]$ $\Delta u= \left(k_{II}^T+k_{z\Delta}^T N_I\right)\left[r-C_{II}\left(\hat{x}(k)+\hat{x}(k-1)\right)\right]$
25	III	Standard	Standard	None	Step	None	$\hat{y}= C_I\hat{x}(k)+D_I\Delta_0^{-1}\Delta u+D_I\Delta_2 u(k-1)$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right]+N_I\left[r-\hat{y}\right]$ $\Delta u= \left(k_{II\Delta}^T+k_{z\Delta}^T N_I\right)\left[r-C_I\hat{x}(k)\right]$ $+k_{z\Delta}^T M_I\left[z(k)+r(k)-\hat{y}(k)\right]$
26	III	Standard	Standard	Standard	Step	None	$\hat{y}= C_I\hat{x}(k)+D_I\Delta_0^{-1}\Delta u+D_I\Delta_2 u(k-1)+F_I\hat{d}(k)+E_I\hat{p}(k)$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right]+N_I\left[r-\hat{y}\right]$ $\hat{x}(k+1)= A\hat{x}(k)+Bu(k)+\hat{d}(k)+L_x\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k)+L_d\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k)+L_p\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\Delta u= \left(k_{II\Delta}^T+k_{z\Delta}^T N_I\right)\left[r-C_I\hat{x}(k)-F_I\hat{d}(k)+E_I\hat{p}(k)\right]$ $+k_{z\Delta}^T M_I\left[z(k)+r(k)-\hat{y}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
27	III	Standard	Standard	Incremental	Step	None	$\hat{y}= C_I\hat{x}(k) + D_I\Delta_0^{-1}\Delta u + D_I\Delta_2 u(k-1) + F_I\hat{d}(k) + E_I\hat{p}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta u(k-1)$ $+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\Delta u= \left(k_{II\Delta}^T + k_z^T N_I\right)\left[r - C_I\hat{x}(k) - F_I\hat{d}(k) + E_I\hat{p}(k)\right]$ $+k_z^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
28	III	Incremental	Standard	None	Ramp	Both	$\hat{y}= C_{II}^0\hat{x}(k) - C_{II}\hat{x}(k-1) + D_{II}\Delta_0\Delta u$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\Delta u= \left(k_{II}^T + k_{z\Delta}^T N_I\right)\left[r - C_{II}^0\hat{x}(k) + C_{II}\hat{x}(k-1)\right]$ $+k_{z\Delta}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
29	III	Incremental	Standard	Standard	Ramp	Both	$\hat{y}= C_{II}^0\hat{x}(k) - C_{II}\hat{x}(k-1) + D_{II}\Delta_0\Delta u + E_I\hat{p}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\hat{x}(k+1)= A\hat{x}(k) + B\Delta u(k) + \hat{d}(k) + L_x\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\Delta u= \left(k_{II}^T + k_{z\Delta}^T N_I\right)\left[r - C_{II}^0\hat{x}(k) + C_{II}\hat{x}(k-1) - E_I\hat{p}(k)\right]$ $+k_{z\Delta}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
30	III	Incremental	Standard	Incremental	Ramp	Both	$\hat{y}= C_{II}^{0}\hat{x}(k) - C_{II}\hat{x}(k-1) + D_{II}\Delta_0\Delta u + E_I\hat{p}(k)$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right] + N_I\left[r-\hat{y}\right]$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta\Delta u(k-1)$ $+L_x\left[y(k)-y(k-1)-C\hat{x}(k)+C\hat{x}(k-1)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k)-y(k-1)-C\hat{x}(k)+C\hat{x}(k-1)\right]$ $\Delta u= \left(k_{II}^{T} + k_{z\Delta}^{T}N_I\right)\left[r - C_{II}^{0}\hat{x}(k) + C_{II}\hat{x}(k-1) - E_I\hat{p}(k)\right]$ $+k_{z\Delta}^{T}M_I\left[z(k)+r(k)-\hat{y}(k)\right]$
31	III	Standard	Incremental	None	Step	None	$\hat{y}= C_I\hat{x}(k) + D_I\Delta u + E_I y(k) - E_I C_I x(k) + F_I\hat{d}(k)$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right] + N_I\left[r-\hat{y}\right]$ $\Delta u= \left(k_{II\Delta}^{T} + k_{z}^{T}N_I\right)\left[r - C_I\hat{x}(k) - E_I y(k) + E_I C_I x(k) - F_I\hat{d}(k)\right]$ $+k_{z}^{T}M_I\left[z(k)+r(k)-\hat{y}(k)\right]$
32	III	Standard	Incremental	Standard	Step	None	$\hat{y}= C_I\hat{x}(k) + D_I\Delta u + E_I y(k) - E_I C_I x(k) + F_I\hat{d}(k)$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right] + N_I\left[r-\hat{y}\right]$ $\hat{x}(k+1)= A\hat{x}(k) + B\Delta u(k) + \hat{d}(k) + L_x\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\Delta u= \left(k_{II\Delta}^{T} + k_{z}^{T}N_I\right)\left[r - C_I\hat{x}(k) - E_I y(k) + E_I C_I x(k) - F_I\hat{d}(k)\right]$ $+k_{z}^{T}M_I\left[z(k)+r(k)-\hat{y}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
33	III	Standard	Incremental	Incremental	Step	None	$\hat{y}= C_I\hat{x}(k) + D_I\Delta u + E_I y(k) - E_I C_I x(k) + F_I\hat{d}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\hat{x}(k+1)= \left(A + I\right)\hat{x}(k) - A\hat{x}(k-1) + B\Delta\Delta u(k-1)$ $+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\Delta u= \left(k_{II\Delta}^T + k_z^T N_I\right)\left[r - C_I\hat{x}(k) - E_I y(k) + E_I C_I x(k) - F_I\hat{d}(k)\right]$ $+k_z^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
34	III	Incremental	Incremental	None	Ramp	Both	$\hat{y}= C_{II}\left(\hat{x}(k) - \hat{x}(k-1)\right) + D_{II}\Delta_0\Delta u$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\Delta u= \left(k_{II}^T + k_{z\Delta}^T N_I\right)\left[r - C_{II}\left(\hat{x}(k) + \hat{x}(k-1)\right)\right]$ $+k_{z\Delta}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
35	III	Incremental	Incremental	Standard	Ramp	Both	$\hat{y}= C_{II}\left(\hat{x}(k) - \hat{x}(k-1)\right) + D_{II}\Delta_0\Delta u$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $\hat{x}(k+1)= A\hat{x}(k) + B\Delta u(k) + \hat{d}(k) + L_x\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\Delta u= \left(k_{II}^T + k_{z\Delta}^T N_I\right)\left[r - C_{II}\left(\hat{x}(k) + \hat{x}(k-1)\right)\right]$ $+k_{z\Delta}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
36	III	Incremental	Incremental	Incremental	Ramp	Both	$\hat{y}= C_{II}\left(\hat{x}(k)-\hat{x}(k-1)\right)+D_{II}\Delta_0\Delta u$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right]+N_I\left[r-\hat{y}\right]$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k)-A\hat{x}(k-1)+B\Delta\Delta u(k-1)$ $+L_x\left[y(k)-y(k-1)-C\hat{x}(k)+C\hat{x}(k-1)\right]$ $\Delta u= \left(k_{II}^T+k_{z\Delta}^T N_I\right)\left[r-C_{II}\left(\hat{x}(k)+\hat{x}(k-1)\right)\right]$ $+k_{z\Delta}^T M_I\left[z(k)+r(k)-\hat{y}(k)\right]$
37	IV	Standard	Standard	None	Ramp	Both	$\hat{y}= C_I\hat{x}(k)+D_I u$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right]+N_I\left[r-\hat{y}\right]$ $w= M_I\left[w(k)+z(k)\right]+N_I z$ $u= \left(k_{IV}^T+k_{zIV}^T N_I+k_w^T N_I^2\right)\left[r-C_I\hat{x}(k)\right]$ $+k_{zIV}^T M_I\left[z(k)+r(k)-\hat{y}(k)\right]$
38	IV	Standard	Standard	Standard	Ramp	Both	$\hat{y}= C_I\hat{x}(k)+D_I u+F_I\hat{d}(k)+E_I\hat{p}(k)$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right]+N_I\left[r-\hat{y}\right]$ $w= M_I\left[w(k)+z(k)\right]+N_I z$ $\hat{x}(k+1)= A\hat{x}(k)+Bu(k)+\hat{d}(k)+L_x\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k)+L_d\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k)+L_p\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $u= \left(k_{IV}^T+k_{zIV}^T N_I+k_w^T N_I^2\right)\left[r-C_I\hat{x}(k)-F_I\hat{d}(k)+E_I\hat{p}(k)\right]$ $+k_{zIV}^T M_I\left[z(k)+r(k)-\hat{y}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
39	IV	Standard	Standard	Incremental	Ramp	Both	$\hat{y}= C_I\hat{x}(k) + D_I u + F_I\hat{d}(k) + E_I\hat{p}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $w= M_I\left[w(k) + z(k)\right] + N_I z$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta u(k-1)$ $+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $u= \left(k_{IV}^T + k_{zIV}^T N_I + k_w^T N_I^2\right)\left[r - C_I\hat{x}(k) - F_I\hat{d}(k) + E_I\hat{p}(k)\right]$ $+k_{zIV}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
40	IV	Incremental	Standard	None	Ramp	Both	$\hat{y}= C_{II}^0\hat{x}(k) - C_{II}\hat{x}(k-1) - D_{II}\Delta_1 u(k-1) + D_{II}\Delta_0 u$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $w= M_I\left[w(k) + z(k)\right] + N_I z$ $u= \left(k_{IV\Delta}^T + k_{zIV\Delta}^T N_I + k_w^T N_I^2\right)\times$ $\left[r - C_{II}^0\hat{x}(k) + C_{II}\hat{x}(k-1) + D_{II}\Delta_1 u(k-1)\right]$ $+k_{zIV\Delta}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
41	IV	Incremental	Standard	Standard	Ramp	Both	$\hat{y}= C_{II}^{0}\hat{x}(k) - C_{II}\hat{x}(k-1) - D_{II}\Delta_1 u(k-1) + D_{II}\Delta_0 u + E_I\hat{p}(k)$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right] + N_I\left[r-\hat{y}\right]$ $w= M_I\left[w(k)+z(k)\right] + N_I z$ $\hat{x}(k+1)= A\hat{x}(k) + Bu(k) + \hat{d}(k) + L_x\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $u= \left(k_{IV\Delta}^{T} + k_{zIV\Delta}^{T}N_I + k_w^{T}N_I^{2}\right)\times$ $\left[r - C_{II}^{0}\hat{x}(k) + C_{II}\hat{x}(k-1) + D_{II}\Delta_1 u(k-1) - E_I\hat{p}(k)\right]$ $+k_{zIV\Delta}^{T}M_I\left[z(k)+r(k)-\hat{y}(k)\right]$
42	IV	Incremental	Standard	Incremental	Ramp	Both	$\hat{y}= C_{II}^{0}\hat{x}(k) - C_{II}\hat{x}(k-1) - D_{II}\Delta_1 u(k-1) + D_{II}\Delta_0 u + E_I\hat{p}(k)$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right] + N_I\left[r-\hat{y}\right]$ $w= M_I\left[w(k)+z(k)\right] + N_I z$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta u(k-1)$ $+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $u= \left(k_{IV\Delta}^{T} + k_{zIV\Delta}^{T}N_I + k_w^{T}N_I^{2}\right)\times$ $\left[r - C_{II}^{0}\hat{x}(k) + C_{II}\hat{x}(k-1) + D_{II}\Delta_1 u(k-1) - E_I\hat{p}(k)\right]$ $+k_{zIV\Delta}^{T}M_I\left[z(k)+r(k)-\hat{y}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
43	IV	Standard	Incremental	None	Ramp	Both	$\hat{y}= C_I\hat{x}(k) + D_Iu + E_Iy(k) - E_IC_Ix(k) + F_I\hat{d}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $w= M_I\left[w(k) + z(k)\right] + N_Iz$ $u= \left(k_{IV}^T + k_{zIV}^T N_I + k_w^T N_I^2\right) \times$ $\left[r - C_I\hat{x}(k) - E_Iy(k) + E_IC_Ix(k) - F_I\hat{d}(k)\right]$ $+k_{zIV}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
44	IV	Standard	Incremental	Standard	Ramp	Both	$\hat{y}= C_I\hat{x}(k) + D_Iu + E_Iy(k) - E_IC_Ix(k) + F_I\hat{d}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $w= M_I\left[w(k) + z(k)\right] + N_Iz$ $\hat{x}(k+1)= A\hat{x}(k) + Bu(k) + \hat{d}(k) + L_x\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k) + L_p\left[y(k) - C\hat{x}(k) - \hat{p}(k)\right]$ $u= \left(k_{IV}^T + k_{zIV}^T N_I + k_w^T N_I^2\right) \times$ $\left[r - C_I\hat{x}(k) - E_Iy(k) + E_IC_Ix(k) - F_I\hat{d}(k)\right]$ $+k_{zIV}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
45	IV	Standard	Incremental	Incremental	Ramp	Both	$\hat{y}= C_I\hat{x}(k) + D_I u + E_I y(k) - E_I C_I x(k) + F_I\hat{d}(k)$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $w= M_I\left[w(k) + z(k)\right] + N_I z$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k) - A\hat{x}(k-1) + B\Delta u(k-1)$ $+L_x\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $\hat{d}(k+1)= \hat{d}(k) + L_d\left[y(k) - y(k-1) - C\hat{x}(k) + C\hat{x}(k-1)\right]$ $u= \left(k_{IV}^T + k_{zIV}^T N_I + k_w^T N_I^2\right)\times$ $\left[r - C_I\hat{x}(k) - E_I y(k) + E_I C_I x(k) - F_I\hat{d}(k)\right]$ $+k_{zIV}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$
46	IV	Incremental	Incremental	None	Ramp	Both	$\hat{y}= C_{II}\left(\hat{x}(k) - \hat{x}(k-1)\right) - D_{II}\Delta_1 u(k-1) + D_{II}\Delta_0 u$ $z= M_I\left[z(k) + r(k) - \hat{y}(k)\right] + N_I\left[r - \hat{y}\right]$ $w= M_I\left[w(k) + z(k)\right] + N_I z$ $u= \left(k_{IV\Delta}^T + k_{zIV\Delta}^T N_I + k_w^T N_I^2\right)\times$ $\left[r - C_{II}\left(\hat{x}(k) + \hat{x}(k-1)\right) + D_{II}\Delta_1 u(k-1)\right]$ $+k_{zIV\Delta}^T M_I\left[z(k) + r(k) - \hat{y}(k)\right]$

Table 4-1 Continued

Method	State	Output	Estimator	Tracking	Disturbance	Equations
47	IV	Incremental	Incremental	Standard	Ramp	Both	$\hat{y}= C_{II}\left(\hat{x}(k)-\hat{x}(k-1)\right)-D_{II}\Delta_1 u(k-1)+D_{II}\Delta_0 u$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right]+N_I\left[r-\hat{y}\right]$ $w= M_I\left[w(k)+z(k)\right]+N_I z$ $\hat{x}(k+1)= A\hat{x}(k)+Bu(k)+\hat{d}(k)+L_x\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\hat{d}(k+1)= \hat{d}(k)+L_d\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $\hat{p}(k+1)= \hat{p}(k)+L_p\left[y(k)-C\hat{x}(k)-\hat{p}(k)\right]$ $u= \left(k_{IV\Delta}^T+k_{zIV\Delta}^T N_I+k_w^T N_I^2\right)\times$ $\left[r-C_{II}\left(\hat{x}(k)+\hat{x}(k-1)\right)+D_{II}\Delta_1 u(k-1)\right]$ $+k_{zIV\Delta}^T M_I\left[z(k)+r(k)-\hat{y}(k)\right]$
48	IV	Incremental	Incremental	Incremental	Ramp	Both	$\hat{y}= C_{II}\left(\hat{x}(k)-\hat{x}(k-1)\right)-D_{II}\Delta_1 u(k-1)+D_{II}\Delta_0 u$ $z= M_I\left[z(k)+r(k)-\hat{y}(k)\right]+N_I\left[r-\hat{y}\right]$ $w= M_I\left[w(k)+z(k)\right]+N_I z$ $\hat{x}(k+1)= (A+I)\,\hat{x}(k)-A\hat{x}(k-1)+B\Delta u(k-1)$ $+L_x\left[y(k)-y(k-1)-C\hat{x}(k)+C\hat{x}(k-1)\right]$ $u= \left(k_{IV\Delta}^T+k_{zIV\Delta}^T N_I+k_w^T N_I^2\right)\times$ $\left[r-C_{II}\left(\hat{x}(k)+\hat{x}(k-1)\right)+D_{II}\Delta_1 u(k-1)\right]$ $+k_{zIV\Delta}^T M_I\left[z(k)+r(k)-\hat{y}(k)\right]$

Table 4-1 Continued

Method State	Output	Estimator	Tracking Disturbance Equations

$$C_I = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix} \quad D_I = \begin{bmatrix} CB & 0 & \dots & 0 & 0 \\ CAB & CB & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{N_c-1}B & CA^{N_c-2}B & \dots & CAB & B \\ CA^{N_c}B & CA^{N_c-1}B & \dots & CA^2B & \sum_{i=0}^{1} CA^iB \\ CA^{N_c+1}B & CA^{N_c}B & \dots & CA^3B & \sum_{i=0}^{2} CA^iB \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & \dots & CA^{Np-Nc+1}B & \sum_{i=0}^{Np-Nc} CA^iB \end{bmatrix} \quad F_I = \begin{bmatrix} C \\ \sum_{i=0}^{1} CA^i \\ \vdots \\ \sum_{i=0}^{N_p-1} CA^i \end{bmatrix} \quad E_I = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}$$

$$C_{II}^0 = \begin{bmatrix} C\sum_{i=0}^{1} A^i \\ C\sum_{i=0}^{2} A^i \\ \vdots \\ C\sum_{i=0}^{N_p} A^i \end{bmatrix} \quad C_{II} = \begin{bmatrix} C\left(\sum_{i=0}^{1} A^i - I\right) \\ C\left(\sum_{i=0}^{2} A^i - I\right) \\ \vdots \\ C\left(\sum_{i=0}^{N_p} A^i - I\right) \end{bmatrix} \quad D_{II} = \begin{bmatrix} CB & 0 & \dots & 0 & 0 \\ C\sum_{i=0}^{1} A^i & CB & \dots & 0 & \\ \vdots & \vdots & \vdots & \vdots & \\ C\sum_{i=0}^{N_p-1} A^i & C\sum_{i=0}^{N_p-2} A^i & \dots & C\sum_{i=0}^{N_p-N_c+1} A^i & C\sum_{i=0}^{N_p-N_c} A^i \end{bmatrix}$$

$$M_I = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} \quad N_I = \begin{bmatrix} 0 & 0 & \dots & 0 \\ I & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ I & I & \dots & I \end{bmatrix} \quad \Delta_0 = \begin{bmatrix} I & 0 & 0 & \dots & 0 \\ -I & I & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & I \end{bmatrix} \quad \Delta_1 = \begin{bmatrix} I \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

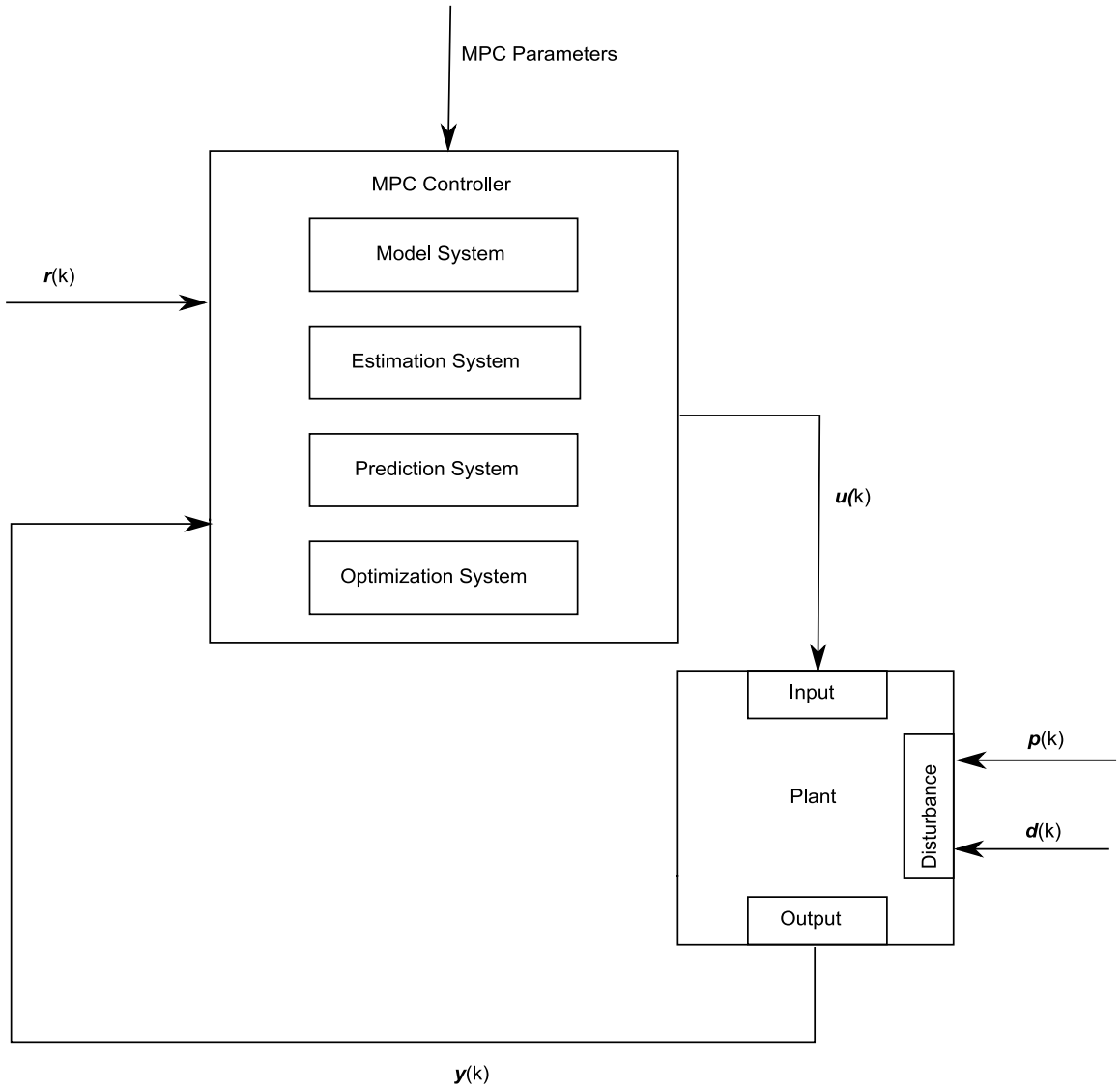

Figure 4-1. A block diagram illustrating the various elements of an MPC system.

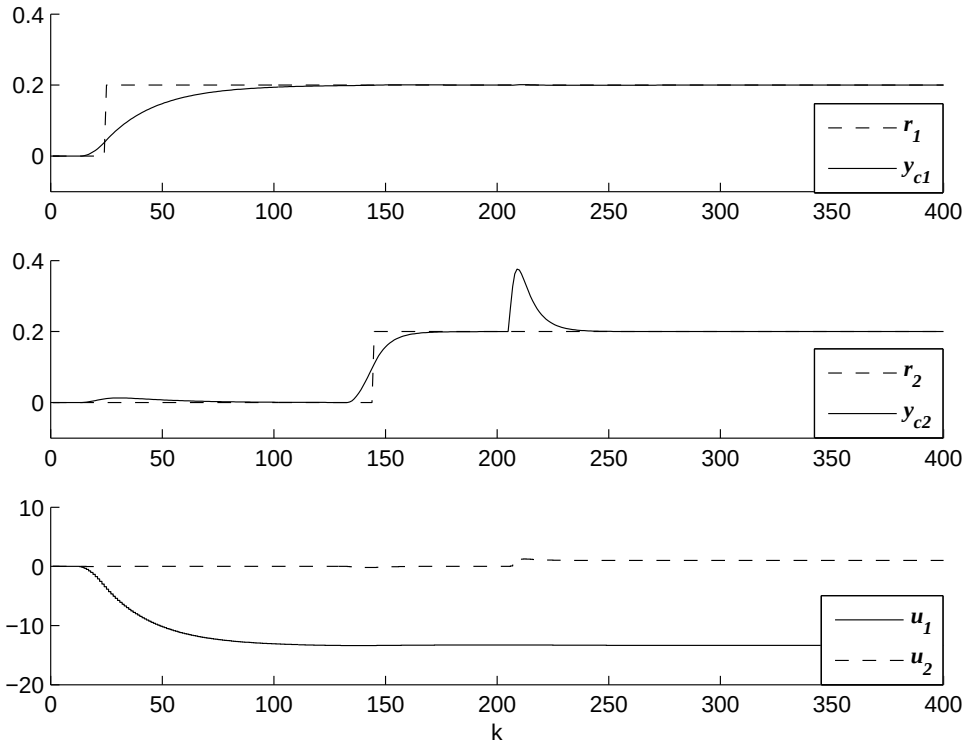

Figure 4-2. The results of a Method I MPC design applied to a simulated chemical reactor plant.

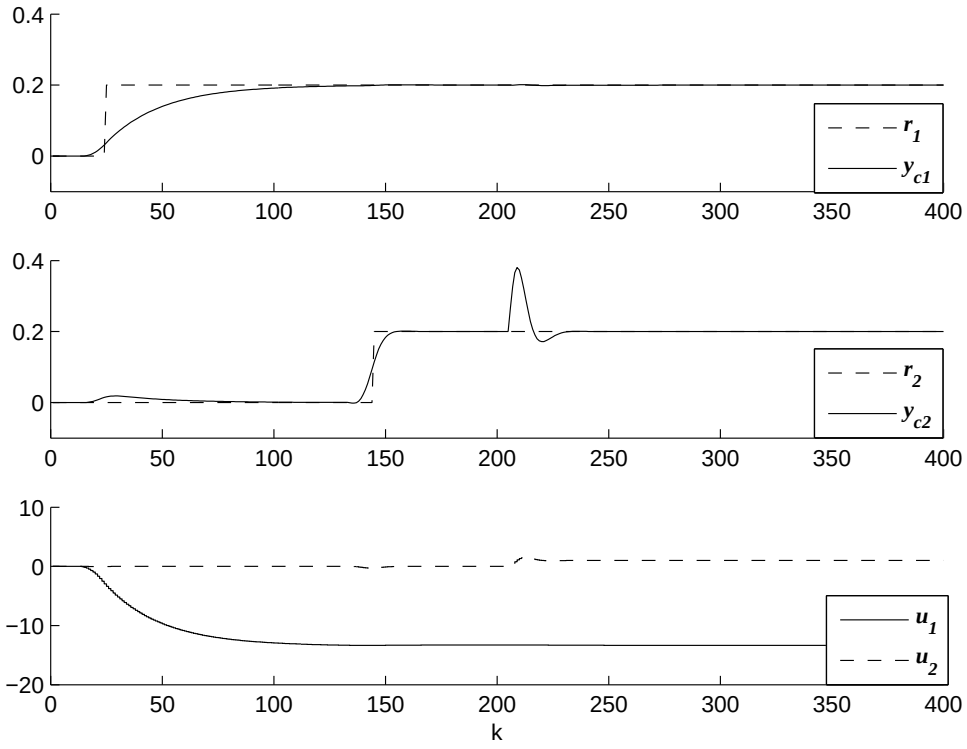

Figure 4-3. The results of a Method II MPC design applied to a simulated chemical reactor plant.

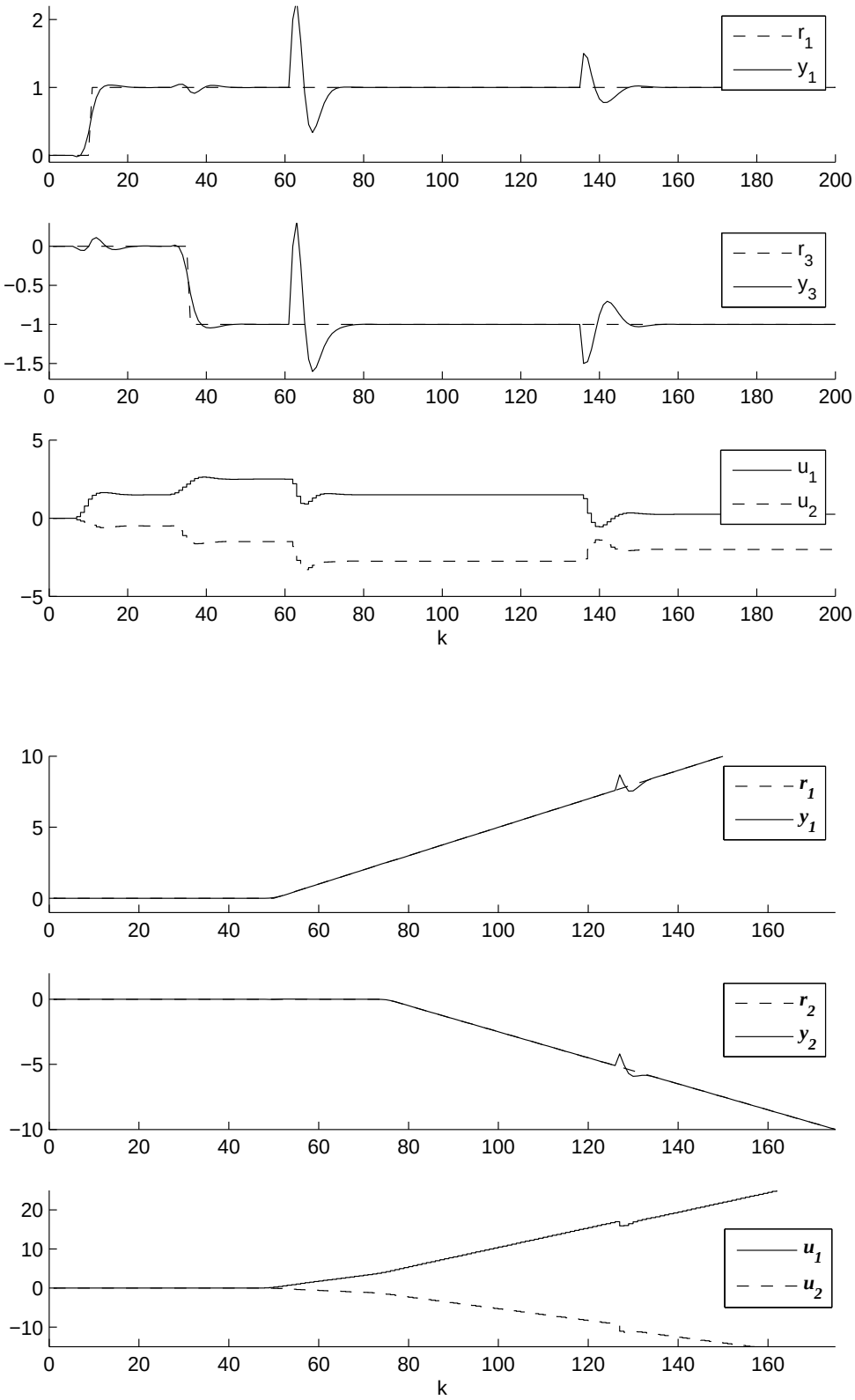

Figure 4-4. The results of a Method III MPC design applied to a simulated plant.

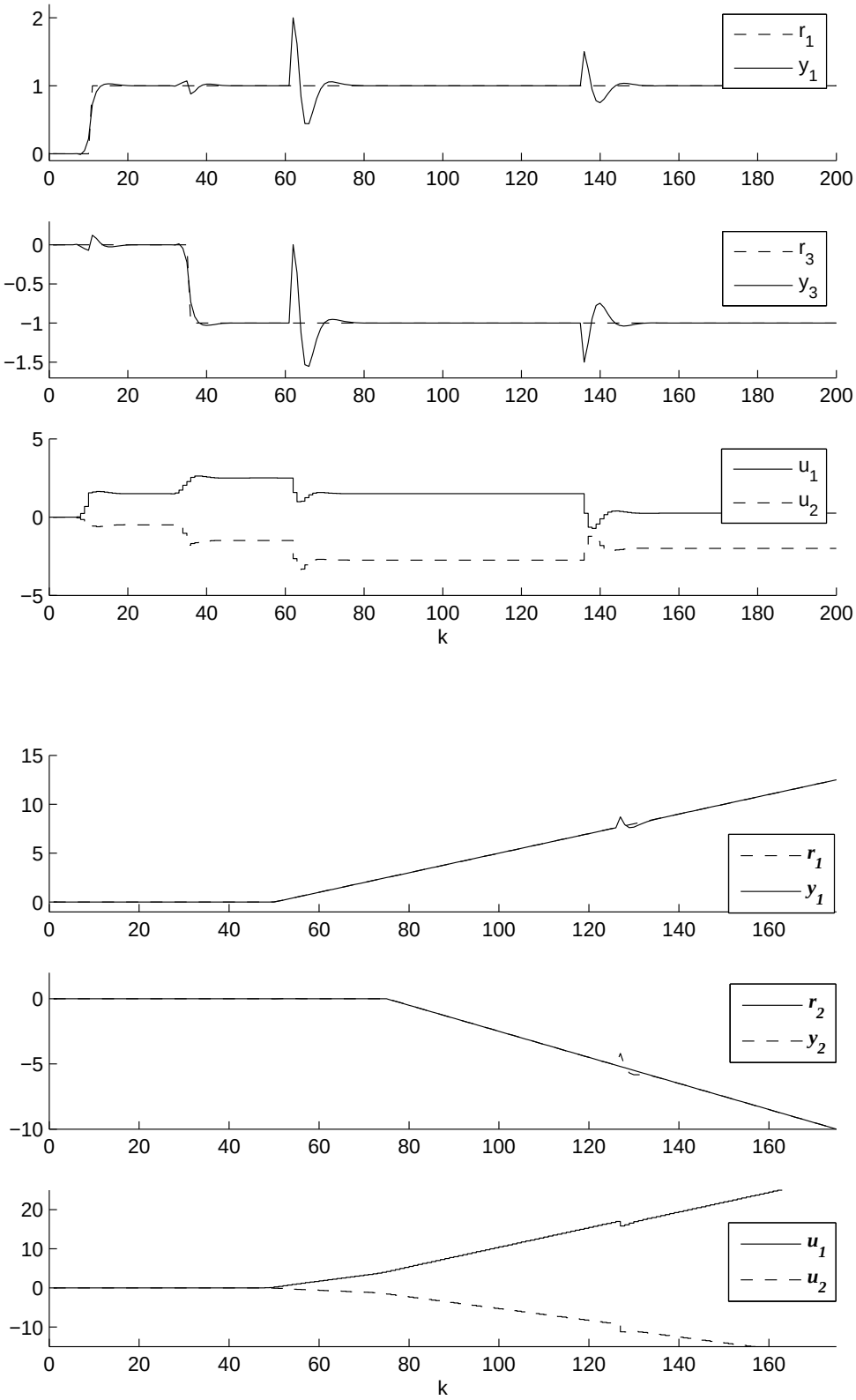

Figure 4-5. The results of a Method IV MPC design applied to a simulated plant.

CHAPTER 5
COLLECTIVE MOTION USING FLATNESS-BASED CONTROL METHODS

5.1 Introduction

Swarm intelligence has been a growing area of research interest due to its multiple potential applications, such as victim search-and-rescue and military operations. The increasingly more powerful computing and communications capabilities now available for mobile robots has also contributed to the surge in interests on this topic.

This paper addresses two fundamental problems in swarm control. The first is the design of a *flocking strategy*, which seeks to ensure that multiple vehicles follow a prespecified trajectory while maintaining a preset formation. The second is the design of a *formation-recovery* and *collision-avoidance strategy*, which consists of a set of tactics deployed to avoid collisions and to restore the preset formation when one or more vehicles deviate from their assigned positions.

Whether a rotating or non-rotating formation is used, the controller encourages each vehicle in the platoon to converge to its correct position in the formation, but it does not provide any guarantee collisions between members of the platoon will be avoided.

Mobile robots operating in real environments are likely to encounter obstacle and uneven terrain, which may cause deviations from their desired paths. There is also potential for equipment malfunctions, which could cause a robot to travel in an undesired direction, spin or stop altogether. In order to be useful a platoon must be robust against these problems, so recovery strategies that allow vehicles to avoid any platoon-mates that deviate from their designed trajectories are essential. These avoidance algorithms are also useful when vehicles attempt to assume formation from an arbitrary initial condition. In order to be successful a recovery strategy must be able to maintain cohesion in the platoon while avoiding an out-of-control vehicle and to reform the desire formation after the danger has passed.

This paper presents solutions to both of these problems for the case of nonholonomic two-wheeled mobile robots.

5.1.1 Literature Review

Most of the solutions proposed for flocking control deal with the formation recovery and collision avoidance problems in an indirect manner. For instance, the methodology of Elkaim and Kelbley [35] is based on potential functions, as commonly used in mobile robotics for obstacle avoidance problems. The net effect is the addition of virtual forces that determine an appropriate direction of travel since the vehicle is attracted to its goal, in this case a circle around a virtual leader, while repulsed by obstacles and other vehicles.

If the virtual leader moves sufficiently slowly, the platoon can avoid collision and travel between obstacles while maintaining its overall formation. The drawbacks of this technique are that the vehicles do not use a control law that guarantees convergence to the desired position. Instead, these schemes compute only an optimal direction of travel, without taking into account the dynamics of the vehicles. To ensure that obstacles can be actually avoided, a conservative approach can be adopted by defining a safe zone around each vehicle. The radius of the zone is defined so that it is large enough for the vehicles to stop without actually touching any obstacles. An approach of this nature is explored by Justh and Krishnaprasad [36], but enhanced by the incorporation of available insight about the vehicle dynamics. More specifically, each vehicle has a control law that includes one component that makes them head in the same direction as their neighbors, and a second component that makes them steer away from their neighbors if they get too close to another vehicle, or alternatively, to steer towards each other if they are too far apart. A target separation distance between the vehicles is established via a user-defined parameter.

5.1.2 Proposed Design

This paper extends the control strategy proposed for a single two-wheeled mobile robot by Buccieri *et. al.* to address a swarm of such vehicles [37].

The collision avoidance technique presented in this paper is inspired on the ideas of references [35] and [36] discussed above, since the approach proposed involves the explicit inclusion of potential functions to create virtual attraction and repulsion forces. However, instead of adopting a global perspective, collision avoidance and flocking control are treated separately by means of switching controllers that are activated based on the merits of particular situations. Furthermore, smooth switching functions are introduced to address potential instability problems associated with switching strategies [38] [39] and to augment the size of the domain of stabilizing parameters.

The paper is organized as follows. Section 5.2 introduces the dynamics of the vehicle, a restatement of the arguments that the underlying nonlinear dynamics are flat, and a review of the flatness-based controller for a single vehicle originally proposed by [37]. Section 5.3 introduces the extensions of the single-vehicle control design to the case of a collective motion problem involving multiple vehicles. Section 5.4 presents the results of comprehensive simulation studies to illustrate the performance of the proposed control strategy.

5.2 Flatness-Based Controller for a Single Vehicle

This section discusses the single-vehicle dynamics considered, and restates the controller design [37].

The nonholonomic wheeled mobile robot considered in this study can be represented by the kinematic model

$$\begin{aligned} \dot{x}_1 &= v_1 \cos x_3 \\ \dot{x}_2 &= v_1 \sin x_3 \\ \dot{x}_3 &= v_2 \end{aligned} \tag{5–1}$$

where the inputs v_1 and v_2 respectively represent the scalar linear and angular velocities of the robot, the states x_1 and x_2 are respectively denote the horizontal and vertical Cartesian coordinates of the robot's position, and state x_3 is the heading angle, defined as

the angle between the linear velocity vector and the positive horizontal Cartesian semiaxis (Figure 5-1).

Buccieri *et al.* [37] showed that the nonlinear dynamics (5–1) are flat. To be considered flat, a generic dynamic system $\dot{x} = f(x, v)$, where $x \in \Re^n$ and $v \in \Re^m$, must have an output $y \in \Re^m$ such that x and u can be written as explicit functions of y and its derivatives. More precisely, there must exist explicit maps $F : \Re^{np} \rightarrow \Re^n$ and $P : \Re^{(n+1)p} \rightarrow \Re^n$, and an integer $p \in \mathcal{N}$ such that

$$\begin{aligned} x &= F(y, y^{(1)}, \ldots, y^{(p-1)}) \\ v &= P(y, y^{(1)}, \ldots, y^{(p)}) \end{aligned} \tag{5–2}$$

where $y^{(i)}$ represents the $i - th$ derivative of the output. The dynamics (5–1) with the output defined as $y := [x_1 x_2]^T$ are flat because the third state and the inputs can be expressed in terms of the maps in equation

$$\begin{aligned} x_3 &= arctan2(\dot{x}_1, \dot{x}_2) \\ v_1 &= \sqrt{\dot{x}_1^2 + \dot{x}_2^2} \\ v_2 &= \frac{-\dot{x}_2\ddot{x}_1 + \dot{x}_1\ddot{x}_2}{\dot{x}_1^2 + \dot{x}_2^2} \end{aligned} \tag{5–3}$$

where $arctan2(.,.)$ is the arc-tangent function adjusted to return an angle in the range $(0, 2\pi)$ to correctly reflect the quadrant where the Cartesian coordinate point is located. Using these expressions it is straightforward to show that the remaining states x_1 and x_2 in (5–1) also satisfy the flatness conditions (5–2) [37].

Buccieri *et al.* [37] exploit the flatness of the dynamics in (5–1) to produce two control states, χ_1 and χ_2, that have linear state-space dynamics and that can be related to the original states and inputs using the flatness maps given in (5–3). The resulting control strategy for the case of the regulation problem consisting of tracking twice-differentiable

and continuous state trajectories x_1^{ref} for x_1 and x_2^{ref} for x_2, is given by the control laws

$$v_1 = \sqrt{\dot{\chi}_1^2 + \dot{\chi}_2^2} \tag{5–4}$$

$$v_2 = \frac{-\chi_2\dot{\chi}_1 + \chi_1\dot{\chi}_2}{{\chi_1}^2 + {\chi_2}^2} - k_p[x_3 - \arctan 2(\chi_1, \chi_2)] \tag{5–5}$$

that respectively define the linear and angular velocities of the vehicle, where the control states evolve according to the linear dynamical system

$$\dot{\chi}_1 = k_1(x_1 - x_1^{ref}) + k_2(\chi_1 - \dot{x}_1^{ref}) + \ddot{x}_1^{ref} \tag{5–6}$$

$$\dot{\chi}_2 = k_3(x_2 - x_2^{ref}) + k_4(\chi_2 - \dot{x}_2^{ref}) + \ddot{x}_2^{ref} \tag{5–7}$$

where x_1^{ref}, $\dot{x}_1^{ref}$, and $\ddot{x}_1^{ref}$ are respectively the reference position, velocity, and acceleration of the vehicle, and where the proportional feedback gain k_p and the constants k_i, $i = 1, 2, 3, 4$, and are tunable parameters. Note that the control states χ_1 and χ_2 in (5–6)-(5–7) are interpreted as the elements of the target trajectory-velocity vector $\chi = [\chi_1\chi_2]^T \in \Re^2$ in the flat state-space. The proportional feedback term with adjustable gain k_p is added to the angular velocity control law (5–5) to force the heading angle x_3 to converge to its specified value. This feature has been shown to ensure asymptotic convergence to the reference trajectory [37].

Note that for the special case of regulation, where $x_1^{ref} = x_2^{ref} = 0$ for all times, the controller's state-space equations (5–6)-(5–7) reduce to the simpler linear dynamics

$$\begin{aligned} \dot{\chi}_1 &= k_1 x_1 + k_2\chi_1 \\ \dot{\chi}_2 &= k_3 x_2 + k_4\chi_2 \end{aligned} \tag{5–8}$$

5.3 Collective Motion

It has been shown via numerical and experimental studies that the control strategy (5–4)-(5–7) yields substantially better tracking performance than classical dynamic feedback for the case of a single robot [37]. This section presents an extension of the

flatness-based single-vehicle controller discussed in the section 5.3 to the case of collective motion involving a platoon comprising N identical vehicles defining the platoon dynamics:

$$\begin{aligned} \dot{x}_{1i} &= v_{1i} \cos x_{3i} \\ \dot{x}_{2i} &= v_{1i} \sin x_{3i} \\ \dot{x}_{3i} &= v_{2i} \\ i &= 1, 2, \ldots, N \end{aligned} \tag{5–9}$$

The approach consists of assigning to each robot in the formation a flatness-based tracking controller, modified to address the additional constraints imposed by the requirement of preserving formation. Robustness against obstacles, or the failure of an agent in the platoon are discussed in an ensuing section.

The reference trajectory of each vehicle in a platoon is defined relative to that of a leader. One of the physical vehicles in the swarm is designated as the leader, or alternatively a virtual leader may be used. In the latter case each vehicle trajectory is specified relative to a desired trajectory for the platoon rather than the measured position of any vehicle. This desired trajectory may not necessarily correspond to the desired trajectory of any particular vehicle in the platoon. In this discussion we assume that the platoon has an assigned physical leader.

5.3.1 Notation and Definitions

Let the platoon be comprised of N identical vehicles of the kind described in section 5.3. The horizontal and vertical Cartesian coordinates and the heading angle for each individual robot i in the platoon are respectively denoted as x_{1i}, x_{2i}, and x_{3i}, $i = 1, 2, \ldots, N$, and for the special case of the platoon leader as (identified by the subscript $i = L$) as x_{1L}, x_{2L}, and x_{3L}. Analogously, the coordinates for the reference trajectory for each robot are denoted as x_{1i}^{ref} and x_{2i}^{ref}, and the coordinates of the reference trajectory for the real or virtual leader of the platoon are denoted as x_{1L}^{ref} and x_{2L}^{ref}.

A *formation* is defined as the set of position trajectories specified for the entire platoon

$$x^{ref} := \{(x_{1i}^{ref}, x_{2i}^{ref}), i = 1, 2, \ldots, N\} \tag{5–10}$$

where x_{1i}^{ref} and x_{2i}^{ref} are continuous twice-differentiable functions of time. The formation then uniquely defines the set of velocities and accelerations

$$\dot{x}^{ref} := \{(\dot{x}_{1i}^{ref}, \dot{x}_{2i}^{ref}), i = 1, 2, \ldots, N\} \tag{5–11}$$

and

$$\ddot{x}^{ref} := \{(\ddot{x}_{1i}^{ref}, \ddot{x}_{2i}^{ref}), i = 1, 2, \ldots, N\} \tag{5–12}$$

respectively.

5.3.2 Control Design for Trajectory Tracking

The control law

$$v_{1i} = \sqrt{\dot{\chi}_{1i}^2 + \dot{\chi}_{2i}^2} \tag{5–13}$$

$$\begin{aligned} v_{2i} = & \frac{-\chi_{2i}\dot{\chi}_{1i} + \chi_1\dot{\chi}_{2i}}{{\chi_{1i}}^2 + {\chi_{2i}}^2} \\ & -k_p[x_3 - \arctan 2(\chi_{1i}, \chi_{2i})] \end{aligned} \tag{5–14}$$

$$\begin{aligned} \dot{\chi}_{1i} = & k_1(x_{1i} - x_{1i}^{ref}) \\ & +k_2(\chi_{1i} - \dot{x}_{1i}^{ref}) + \ddot{x}_{1i}^{ref} \end{aligned} \tag{5–15}$$

$$\begin{aligned} \dot{\chi}_{2i} = & k_3(x_2 - x_{2i}^{ref}) \\ & +k_4(\chi_{2i} - \dot{x}_{2i}^{ref}) + \ddot{x}_{2i}^{ref} \end{aligned} \tag{5–16}$$

results when the scheme (5–4)-(5–7) is applied to each individual robot. The state equations (5–15) - (5–16) must be specialized for each formation under consideration, substituting the corresponding reference trajectory appearing on the right hand side.

5.3.2.1 Formation Schemes

Two formations are considered, namely a *nonrotational* and a *rotational* scheme . In both cases the reference trajectory for each robot is defined in terms of an offset from the

reference trajectory of the leader. In a nonrotational formation two *constant* offsets δ_{1i} and δ_{2i} from a fixed Cartesian-coordinates frame are used to define the referenceNote that $\delta_{1L} = \delta_{2L} = 0$.

$$\begin{aligned} x_{1i}^{ref} &:= x_{1L}^{ref} + \delta_{1i} \\ x_{2i}^{ref} &:= x_{2L}^{ref} + \delta_{2i} \\ i &= 1, 2, \ldots, N \end{aligned} \tag{5–17}$$

as illustrated in Figure 5-2.

In the alternative case of a rotational formation, two *time-dependent* rotational operators Δ_{1i} and Δ_{2i} are used to define equations

$$x_{1i}^{ref} = x_{1L}^{ref} + \Delta_{1i} \tag{5–18}$$

$$x_{2i}^{ref} = x_{2L}^{ref} + \Delta_{2i} \tag{5–19}$$

$$\Delta_{1i} = \delta_{1i} \cos x_{3L} - \delta_{2i} \sin x_{3L} \tag{5–20}$$

$$\Delta_{2i} = \delta_{2i} \cos x_{3L} + \delta_{1i} \sin x_{3L} \tag{5–21}$$

$$\tag{5–22}$$

where δ_{1i} and δ_{2i} are two constant offsets defined in a relative frame of reference centered on the leader vehicle, and oriented so that the positive vertical-semiaxis of the moving frame is oriented along the linear velocity vector $[\dot{x}_{1L}\ \dot{x}_{2L}]^T$, as illustrated in Figure 5-3. Obviously, $\Delta_{1L} = \Delta_{2L} = 0$.

The relative positions of the vehicles stay invariant in the nonrotating formation, regardless of the position and orientation of the leader of the platoon. This type of formation is simple to implement and imposes a low computational burden; however, because the size of the entire formation does not change, it may be an undesirable choice in cases where the platoon must maneuver in small spaces. In contrast, in a rotating formation each robot has constant offsets δ_1 and δ_2 with respect to a moving

Cartesian-coordinates frame of reference centered on the leader, and hence have non-constant offsets Δ_1 and Δ_2 with respect to a fixed Cartesian-coordinates frame.

5.3.2.2 Leader Controller

Before developing the governing control equations for the leader of the platoon, it is useful to first establish the leader's angular acceleration

$$\ddot{x}_{3L} = -2\frac{\tan x_{3L}}{(1+\tan^2 x_{3L})^2} \tag{5–23}$$

This expression is readily obtained after recognizing from Figure (5-1)

$$x_{3L} = \arctan 2(\dot{x}_{1L}, \dot{x}_{2L}) \tag{5–24}$$

taking the second derivative with respect to time to obtain

$$\ddot{x}_{3L} = -2(\dot{x}_{2L}/\dot{x}_{1L})\ [1+(\dot{x}_{2L}/\dot{x}_{1L})^2]^2 \tag{5–25}$$

and finally invoking the relationship

$$\dot{x}_{2L}/\dot{x}_{1L} = \tan x_{3L} \tag{5–26}$$

that is implied in the dynamics (5–9). Furthermore, since (5–9) also implies that

$$\ddot{x}_{3L} = \dot{v}_{3L} \tag{5–27}$$

from (5–23) it follows that

$$\dot{v}_{2L} = -2\frac{\tan x_{3L}}{(1+\tan^2 x_{3L})^2} \tag{5–28}$$

For the case of the platoon leader, where $i = L$, the reference trajectories (5–17) and (5–19) reduce to the simple form

$$\begin{aligned} x_{1i}^{ref} &:= x_{1L}^{ref} \\ x_{2i}^{ref} &:= x_{2L}^{ref} \end{aligned} \tag{5–29}$$

because the offsets and rotational operators are identically zero.

Then, the flatness-based controller for the leader of the platoon results when equations (5–13)-(5–16) are specialized to the case where $i = L$, and where the reference trajectories of the vehicle are given by (5–29), yielding:

$$v_{1L} = \sqrt{\dot{\chi}_{1L}^2 + \dot{\chi}_{2i}^2} \tag{5–30}$$

$$v_{2L} = \frac{-\chi_{2i}\dot{\chi}_{1L} + \chi_1\dot{\chi}_{2i}}{{\chi_{1L}}^2 + {\chi_{2i}}^2} - k_p[x_3 - \arctan 2(\chi_{1L}, \chi_{2i})] \tag{5–31}$$

$$\dot{\chi}_{1L} = k_1(x_{1L} - x_{1L}^{ref}) + k_2(\chi_{1L} - \dot{x}_{1L}^{ref}) + \ddot{x}_{1L}^{ref} \tag{5–32}$$

$$\dot{\chi}_{2L} = k_3(x_2 - x_{2L}^{ref}) + k_4(\chi_{2L} - \dot{x}_{2L}^{ref}) + \ddot{x}_{2L}^{ref} \tag{5–33}$$

$$\dot{v}_{2L} = -2\frac{\tan x_{3L}}{(1 + \tan^2 x_{3L})^2} \tag{5–34}$$

Note that the angular-velocity derivative $\dot{v}_{2L}$ given in the state equation (5–34) is not needed for the purposes of controlling the position of the leader of the platoon. It plays an important role, however, in the control algorithm for the other vehicles in the case of rotational formations. Note also that the control equations for the leader are identical for both formations under consideration, given that by definition, the offsets and hence the rotational operators, are identically zero for the case $i = L$. The control design for nonleader vehicles is presented in the ensuing sections, specialized for each formation of interest.

5.3.2.3 Nonleader Controller: Nonrotational Formation

In a nonrotational formation the trajectories of each vehicle are given by (5–17), which features constant offsets. Hence, the velocity and acceleration trajectories are the same as those of the leader. Then, the control dynamics (5–13)-(5–16) for each robot in

the platoon is expressed as follows:

$$
\begin{aligned}
v_{1i} &= \sqrt{\dot{\chi}_{1i}^2 + \dot{\chi}_{2i}^2} && (5\text{–}35)\\
v_{2i} &= \frac{-\chi_{2i}\dot{\chi}_{1i} + \chi_1\dot{\chi}_{2i}}{{\chi_{1i}}^2 + {\chi_{2i}}^2} \\
&\quad -k_p[x_3 - \arctan 2(\chi_{1i}, \chi_{2i})] && (5\text{–}36)\\
\dot{\chi}_{1i} &= k_1(x_{1i} - x_{1L}^{ref} - \delta_{1i}) && (5\text{–}37)\\
&\quad +k_2(\chi_{1i} - \dot{x}_{1L}^{ref}) + \ddot{x}_{1L}^{ref} && (5\text{–}38)\\
\dot{\chi}_{2i} &= k_3(x_{2i} - x_{2L}^{ref} - \delta_{2i}) && (5\text{–}39)\\
&\quad +k_4(\chi_{2i} - \dot{x}_{2L}^{ref}) + \ddot{x}_{2L}^{ref} && (5\text{–}40)\\
& && (5\text{–}41)
\end{aligned}
$$

The case of rotational formations is somewhat more complicated, and the details of the control design proposed are given below.

5.3.2.4 Nonleader Controller: Rotational Formation

The reference velocity and acceleration trajectories for each vehicle in a rotational formation are determined taking the appropriate first- and second-order time derivatives of (5–19). This yields the expressions:

$$
\begin{aligned}
\dot{x}_{1i}^{ref} &= \dot{x}_{1L}^{ref} + \dot{\Delta}_{1i} && (5\text{–}42)\\
\dot{x}_{2i}^{ref} &= \dot{x}_{2L}^{ref} + \dot{\Delta}_{2i} && (5\text{–}43)\\
\ddot{x}_{1i}^{ref} &= \ddot{x}_{1L}^{ref} + \ddot{\Delta}_{1i} && (5\text{–}44)\\
\ddot{x}_{2i}^{ref} &= \ddot{x}_{2L}^{ref} + \ddot{\Delta}_{2i} && (5\text{–}45)
\end{aligned}
$$

Then, specializing the control dynamics (5–14)-(5–16) to an individual member i of the platoon in a rotational formation, while obeying these trajectory derivatives, leads to the

control equations

$$v_{1i} = \sqrt{\dot{\chi}_{1i}^2 + \dot{\chi}_{2i}^2} \tag{5–46}$$

$$v_{2i} = \frac{-\chi_{2i}\dot{\chi}_{1i} + \chi_1\dot{\chi}_{2i}}{{\chi_{1i}}^2 + {\chi_{2i}}^2} - k_p[x_3 - \arctan 2(\chi_{1i}, \chi_{2i})] \tag{5–47}$$

$$\tag{5–48}$$

where

$$\dot{\chi}_{1i} = k_1(x_{1i} - x_{1L}^{ref} - \Delta_{1i}) + k_2(\chi_{1i} - \dot{x}_{1L}^{ref} - \dot{\Delta}_{1i}) + \ddot{x}_{1L}^{ref} + \ddot{\Delta}_{1i} \tag{5–49}$$

$$\dot{\chi}_{2i} = k_3(x_{2i} - x_{2L}^{ref} - \Delta_{2i}) + k_4(\chi_{2i} - \dot{x}_{2L}^{ref} - \dot{\Delta}_{2i}) + \ddot{x}_{2L}^{ref} + \ddot{\Delta}_{2i} \tag{5–50}$$

and where the rotational operators and their first and second derivatives are defined as

$$\Delta_{1i} = \delta_{1i}\cos x_{3L} - \delta_{2i}\sin x_{3L} \tag{5–51}$$

$$\Delta_{2i} = \delta_{2i}\cos x_{3L} + \delta_{1i}\sin x_{3L} \tag{5–52}$$

$$\dot{\Delta}_{1i} = -v_{2L}\Delta_{2i} \tag{5–53}$$

$$\dot{\Delta}_{2i} = v_{2L}\Delta_{1i} \tag{5–54}$$

$$\ddot{\Delta}_{1i} = -\dot{v}_{2L}\Delta_{2i} - v_{2L}x_{3L}\Delta_{1i} \tag{5–55}$$

$$\ddot{\Delta}_{2i} = \dot{v}_{2L}\Delta_{1i} - v_{2L}x_{3L}\Delta_{2i} \tag{5–56}$$

Equation (5–55) results after differentiating (5–53) and using the relationships (5–23) and (5–28); analogously for equation (5–56). Note that $\dot{v}_{2L}$, the derivative of the angular velocity of the leader given in (5–34), is utilized in (5–55) and (5–56) to calculate the second-derivatives of the rotational operators.

For coding purposes, it is useful to remark that the control algorithm for rotational formations (5–49)-(5–56) presented in this section reduces to the case of control for a

nonrotational formation (5–35)-(5–37) by incorporating the following redefinition of variables in (5–51)-(5–56):

$$\Delta_{1i} := \delta_{1i} \tag{5–57}$$

$$\Delta_{2i} := \delta_{2i} \tag{5–58}$$

$$\dot{\Delta}_{1i} := 0 \tag{5–59}$$

$$\dot{\Delta}_{2i} := 0 \tag{5–60}$$

$$\ddot{\Delta}_{1i} := 0 \tag{5–61}$$

$$\ddot{\Delta}_{2i} := 0 \tag{5–62}$$

Hence, in this sense, the rotational-formation control algorithm can be considered as a general case for design encompassing both of the formations considered in the scope of this work.

5.3.2.5 Numerical Simplifications

The user may find it desirable to reduce the computational burden associated with the calculation of the tangent function in (5–34. One viable option is to substitute the exact expression for $\dot{v}_{2L}$ given in (5–34) with the numerical differentiation approximation

$$\hat{\dot{v}}_{2L} = \frac{v_{2L}(t) - v_{2L}(t-\tau)}{\tau} \tag{5–63}$$

where t denotes the continuous time variable, and $\tau > 0$ is a small time interval, and where the variable $v_{2L}(t-\tau)$ represents a time-delayed version of $v_{2L}(t)$. Such an approximation may require the adoption of standard techniques needed to robustify the numerical estimation of derivatives, including appropriate signal filtering operations to avoid the propagation of noise-induced errors.

Another simplification that leads to a reduction in computational cost is the gross approximation

$$\begin{aligned} \ddot{x}_{1i}^{ref} &:= \ddot{x}_{1L}^{ref} \\ \ddot{x}_{2i}^{ref} &:= \ddot{x}_{2L}^{ref} \end{aligned} \tag{5–64}$$

that is obtained after ignoring the offsets in (5–19) for the purpose of calculation of the second derivatives of the rotational operators. The result is a modified algorithm readily obtained by setting $\ddot{\Delta}_{i1} = \ddot{\Delta}_{i2} = 0$ in (5–46)-(5–56), and ignoring equation (5–34) because the variable $\dot{v}_{2L}$ is no longer needed. Hence, this approach also avoids a potentially expensive tangent-function calculation. Unfortunately this approximation leads to steady-state errors in position. Therefore, the approximation (5–64) is only acceptable when it can be established *a priori* that the unavoidable steady-state errors in position are acceptable for the formation.

When the reference velocities $\dot{x}_{1L}^{ref}$ and $\dot{x}_{2L}^{ref}$ and accelerations $\ddot{x}_{1L}^{ref}$ and $\ddot{x}_{2L}^{ref}$ for the leader are not explicitly given in analytical form, they can be estimated on-line using the finite-difference approximations

$$\hat{\dot{x}}_{1L} = \frac{x_{1L}(t) - x_{1L}(t-\tau)}{\tau} \tag{5–65}$$

$$\hat{\dot{x}}_{2L} = \frac{x_{2L}(t) - x_{2L}(t-\tau)}{\tau} \tag{5–66}$$

$$\hat{\ddot{x}}_{1L} = \frac{x_{1L}(t) - 2x_{1L}(t-\tau) + x_{1L}(t-2\tau)}{\tau^2} \tag{5–67}$$

$$\hat{\ddot{x}}_{2L} = \frac{x_{2L}(t) - 2x_{2L}(t-\tau) + x_{2L}(t-2\tau)}{\tau^2} \tag{5–68}$$

where $\tau > 0$ is a small time interval, and where the argument $t - \tau$ and $t - 2\tau$ represent time-delayed signals. Numerical differentiation formulas of higher order can be alternatively used for improved precision.

5.3.3 Control Design for Collision Avoidance and Recovery

The literature reports collision avoidance schemes for platoons of robots implemented using simulated repulsion potentials established between two vehicles as they come in

proximity with each other [35]. In this paper we also adopt a potential-based collision avoidance algorithm, but in addition, we introduce a *switch function* σ_{ij} that turns on the collision-avoidance scheme when a safe-distance threshold is violated.

More specifically, each vehicle is assigned a collision-correction zone in the form of a torus centered about the vehicle, and characterized by an outer radius $R > 0$ and an inner radius $r < R$, as shown in Figure 5-4. The distance from a vehicle i to another vehicle $j \neq i$ is

$$d_{ij} := \sqrt{(x_{1i} - x_{1j})^2 + (x_{2i} - x_{2j})^2} \tag{5–69}$$

and the repulsive potential between these two vehicles is defined as $(x_{1i} - x_{1j})/d_{ij}^2$.

The collision-avoidance algorithm proposed consists of replacing the reference trajectories appearing in the leader's state equations (5–32)-(5–34) and in the follower control equations (5–49)-(5–50) with the collision-avoidance reference trajectories

$$\begin{aligned} x_{1i}^{ref,c} &:= x_{1L}^{ref} + \Delta_{1i} \\ &\quad + \sum_{\substack{j=0 \\ (j \neq i)}}^{N} \frac{\sigma_{ij}}{d_{ij}^2} (x_{1i} - x_{1j}) \end{aligned} \tag{5–70}$$

$$\begin{aligned} x_{2i}^{ref,c} &:= x_{2L}^{ref} + \Delta_{2i} \\ &\quad + \sum_{\substack{j=0 \\ (j \neq i)}}^{N} \frac{\sigma_{ij}}{d_{ij}^2} (x_{2i} - x_{1j}) \end{aligned} \tag{5–71}$$

while keeping the reference velocities and accelerations unchanged.

The switching function σ_{ij} appearing in (5–70)-(5–71) can be implemented using the computationally inexpensive but nonsmooth switching map

$$\sigma_{ij} = \begin{cases} 0 & if \quad d_{ij} > R \\ 1 - \frac{d_{ij} - r}{R - r} & if \quad r \leq d_{ij} \leq R \\ 1 & if \quad d_{ij} < r \end{cases} \tag{5–72}$$

(5–73)

or the more desirable, though computationally more demanding, smooth switching map

$$\sigma_{ij} = \frac{1}{2} + \frac{1}{2} \tanh \frac{\pi (R + r - 2d_{ij})}{R - r} \tag{5–74}$$

The nonsmooth switching policy (5–72) is an effective alternative, but may cause abrupt changes in vehicle trajectories, along with large control actions that could to signal saturations should the maximal value of inputs be exceeded. Such adverse effects are avoided using (5–72).

The collision avoidance strategy can easily be extended to avoid collisions with stationary or moving obstacles (other than platoon vehicles), by to the potential function the coordinates of each one of these objects along with their distances to each vehicle.

The switching function σ_{ij} tends to a value of zero for each vehicle if the distances d_{ij} for all vehicle-vehicle and vehicle-obstacle pairs are greater than the radius R. In that case the collision-avoidance policy is switched off. The collision-avoidance mechanism is turned on as soon as a vehicle enters the torus with outer radius R and inner radius r. The switching function progressively activates the collision-avoidance mechanism as the vehicles get closer to a distance r, starting from the threshold-violation distance R.

5.4 Simulation Studies

Simulations to compare recovery strategies are performed using a platoon of three vehicles (*i.e.*, $N = 3$) in a rotating formation. The leader (identified by the subindex $i = L$) is assigned a circular reference trajectory. The vehicles' initial positions and headings are arbitrary and not consistent with the formation.

In all cases considered the initial position of the vehicles as defined at $t = 0$ are listed in Table 5-1 along with the formation offsets and controller parameters. The leader's reference trajectory is the circle defined by

$$x_{1L}^{set} = A[cos\omega t) + sin(\omega t)] \tag{5–75}$$

$$x_{2L}^{set} = A[cos\omega t) + sin(\omega t)] \tag{5–76}$$

Table 5-1. Parameters used in the simulation studies

$x_{1L}(0) = 3$	$\delta_{1L} = 0$
$x_{2L}(0) = 2$	$\delta_{2L} = 0$
$x_{3L}(0) = 0$	$\delta_{12} = -0.5$
$x_{12}(0) = 2$	$\delta_{22} = -0.5$
$x_{22}(0) = 2$	$\delta_{13} = -0.5$
$x_{32}(0) = 0$	$\delta_{23} = 0.5$
$x_{13}(0) = 3.5$	
$x_{23}(0) = 2.5$	
$x_{33}(0) = 0$	
$\chi_{1i}(0) = 2, i = L, 2, 3$	
$\chi_{2i}(0) = 1, i = L, 2, 3$	
$\beta = 1$	$r = 0.3$
$k_1 = -100$	$R = 0.5$
$k_2 = -20$	$A = 1$
$k_3 = -100$	$\omega = 2$
$k_4 = -20$	
$k_p = 2$	

Figure 5-6 a scenario with no recovery strategy employed. Collisions occur as the vehicles attempt to restore their formation. One such collision is denoted by the point at which the curve showing the distance between them drops below the marked threshold.

Figure 5-7 shows the results of a simulation where the nonsmooth collision recovery policy (5–72) is implemented.. Here the collisions are avoided, but at the expense of a fairly energetic control action. Note that dramatic changes in the values of the control inputs are observed when the collision avoidance part of the control law is activated. Since these control inputs are robot velocities, the robots motors much be capable of producing the large accelerations required for this simulation to be realistic. This may not be practical for many applications,

Figure 5-8 shows the improvements in control-energy realized when the smooth recovery policy (5–74) is implemented. In this scenario collisions are again avoided. While more control action is needed than in the case where no collision avoidance is implemented, the undesirably large changes in velocity caused by the nonsmooth recovery strategy are prevented.

5.5 Conclusions

Several solutions, all based on the one-vehicle flatness-based controller, were proposed for the flocking problem and can be used for different applications, depending on the size and the available computing power of the platoon. The collision avoidance and the recovery problems were addressed at once by introducing a switching strategy between the flocking control mode and a potential-based collision avoidance mode. A smoother switching technique was also tested to reduce the likelihood of unwanted oscillations between the two modes. The resulting controller has been simulated, showed good results, and can thus be used for collective motion control.

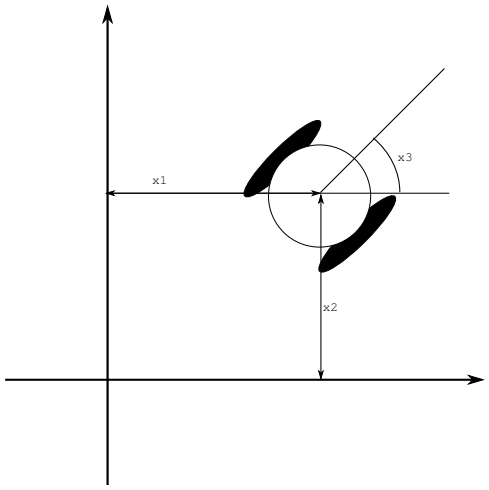

Figure 5-1. The coordinate system of the wheeled mobile robot. x_1 and x_2 are the Cartesian position of the robot, x_3 is the heading angle of the robot and the control inputs v_1 and v_2 are the linear and angular velocities respectively.

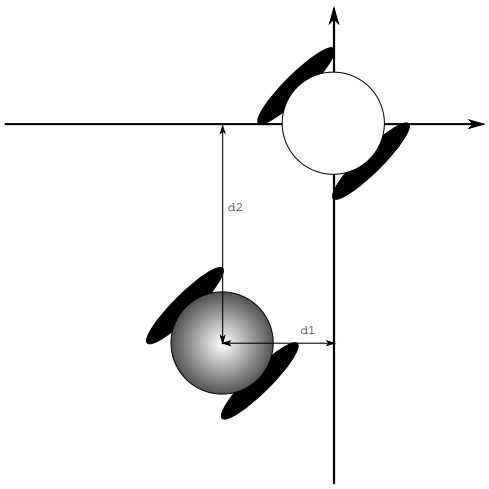

Figure 5-2. A non-rotating formation. δ_1 and δ_2 are the offsets of the shaded vehicle from the white vehicle, the leader, in the lab frame.

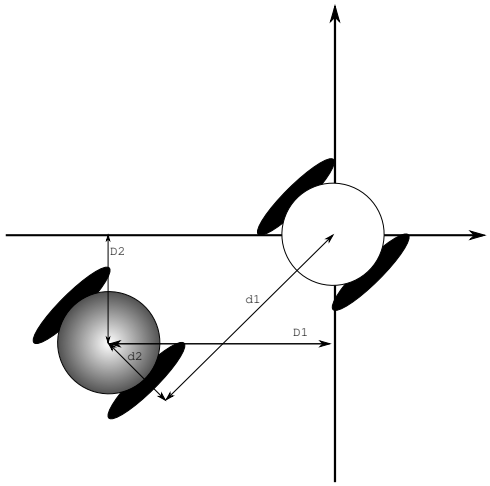

Figure 5-3. A rotating formation. δ_1 and δ_2 are the offsets of the shaded robot from the white robot, the leader, in the leaders frame. These result in the offsets Δ_1 and Δ_2 in the lab frame.

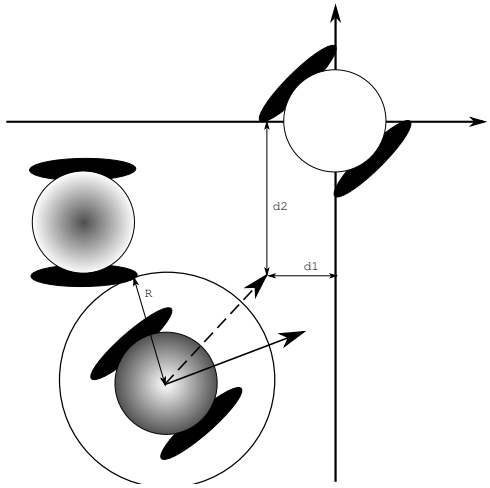

Figure 5-4. A robot using collision avoidance. The robot reacts to another robot approaching to within a distance R by changing its current reference from the bold dashed vector, to the solid one, representing the sum of the tracking and avoidance vectors.

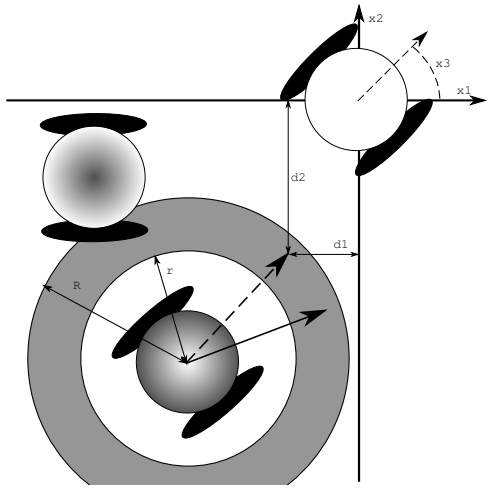

Figure 5-5. The robot reacts to another robot approaching to a distance between R and r by partially activating collision avoidance, resulting in the reference trajectory represented by the bold dashed vector, representing the sum of the tracking and avoidance vectors.

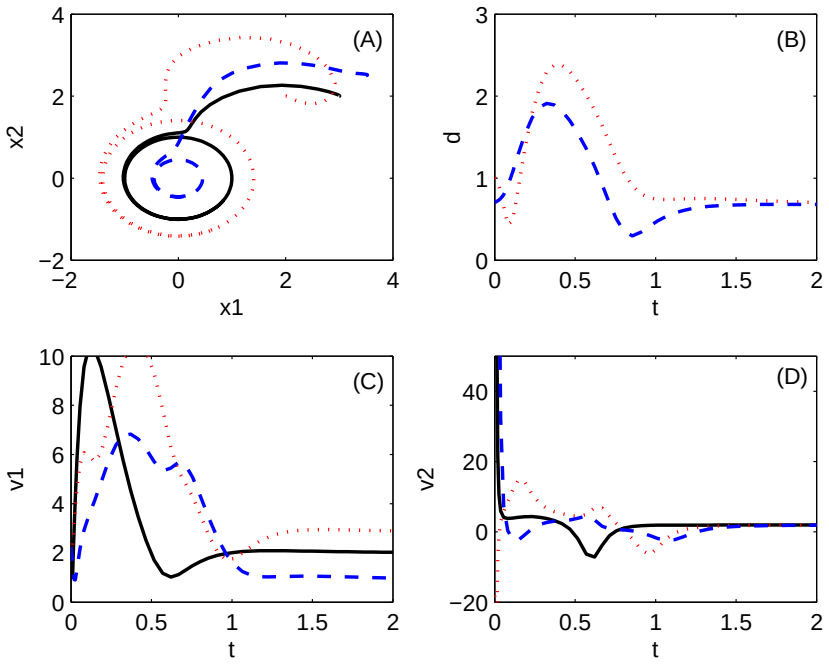

Figure 5-6. Plot A shows the trajectories of a swarm of three robots with no recovery strategy. The solid line represents the leader, which is attempting to track a circular trajectory. The dotted lines are the paths of two other vehicles tracking their positions in the formation. Plot B shows the distanced between each vehicle and the leader as a function of time. The point at which the distance drops below 0.3 indicates a vehicle colliding with the leader. Plots C and D show the control actions v_1 and v_2 respectively needed for each vehicle.

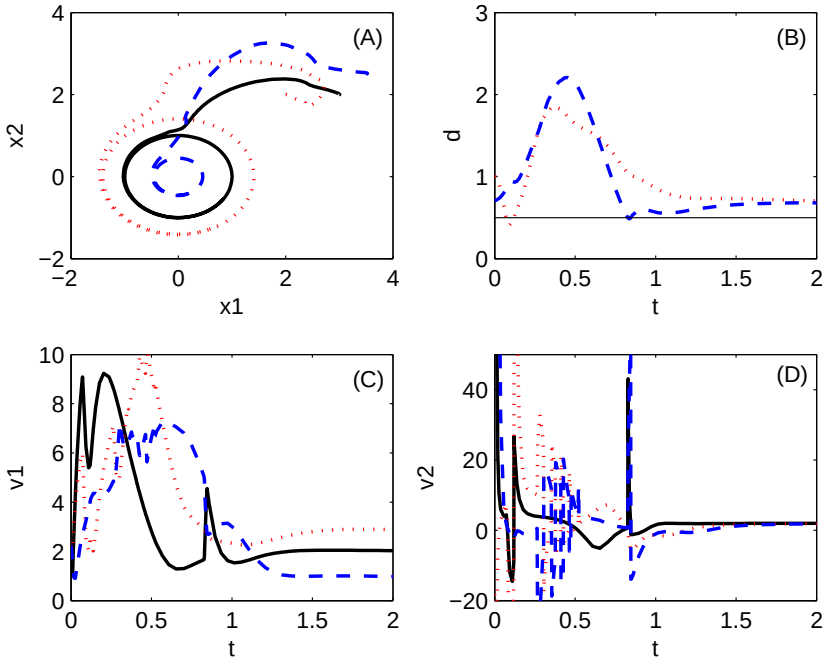

Figure 5-7. Plot A shows the trajectories of a swarm of three robots with a discontinuous recovery strategy. The trajectories are similar to those shown in figure 5-6 except where the robots act to avoid each other. Plot B shows the distance between each vehicle and the leader. Note that the close approach seen in Figure 5-6 is avoided. The horizontal line is added to show the distances r at which collision avoidance begins. Plot C shows the required control action for the second vehicle. Both inputs change dramatically when the collision avoidance mode is activated.

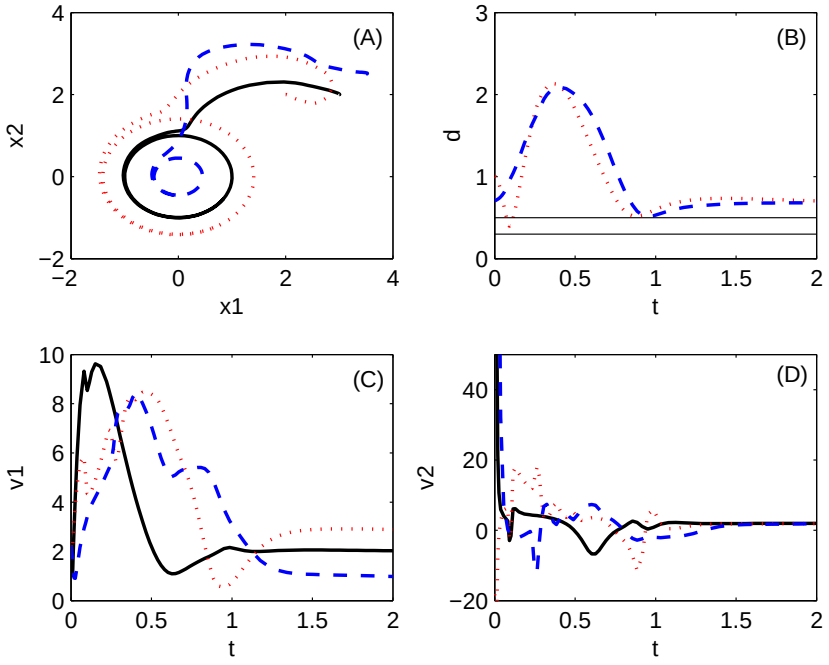

Figure 5-8. Plot A shows the trajectories of a swarm of three robots with a continuous recovery strategy. The trajectories are similar to those shown for the discontinuous strategy. Plot B shows the distanced between each vehicle indicating that no collisions occur. The two horizontal lines show the distance R at which collision avoidance begins and the distance r at which the collision avoidance law reaches its full effect. Plot C shows the required control action for the second vehicle. Both control actions lack the dramatic spikes seen above.

CHAPTER 6
CONCLUSION

The virtual control lab framework outlined here overcomes the limited interactivity and modularity of many other virtual labs allowing it to be used to develop simulation tools for a wide variety of pedagogical scenarios. This should make it even more valuable for providing a student with the learning value of a laboratory without physical access to lab equipment. Some potential scenarios in which it could be particularly successful are also examined, however flexibility is a key trait of the virtual control lab, so it should be easy for an instructor to apply it to scenarios which we have not specifically considered.

A PI^2 type double integral controller implemented either as a specialized three term controller or as two PI controllers in series reduces steady state error to zero for step and ramp type inputs and tuning correlations have been developed for first order plus dead time systems. These correlations give the controller parameters that minimize the ITAE as an exponential function of the nondimentionalized system parameters. Stability of the systems recommended by the correlation has been established via Nyquist methods.

Model predictive controllers modified with integral states or velocity control effectively eliminate offset with respect to step changes in set point or constant value disturbances. Controllers containing both integral states and velocity control or multiple-integral states can successfully eliminate offset with respect to ramp changes in set point.

The flatness-based controller previously implemented for a single wheeled mobile robot. A switching technique allows for flocking control and collision avoidance to be implemented simultaneously. Simulation of a swarm of robots operating under such a controller shows promising results, especially when used with smooth switching.

APPENDIX A
ALGEBRAIC BACKGROUND FOR RAMP TRACKING CONTROLLERS

A.1 Comparison of PI^2 Controllers with Belanger and Luyben's Design

In their work Belanger and Luyben use a PI^2 controller of the form

$$G_c = K_c \left(1 + \frac{1}{\tau_{i1} s} + \frac{1}{\tau_{i2} s^2}\right) \tag{A–1}$$

while the controller referred to as Scheme 2 in chapter 3 is of the form

$$G_c = K_c \left(1 + \frac{1}{\tau_{i1} s}\right)\left(1 + \frac{1}{\tau_{i2} s^2}\right) = K_c \left(1 + \frac{\tau_{i1} + \tau_{i2}}{\tau_{i1}\tau_{i2} s} + \frac{1}{\tau_{i1}\tau_{i2} s^2}\right) \tag{A–2}$$

Despite superficial differences these two controllers are quite similar. This can be established by equating the two transfer functions and labeling Belanger and Luyben's time constants with a prime, yielding

$$K_c \left(1 + \frac{1}{\tau_{i1}' s} + \frac{1}{\tau_{i2}' s^2}\right) = K_c \left(1 + \frac{\tau_{i1} + \tau_{i2}}{\tau_{i1}\tau_{i2} s} + \frac{1}{\tau_{i1}\tau_{i2} s^2}\right) \tag{A–3}$$

Dividing by the gain and equating like powers of s yields the relationships

$$\tau_{i1}' = \frac{\tau_{i1}\tau_{i2}}{\tau_{i1} + \tau_{i2}} \tag{A–4}$$

and

$$\tau_{i2}' = \tau_{i1}\tau_{i2} \tag{A–5}$$

Luyben argues that the relationship

$$\tau_{i2}' = 4\tau_{i1}'^2 \tag{A–6}$$

should be used to reduce the impact on the stability and the number of parameters needing tuning. Substituting the equivalent groupings of the time constants used here yields

$$\begin{aligned}
\tau_{i1}\tau_{i2} &= 4\left(\frac{\tau_{i1}\tau_{i2}}{\tau_{i1} + \tau_{i2}}\right)^2 && \text{(A–7)}\\
(\tau_{i1} + \tau_{i2})^2 &= 4\tau_{i1}\tau_{i2} && \text{(A–8)}\\
0 &= \tau_{i1}^2 - 2\tau_{i1}\tau_{i2} + \tau_{i2}^2 && \text{(A–9)}\\
\tau_{i1}' &= \frac{1}{2}\tau_{i1} = \frac{1}{2}\tau_{i2} && \text{(A–10)}\\
\tau_{i2}' &= \tau_{i1}^2 && \text{(A–11)}
\end{aligned}$$

Thus a controller of the type put forth by Belanger and Luyben can be written for any scheme two or scheme three controller developed from the tuning correlations written here

and any controller that follows Luyben's tuning rules can be written as a scheme three controller since their time constant relationship is analogous to the time constants of two PI controllers being the same.

A.2 Constraints Relating the Different PI^2 schemes

The constraint $K_{i1}^2 > 4K_cK_{i2}$ describes the subset of scheme 1 controllers which can also be represented by scheme 2 controllers with real parameters. This relationship can be found by equating like powers and solving the resulting quadratics. The transfer functions for the scheme one and two controllers are

$$G(s) = K_c + K_{i1}\frac{1}{s} + K_{i2}\frac{1}{s^2} \tag{A–12}$$

and

$$G(s) = K_c\left(1 + \frac{1}{\tau_{i1}s}\right)\left(1 + \frac{1}{\tau_{i2}s}\right) \tag{A–13}$$

respectively. Equating like powers yields

$$\begin{aligned} K_c &= K_c && \text{(A–14)}\\ K_{i1} &= K_c\left(\frac{1}{\tau_{i1}} + \frac{1}{\tau_{i2}}\right) && \text{(A–15)}\\ K_{i2} &= K_c\frac{1}{\tau_{i1}\tau_{i2}} && \text{(A–16)} \end{aligned}$$

$$\begin{aligned} \frac{1}{\tau_{i1}} &= \frac{K_{i2}\tau_{i2}}{K_c} && \text{(A–17)}\\ K_{i1} &= K_c\left(\frac{K_{i2}\tau_{i2}}{K_c} + \frac{1}{\tau_{i2}}\right) && \text{(A–18)}\\ K_{i1}\tau_{i2} &= K_{i2}\tau_{i2}^2 + K_c && \text{(A–19)}\\ 0 &= K_{i2}\tau_{i2}^2 - K_{i1}\tau_{i2} + K_c && \text{(A–20)}\\ \tau_{i2} &= \frac{K_{i1} + -\sqrt{K_{i1}^2 - 4K_{i2}K_c}}{2K_{i2}} && \text{(A–21)} \end{aligned}$$

This equation has real solutions only for

$$\begin{aligned} K_{i1}^2 - 4K_cK_{i2} > 0 && \text{(A–22)}\\ K_{i1}^2 > 4K_cK_{i2} && \text{(A–23)} \end{aligned}$$

A.3 Proof of Offset Elimination for Several Forms of PI^2 Tracking Ramps

One of the goals of the controllers set forth here is to achieve zero steady state offset, so the closed loop error as described in section 2 must go to zero at long times.

$$\lim_{s\to 0} se(s) = 0 \tag{A–24}$$

The error can be expressed as follows for a ramp setpoint input.

$$e(s) = G_{re}(s)r(s) \tag{A–25}$$

$$e(s) = \frac{1}{1 + G_p(s)G_c(s)} \frac{1}{s^2} \tag{A–26}$$

The first order plant is represented by the transfer function

$$G_p(s) = \frac{K}{\tau s + 1} \exp -\theta s \tag{A–27}$$

While there are several schemes for the controller as discussed in section two.

$$G_{c1}(s) = K_c + K_{i1} \frac{1}{s} + K_{i2} \frac{1}{s^2} \tag{A–28}$$

$$G_{c2}(s) = K_c \left(1 + \frac{1}{\tau_{i1} s}\right) \left(1 + \frac{1}{\tau_{i2} s}\right) \tag{A–29}$$

$$G_{c3}(s) = K_c \left(1 + \frac{1}{\tau_{i1} s}\right)^2 \tag{A–30}$$

In general these can be written

$$G_{cj}(s) = \frac{K_c}{s^2} N_j \tag{A–31}$$

$$e(s) = \frac{1}{1 + \frac{K K_c N_j}{(\tau s + 1) s^2} \exp -\theta s} \frac{1}{s^2} \tag{A–32}$$

$$\tag{A–33}$$

The zero-offset requirement is thus equivalent to

$$\lim_{s \to 0} \frac{1}{s + \frac{K K_c N_j}{(\tau s + 1) s} \exp -\theta s} = 0 \tag{A–34}$$

$$\lim_{s \to 0} \frac{s\,(\tau s + 1)}{N_j} = 0 \tag{A–35}$$

This is clearly true in the case when

$$\lim_{s \to 0} Nj \neq 0 \tag{A–36}$$

it may also be true with Nj does approach zero, but that analysis is not needed since it can be shown

$$\lim_{s\to 0} N_1 = \lim_{s\to 0} s^2 + \frac{K_{i1}}{K_c}s + \frac{K_{i2}}{K_c} = \frac{K_{i2}}{K_c} \quad \text{(A–37)}$$

$$\lim_{s\to 0} N_2 = \lim_{s\to 0} s^2 + \frac{\tau_{i1}+\tau_{i2}}{\tau_{i1}\tau_{i2}}s + \frac{1}{\tau_{i1}\tau_{i2}} = \frac{1}{\tau_{i1}\tau_{i2}} \quad \text{(A–38)}$$

$$\lim_{s\to 0} N_2 = \lim_{s\to 0} s^2 + \frac{2}{\tau_{i1}}s + \frac{1}{\tau_{i1}^2} = \frac{1}{\tau_{i1}^2} \quad \text{(A–39)}$$

$$\text{(A–40)}$$

Thus Nj approaches a nonzero constant value for each case and the zero-offset condition is met.

APPENDIX B
EXAMPLE MATLAB CODE

B.1 Double Integral Controller Tuning

The following code calculated the optimal tuning values used to produce the curves in chapter 3 which are in turn the basis for the recommended tuning relationships.

B.1.1 TuneCorMain.m

```
%Main tuning correlation generation routing for PI2 controllers

global tf;
global form;
global Kc;
global taui1;
global taui2;
global Ki;
global Ki2;
global r;
global y;
global t;

form = 3;
sp = 0;

kpoints=20;
tpoints=20;
rpoints=50;

KA=logspace(-1,1,kpoints);
tauA=logspace(0,2,tpoints);
ratioA=logspace(log10(.1),log10(50),rpoints);

flags=zeros(rpoints,kpoints,tpoints);
err=zeros(rpoints,kpoints,tpoints);

if(form==1)
    KcOpt=zeros(rpoints,kpoints,tpoints);
    Taui1Opt=zeros(rpoints,kpoints,tpoints);
    Taui2Opt=zeros(rpoints,kpoints,tpoints);
    clear KiOpt;
    clear Ki2Opt;
elseif(form==2)
    KcOpt=zeros(rpoints,kpoints,tpoints);
    Taui1Opt=zeros(rpoints,kpoints,tpoints);
    clear Taui2Opt;
```

```
    clear KiOpt;
    clear Ki2Opt;
else
    KcOpt=zeros(rpoints,kpoints,tpoints);
    KiOpt=zeros(rpoints,kpoints,tpoints);
    Ki2Opt=zeros(rpoints,kpoints,tpoints);
    clear taui1Opt;
    clear taui2Opt;
end

options=optimset('fminsearch');
options=optimset(options,'MaxFunEvals',1000,'TolFun',1e-2,'Display','off');

fid=fopen(['tc' datestr(date,29) 'f' num2str(form) '.txt'],'a');

%for i=1:rpoints
for i=43:rpoints
    disp(['Processing ' num2str(ratioA(i))]);
    for j=10:kpoints
        K=KA(j);
        for k=1:tpoints
            tau=tauA(k);
            theta=ratioA(i)*tau;
            tf=max(theta,tau)*15;
            %Guessed based on previous runs (improves speed greatly)
            if (form==2)
                if ratioA(i)<1
                    Kc0=(.990*ratioA(i).^-.835)/K;
                elseif ratioA(i)<10
                    Kc0=(0.822*ratioA(i).^-.0975)/K;
                else
                    Kc0=0.6/K;
                end
                taui0=tau/(.443*ratioA(i).^-.672);
                x0=[Kc0 taui0 taui0];
            elseif (form==1)
                if ratioA(i)<1
                    Kc0=(.995*ratioA(i).^-.826)/K;
                elseif ratioA(i)<10
                    Kc0=(.837*ratioA(i).^-.1152)/K;
                else
                    Kc0=0.65/K;
                end
                x0=[Kc0 tau/(.505*ratioA(i).^-.558)...
                tau/(.390*ratioA(i).^-.789)];
```

```
            else
                if ratioA(i)<1
                    Kc0=(1.076*ratioA(i).^-.7868)/K;
                elseif ratioA(i)<10
                    Kc0=(0.8318*ratioA(i).^-.0964)/K;
                else
                    Kc0=0.6/K;
                end
                taui0=tau/(.443*ratioA(i).^-.672);
                x0=[Kc0 Kc0*2*(.4317*(ratioA(i)).^-.6820)/tau...
                 Kc0*(.4317*(ratioA(i)).^-.6820)^2/tau^2];
            end
            [minx, min, flag]=fminsearch(@itae,x0,options); %Simplex

            fprintf(fid,'%f\t%f\t%f\t%f\t%f\t%f\n\r',K,tau,theta,minx(1),minx(2),minx(3)
        end
    end
end

fclose(fid);
```

B.1.2 itae.m

```
function e = itae(x)
%Generates ITAE cost function
% requires Kc, taui1, taui1, Ki,Ki2,t,y,r,tf and form already be set

global Kc;
global taui1;
global taui2;
global Ki;
global Ki2;
global form;

global t;
global y;
global r;
global tf;

Kc=x(1);

if (form~=3)
    taui1=x(2);
    Ki=1;
    Ki2=1;
else
```

```
    Ki=x(2);
    Ki2=x(3);
    taui1=1;
    taui2=1;
end

if (form==1)
    taui2=x(3);
else
    taui2=1;
end

try
    %Simulate
    sim('tunecor',tf);
    %Calculate Error
    e=0;
    for k=1:size(y)-1
       e=e+abs(r(k)-y(k))*t(k)*(t(k+1)-t(k));
   end
catch
   warning('Simulation Error');
   e=1E100;
end

if (sum(abs(r(length(t)-5:length(t))-y(length(t)-5:length(t)))...
./r(length(t)-5:length(t)))>.05)
    e=e+1E20;
end
```

B.2 Offset Free MPC

The following is the code for producing figure 4.6. Similar scripts to example1.m can be created to simulate other plant and controller combinations.

B.2.1 Example1.m

```
%Generate CSTR example with Method I controller

clear;
close all;

plant=2;
%Weights
Q = [10 0 1];
S = [10 0 1];%number of elements equal to the number of rows of Cm
```

```
T = [5 0 0.5];%number of elements equal to the number of rows of Cm
R = [0.5 0.5];%number of elements equal to the number of columns of Bm

%Now define the prediction Np and the control Nc horizon.
Np  = 10;
Nc  = 3;

%Observer
Lx=0.2*eye(3);
Ld=0.1*eye(3);
Lp=0.03*eye(3);

method=1;
stateEqn=0;
outputEqn=0;
est=0;

[r d p] = MPCGenerateSP(Np,Nc,[0 0 0],[0.2 0 0.2],[0 0 0],...
[-1.762e-5 7.784e-2 6.592e-2],[0 0 0],[-1.762e-5 0 6.592e-2]);

MPCMain;

subplot(311);
hold on;
stairs(1:ttotal,r(1:ttotal,1),'k-');
stairs(1:ttotal,y(1:ttotal,1),'k--');
h=legend(['r_1';'y_1']);
set(h,'FontName','Times','FontAngle','italic','FontWeight','Bold');
axis([0 400 -0.1 0.4]);
subplot(312);
hold on;
stairs(1:ttotal,r(1:ttotal,3),'k-');
stairs(1:ttotal,y(1:ttotal,3),'k--');
h=legend(['r_3';'y_3']);
set(h,'FontName','Times','FontAngle','italic','FontWeight','Bold');
axis([0 400 -0.1 0.4]);
subplot(313);
hold on;
stairs(1:ttotal,u(1:ttotal,1),'k');
stairs(1:ttotal,u(1:ttotal,2),'k--');
h=legend(['u_1';'u_2']);
set(h,'FontName','Times','FontAngle','italic','FontWeight','Bold');
axis([0 400 -20 10]);
xlabel('t');
set(gcf,'PaperPosition',[0 0 7 5])
```

B.2.2 MPCMain.m

```
%Main Operating Code for Offset Free MPC
%Call after model, method, weights and setpoint have been initialized


%===INITIALIZATION===%

if method==0 || method==2
    S=zeros(size(S));
end

if method~=4
    T=zeros(size(T));
end

[Ap Bp Cp Am Bm Cm]=MPCLoadPlant(plant);
[Q R S T]=MPCAugmentWeights(Q,R,S,T,Np,Nc);
[delta0 delta1 delta2] = MPCDeltas(Bm,Np,Nc);
if(stateEqn==1)
    [AIIa AIIad BII CII CII0 DII EII MI NI] = MPCMatsII(Am,Bm,Cm,Np,Nc);
    [Ky Kz Kw] = MPCGains(1,method,DII,NI,delta0,Q,R,S,T);
else
    [AI BI CI CI0 DI EI FI MI NI] = MPCMatsI(Am,Bm,Cm,Np,Nc);
    [Ky Kz Kw] = MPCGains(0,method,DI,NI,delta0,Q,R,S,T);
end

%Error Check
if(method>4)
    error('Method not currently available');
end
if(est>2||est<0)
    error('Unknown estimator type');
end
if(outputEqn>3 || outputEqn<0)
    error('Unknown ouput equation type');
end
if(stateEqn>2 || stateEqn<0)
    error('Unknown state equation type');
end


%===CALCULATION LOOP==%
ykm1 = zeros(size(Cm,1),1);
```

```
ykm2 = ykm1;
xkm1 = zeros(size(Am,1),1);
ukm1 = zeros(size(Bm,2),1);
dukm1 = ukm1;
dkm1 = zeros(size(Am,1),1);
xhatkm1 = xkm1;
xhatkm2 = xkm1;
dhatkm1 = dkm1;
phatkm1 = ykm1;
zk = ykm1;
wk = ykm1;
rkm1 = r(1,:);

y=[];
x=[];
u=[];
xhat=[];
dhat=[];
phat=[];

ttotal=size(r,1)-Np;

for k=1:ttotal

    %Plant
    xk = Ap*xkm1 + Bp*ukm1 + dkm1;
    yk = Cp*xk + p(k,:)';

    %Construct r vector from setpoint data
    rAug = reshape(r(k+1:k+Np,:)',[],1);

    %Estimator
    if(est==0)
        xhatk=xk;
        dhatk=zeros(size(xk));
        phatk=zeros(size(yk));
    elseif(est==1)
        xhatk = Am*xhatkm1 + Bm*ukm1 + dhatkm1 + Lx*(ykm1-Cm*xhatkm1-phatkm1);
        dhatk = dhatkm1 + Ld*(ykm1-Cm*xhatkm1-phatkm1);
        phatk = phatkm1 + Lp*(ykm1-Cm*xhatkm1-phatkm1);
    elseif(est==2)
        %incremental predictor
        xhatk = (Am+eye(size(Am)))*xhatkm1 - Am*xhatkm2 + Bm*dukm1...
         + Lx*(ykm1 - ykm2 - Cm*xhatkm1 + Cm*xhatkm2);
        dhatk = dhatkm1 + Ld*(ykm1-Cm*xhatkm1-phatkm1);
```

```
    phatk = phatkm1 + Lp*(ykm1-Cm*xhatkm1-phatkm1);
end

if(stateEqn==0)
    if(outputEqn==0)
        yhat=CI*xhatk;
    elseif(outputEqn==1)
        yhat=CI0*xhatk+EI*yk;
    elseif(outputEqn==2)
        yhat=CI0*xhatk+EI*yk;
    elseif(outputEqn==3)
        yhat=CI*xhatk+EI*phatk;
    end
    if(method==2 || method==3)
        yhat=yhat + DI*delta2*ukm1;
    end
elseif(stateEqn==1)
    if(outputEqn==0)
        yhat=CII0*xhatk+CII*xhatkm1;
    elseif(outputEqn==1)
        yhat=CII0*(xhatk-xhatkm1)+EII*yk;
    elseif(outputEqn==2)
        yhat=CII0*(xhatk-xhatkm1)+EII*yk;
    elseif(outputEqn==3)
        yhat=CII0*xhatk+CII*xhatkm1+EII*phatk;
    end
    if~(method==2 || method==3)
        yhat=yhat - DII*delta1*ukm1;
    end
elseif(stateEqn==2)
    if(outputEqn==0)
        yhat=CI*xhatk+FI*dhatk;
    elseif(outputEqn==1)
        yhat=CI0*xhatk+FI*dhatk+EI*yk;
    elseif(outputEqn==2)
        yhat=CI0*xhatk+FI*dhatk+EI*yk;
    elseif(outputEqn==3)
        yhat=CI*xhatk+FI*dhatk+EI*phatk;
    end
    if(method==2 || method==3)
        yhat=yhat + DI*delta2*ukm1;
    end
end

%Integrating State (if needed)
```

```
    z = MI*(zk+r(k,:)'-yk) + NI*(rAug-yhat);
    zkp1=zk+r(k,:)'-yk;

    w = MI*(wk+zk) + NI*z;
    wkp1=wk+zk;

    %Control Law
    uk = Ky*(rAug - yhat) + Kz*z +Kw*w;
    uk=uk(1:length(ukm1)); %Only implement next u
    if(method==2 || method == 3)
    %For velocity weighted methods calculated "u(k)" is really delta u(k)
        dukm1 = uk;
        uk=uk+ukm1;
    else
        dukm1 = uk-ukm1;
    end

    x = [x; xk'];
    y = [y; yk'];
    u = [u; uk'];
    xhat = [xhat; xhatk'];
    dhat = [dhat; dhatk'];
    phat = [phat; phatk'];

    xkm1 = xk;
    ykm2 = ykm1;
    ykm1 = yk;
    ukm1 = uk;
    dkm1 = d(k,:)';
    xhatkm2 = xhatkm1;
    xhatkm1 = xhatk;
    dhatkm1 = dhatk;
    phatkm1 = phatk;
    zk = zkp1;
    wk = wkp1;
end
```

B.2.3 MPCLoadPlant.m

```
function [Ap Bp Cp Am Bm Cm] = MPCloadPlant(p)
%Load model and related data for one of several predefined plants
%Usage [Ap Bp Cp Am Bm Cm] = MPCloadPlant(p)

switch p
    case 1
        %system: two interconected tanks, from Coughanowr
```

```
    %D. Process Systems Analysis and Control. pg 459
    Am = [-3 2;4 -5];
    Bm = [1 0;0 2];
    Cm = [1 0;0 1];
    Ap = Am;
    Bp = Bm;
    Cp = Cm;
    Bd = 0.7*[1 1]';
    Dp = 0.7*[1 1]';

case 2
    %Tank Reactor model for the problem in Rawlings AICHE paper.
    Am = [0.2511 -3.368e-3   -7.056e-4;
          11.06   0.3296     -2.545;
          0       0           1];
    Bm = [-5.426e-3  1.530e-5;
           1.297     0.1218;
           0        -6.592e-2];
    %Cm = [1 0 0; 0 0 1];
    Cm = eye(3);
    Ap = Am;
    Bp = Bm;
    Cp = Cm;
    Bd =  [-1.762e-5 7.784e-2 6.592e-2]';
    Dp = [0 0 0]';

case 3
    %Ill-conditioned distilation model from Rawlings
    phi = exp(-5/75);
    Am  = [phi 0;0 phi];
    Bm  = phi*[0.878 -0.864;1.082 -1.096];
    Cm = eye(3);
    Ap = Am;
    Bp = phi*[0.878 -0.88; 1.1 -1.096];
    Cp = Cm;
    Bd = [1 1]';
    Dp = [0 0]';

case 4
    %Simple integrating model from Muske and Badgewell
case 5
    %Plant A from Rawlings '93
    Am=[4/3 -2/3; 1 0];
    Bm=[1; 0];
    Cm=[-2/3 1];
```

```
    Ap=Am;
    Bp=Bm;
    Cp=Cm;

case 6
    %Plant C from Rawlings '93
    Am = diag([0.5 0.6 0.5 0.6]);
    Bm = [0.5 0; 0 0.4; 0.25 0; 0 0.6];
    Cm = [1 1 0 0; 0 0 1 1];
    Ap=Am;
    Bp=Bm;
    Cp=Cm;


case 7
    %Large distillation model frorm Muske and Badgewell
    %Note: only 8 state version implemented. Full 28 state model not
    %available

    Am = [1 0 0      0        0     0         0       0;
          0 1 0      0        0     0         0       0;
          0 0 0.799 -0.00172 -0.406 0.000867 -0.0457 -0.0158;
          0 0 0      0.645    0.444 0.535     0.159   0.139;
          0 0 0      0        0.120 0.00229  -0.128  -0.0431;
          0 0 0      0        0     0.0362    0.0127 -0.296;
          0 0 0      0        0     0         0.00386 0.00220;
          0 0 0      0        0     0         0       0.000372];

    Bm = [-7.5      7.5     0   7.5 -7.5;
          -7.5      7.5    -7.5 0    0;
           0.0713  -0.0729  0   0    0.374;
          -0.00697 -0.0757  0   0    0.179;
           0.0131  -0.0148  0   0    0.00567;
          -0.0350   0.0392  0   0    0.00265;
           0.00323 -0.00373 0   0   -0.00633;
          -0.00170  0.00201 0   0    0.00435];

    Cm = [0 0 1.86 -0.396 0.318 -0.00136 0.00139 0.000702;
          0 0 2.28 0.354 -0.0741 -0.00396 -0.00102 -0.000229;
          0 1 0.00102 -0.000119 -0.000498 0 -0.000203 -0.000131;
         -1 0 -0.00119 -0.000106 0.000460 0 0.000139 -0.000170];


    Ap = Am;
    Bp = Bm/.75;
```

```
        Cp = Cm;
        Bd =  [1 1 1 1 1 1 1 1]';
        Dp = [0 0 0 0]';

end
```

B.2.4 MPCGenerateSP.m

```
function [r d p] = MPCGenerateSP(Np,Nc,r0,rf,d0,df,p0,pf)
%Generate step set point and step disturbances for use in simulations
%Usage: [r d p] = MPCGenerateSP(Np,Nc,r0,rf,d0,df,p0,pf)
    tfinal = Np*(32+5*length(r0));
    r=[];
    d=[];
    p=[];

    for k=1:tfinal+Np

        rk=r0;
        for i=1:length(r0)
            if(k>Np*(2+5*(i-1)))
                rk(i) = rf(i);
            end
        end

        if(k>Np*(2+5*length(r0)))
            dk=df;
        else
            dk = d0;
        end
        if(k>Np*(17+5*length(r0)))
            pk=pf;
        else
            pk = p0;
        end
        r = [r; rk];
        d = [d; dk];
        p = [p; pk];
    end
end
```

B.2.5 MPCGenerateRampSP.m

```
function [r d p] = MPCGenerateRampSP(Np,Nc,r0,rSlope,d0,df,p0,pf)
%Generate ramp set point and step disturbances for use in simulations
%Usage: [r d p] = MPCGenerateRampSP(Np,Nc,r0,rSlope,d0,df,p0,pf)
    tfinal = Np*(32+5*length(r0));
    r=[];
```

```
    d=[];
    p=[];

    for k=1:tfinal+Np
        rk=r0;
        for i=1:length(r0)
            if(k>Np*(32+5*(i-1)))
                rk(i) = rSlope(i)*(k-Np*(32+5*(i-1)));
            end
        end

        if(k>Np*2)
            dk=df;
        else
            dk = d0;
        end
        if(k>Np*17)
            pk=pf;
        else
            pk = p0;
        end
        r = [r; rk];
        d = [d; dk];
        p = [p; pk];
    end
end
```

B.2.6 MPCDeltas.m

```
function [delta0 delta1 delta2]=MPCDeltas(B,Np,Nc)
%Generates the delta matrices required for relating
%  standard and incremental output equations
%Usage: [delta0 delta1 delta2]=MPCDeltas(B,Np,Nc)
% delta u = delta0*u+delta1*u(k-1)
% u(k) = delta0^-1*delta u(k)+delta2*u(k-1)
    nu = size(B,2);
    delta0=[];
    delta1=eye(nu);
    delta2=[];
    for i=1:Nc
        if(i~=1)
            delta1 = [delta1 ; zeros(nu)];
        end
        delta2 = [delta2 ; eye(nu)];
    end
    for i=1:Nc
```

```
        rd=[];
        for j=1:Nc
            if(i==j)
                rd=[rd eye(nu)];
            elseif i==(j+1)
                rd=[rd -eye(nu)];
            else
                rd=[rd zeros(nu)];
            end
        end
        delta0=[delta0 ; rd];
    end
end
```

B.2.7 MPCMatsI.m

```
function [AI BI CI CI0 DI EI FI MI NI] =MPCMatsI(A,B,C,Np,Nc)
%Generate augmented matrices for non-incremental state models
%Usage:  [AI BI CI CI0 DI EI FI MI NI] =MPCMatsI(A,B,C,Np,Nc)
    nx=size(A,1);
    nu=size(B,2);
    ny=size(C,1);

    AI=[];
    BI=[];
    CI=[];
    CI0=[];
    DI=[];
    EI=[];
    FI=[];
    MI=[];
    NI=[];

    TF=0;
    TB=eye(nx);
    for i=1:Np
        AI = [AI ; A^i];
        CI = [CI ; C*A^i];
        CI0 = [CI0 ; C*(A^i-eye(nx))];
        EI = [EI ; eye(ny)];
        TF = TF + A^(i-1);
        FI = [FI ; C*TF];
        MI = [MI ; eye(ny)];

        rB=[];
        rD=[];
```

```
        rN=[];
        for j=1:Nc
            if j<i
                if(j==Nc)
                    TB=TB+A^(i-Nc);
                    rB = [rB TB*B];
                    rD = [rD C*TB*B];
                else
                    rB = [rB A^(i-j)*B];
                    rD = [rD C*A^(i-j)*B];
                end
            elseif j==i
                rB=[rB B];
                rD=[rD C*B];
            else
                rB=[rB zeros(nx,nu)];
                rD=[rD zeros(ny,nu)];
            end
        end
        for j=1:Np
            if j<i
                rN = [rN eye(ny)];
            else
                rN = [rN zeros(ny)];
            end
        end

        BI = [BI ; rB];
        DI = [DI ; rD];
        NI = [NI ; rN];
    end
end
```

B.2.8 MPCMatsII.m

```
function [AIIa AIIad BII CII CII0 DII EII MII NII] = MPCMatsII(A,B,C,Np,Nc)
%Generate augmented matrices for incremental state models
%Usage: [AIIa AIIad BII CII CII0 DII EII MII NII] = MPCMatsII(A,B,C,Np,Nc)
    nx=size(A,1);
    nu=size(B,2);
    ny=size(C,1);

    AIIa=[];
    AIIad=[];
    BII=[];
    CII0=[];
```

```
CII=[];
DII=[];
EII=[];
MII=[];
NII=[];

TA=eye(nx);
for i=1:Np
    TA=TA+A^i;
    AIIa = [AIIa ; TA];
    AIIad = [AIIad; eye(nx)-TA];
    CII0 = [CII0 ; C*TA];
    CII = [CII ; C*(eye(nx)-TA)];
    EII = [EII ; eye(ny)];
    MII = [MII ; eye(ny)];

    TB=B;
    rB=[];
    rD=[];
    rN=[];
    if i>Nc+1
        for j=Nc+1:i-1
            TB=TB+A^(j-Nc)*B;
        end
    end
    for j=1:Nc %Build rows backwars so summations work out
        if(j<Nc-i+1)
            rB=[zeros(nx,nu) rB];
            rD=[zeros(ny,nu) rD];
        elseif(j==Nc-i+1)
            rB=[B rB];
            rD=[C*B rD];
        else
            TB=TB+A^(i-Nc-1+j)*B;
            rB=[TB rB];
            rD=[C*TB rD];
        end
    end
    for j=1:Np
        if(j<i)
            rN=[rN eye(ny)];
        else
            rN=[rN zeros(ny)];
        end
    end
```

```
        BII = [BII ; rB];
        DII = [DII ; rD];
        NII = [NII ; rN];
    end
end
```

B.2.9 MPCGains.m

```
function [Ky Kz Kw] = MPCGains(stateEqn, method, D, N, delta0,Q,R,S,T)
%Determine gains
%Usage: [Ky Kz Kw] = MPCGains(stateEqn, method, D, N, delta0,Q,R,S,T)

    %Set gradient of y with respect to u (useful for calculating gains)
    if method==2 || method ==3
        if(stateEqn==1)
            grad=D;
        else
            grad=D*inv(delta0);
        end
    else
        if(stateEqn==1)
            grad=D*delta0;
        else
            grad=D;
        end
    end


    Kd = grad'*(Q + N'*S*N + N'*N'*T*N*N)*grad+R;
    Ky = inv(Kd)*grad'*Q;
    Kz = inv(Kd)*grad'*N'*S;
    Kw = inv(Kd)*grad'*N'*N'*T;
end
```

REFERENCES

[1] T. F. Junge and C. Schmid, "Web-based remote experimentation using a laboratory-scale optical tracker," in *Proceedings of the American Control Conference*, vol. 4, Chicago Illinois, June 2000, pp. 2951–2954.

[2] D. Gillet, C. Salzmann, R. Longchamp, and D. Bonvin, "Telepresence: An opportunity to develop practical experimentation in automatic control education," in *European Controls Conference*, Brussels Belgium, July 1997.

[3] D. Gillet, F. Geoffroy, K. Zeramdini, A. V. Nguyen, Y. Rekik, and Y. Piguet, "The cockpit: An effective metaphor for web-based experimentation in engineering education," *International Journal of Engineering Education*, vol. 19, no. 3, pp. 389–397, 2002.

[4] D. Gillet and G. Fakas, "eMersion: A new paradigm for web-based training in engineering education," in *International Conference on Engineering Education*, Oslo Norway, July 2002.

[5] A. Bhandari and M. H. Shor, "Access to an instructional control laboratory experiment through the world wide web," in *Proceedings of the American Control Conference*, Philadelphia, June 1998.

[6] H. Latchman, C. Salzmann, S. Thotapilly, and H. Bouzekri, "Hybrid asynchronous and synchronous learning networks in distance education," in *International Conference on Engineering Education*, Rio de Janeiro, Brazil, 1998.

[7] B. Armstrong and R. Perez, "Controls laboratory program with an accent on discovery learning," *IEEE Control Systems Magazine, February*, pp. 1–20, 2001.

[8] D. R. Yang and J. Lee, "Java control module," http://dot.che.gatech.edu/information/research/issicl/che4400/javamodule.html, Georgia Institute of Technology, Atlanta, GA.

[9] W. Messner and D. Tilbury, "Control tutorials for matlab," http://www.engin.umich.edu/group/ctm/, University of Michigan, Ann Arbor, MI.

[10] J. W. Overstreet and A. Tzes, "Internet-based client/server virtual instruments designs for real time remote-access control engineering laboratory," in *Proceedings of the American Control Conference*, vol. 2, San Diego, 1999, pp. 1472–1476.

[11] C. Schmid, "The virtual control lab VCLab for education on the web," in *Proceedings of the American Control Conference*, vol. 2, Philadelphia, 1998, pp. 1314–1318.

[12] O. D. Crisalle and H. A. Latchman, "Virtual control laboratory for multidisciplinary engineering education," 1993, nSF Award No. DUE-9352523, proposal funded by the National Science foundation under the Instrumentation and Laboratory Improvement Program.

[13] D. Gillet, C. Salzmann, H. Latchman, and O. Crisalle, "Recent advances in remote experimentation," in *Proceedings of the American Control Conference*, vol. 4, Chicago Illinois, June 2000, pp. 2955–2956.

[14] X. Vilalta, D. Gillet, and C. Salzmann, "Contribution to the definition of best practices for the implementation of remote experimentation solutions," in *IFAC Workshop on Internet Based Control Education IBCE'01*, Madrid, Spain, December 2001.

[15] B. Kuo, *Automatic Control Systems*, 1995.

[16] W. Luyben, *Process Modeling Simulation and Control*, 1990.

[17] K. Ogata, *Modern Control Engineering*, 2002.

[18] B. Ogunnaike and W. H. Ray, *Process Dynamics, Modeling, and Control*, 1994.

[19] G. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 4th ed. Prentice Hall, 2001.

[20] D. Seborg, T. Edgar, and D. Mellichamp, *Process Dynamics and Control*, 2nd ed. Wiley, 2004.

[21] N. Nise, *Control Systems Engineering*, 4th ed. Wiley, 2004.

[22] R. Stefani, B. Shahian, C. Savant, and G. Hostetter, *Design of Feedback Control Systems*, 4th ed. Oxford University Press, 2002.

[23] S. J, S. Dormido, R. Pastor, and F. Esquembre, "Interactive learning of control concepts using easy java simulations," in *IFAC Workshop on Internet Based Control Education IBCE04*, Grenoble, France, 2004.

[24] C. C. Ko, B. M. Chen, and J. Chen, *Creating Web-Based Laboratories*. Springer-Verlag, 2004.

[25] S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*. McGraw-Hill, 1990.

[26] P. C. Wankat and F. S. Oreovicz, *Teaching Engineering*. McGraw-Hill, 1993.

[27] A. Lopez, P. Murrill, and C. Smith, "Controller tuning relationships based in integral performance criteria," *Instrumentation Technology*, vol. 14, 1967.

[28] C. Smith and A. Corripio, *Principles and Practice of Automation Control*, 1997.

[29] J. D. P. G. Franklin and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 2001.

[30] W. Brogan, *Modern Control Theory*, 1991.

[31] P. Belanger and W. Luyben, "Design of low-frequency compensator for improvement of plantwide regulatory performance," *Ind. Eng. Chem. Res.*, vol. 36, p. 5359.

[32] R. Monroy-Loperena, I. Cervantes, A. Morales, and J. Alvarez-Ramirez, "Robustness and parametrization of the proportional plus double-integral compensator," *Ind. Eng. Chem. Res.*, vol. 38, p. 2013.

[33] K. R. Muske and T. Badgwell, "Disturbance modeling for offset-free linear model predictive control," *Journal of Process Control*, vol. 12, p. 617.

[34] G. Pannocchia and J. Rawlings, "Disturbance models for offset-free model-predictive control," *AIChE Journal*, vol. 49, no. 2, p. 426.

[35] G. Elkaim and R. Kelbley", "A lightweight formation control methodology for a swarm of non-holonomic vehicles," *IEEE Aerospace Conference*, 2006.

[36] E. Justh and P. Krishnaprasad", "Equilibria and steering laws for planar formations," *Systems and Control Letters*, vol. 52, p. 25.

[37] D. Buccieri, D. Perritaz, P. Mullhaupt, Z. Jiang, and D. Bonvin, "Velocity scheduling controller for a nonholonomic mobile robot: Theoretical and experimental results," *IEEE Aerospace Conference*, 2006.

[38] J. Daafouz, P. Riedinger, and C. Iung, "Stability analysis and control synthesis for switched systems: A switched lyapunov function approach," *IEEE Journal of Automatic Control*, vol. 47, no. 11, 2002.

[39] D. Liberzon, *Equilibria and steering laws for planar formations*, 2003.

BIOGRAPHICAL SKETCH

Christopher Scott Peek was born in 1980 in Midlothian, Virginia. He grew up in the Richmond area graduating from Chesterfield County Mathematics and Science High School at Clover Hill in 1999. He earned his B.S. in chemical engineering at the University of Virginia. In August 2003 he joined the University of Florida, where he later became a member of the Process Control Research Group under the guidance of Dr. Oscar D. Crisalle. As a member of the group, he investigated research topics on predictive control from both theoretical and applied perspectives.